Universitat de les Illes Balears

Doctorat en Informàtica

PRIVACY-PROTECTING SYSTEMS FOR ELECTRONIC COMMERCE ON MOBILE DEVICES

AUTOR:
ANDREU PERE ISERN DEYÀ

DIRECTORS:
M. MAGDALENA PAYERAS CAPELLÀ
MACIÀ MUT PUIGSERVER

Departament de Ciències Matemàtiques
i Informàtica

Palma, Octubre 2013

Universitat de les Illes Balears

PRIVACY-PROTECTING SYSTEMS FOR ELECTRONIC COMMERCE ON MOBILE DEVICES

ANDREU PERE ISERN DEYÀ

Departament de Ciències Matemàtiques i Informàtica

Palma, Octubre 2013

M. Magdalena Payeras Capellà, professora del Departament de Ciències Matemàtiques i Informàtica de la Universitat de les Illes Balears

FA CONSTAR:

que la present memòria *Privacy-protecting Systems for Electronic Commerce on Mobile Devices* presentada per *Andreu Pere Isern Deyà* per optar al grau de Doctor en Informàtica, ha estat realitzada sota la seva direcció i compleix els requisists per a ser considerada com a tesi doctoral.

Firma i data

Macià Mut Puigserver, professor del Departament de Ciències Matemàtiques i Informàtica de la Universitat de les Illes Balears

FA CONSTAR:

que la present memòria *Privacy-protecting Systems for Electronic Commerce on Mobile Devices* presentada per *Andreu Pere Isern Deyà* per optar al grau de Doctor en Informàtica, ha estat realitzada sota la seva direcció i compleix els requisists per ser considerada com a tesi doctoral.

Firma i data

CONTENTS

Contents v

List of Figures xiii

List of Tables xvii

Acronyms xix

Resum xxiii

Abstract xxv

I Introduction 1

1 Introduction 3
- 1.1 Dissertation Objectives . . . 4
- 1.2 Dissertation Achievements . . . 5
  - 1.2.1 Micropayment . . . 5
  - 1.2.2 Multicoupons . . . 6
  - 1.2.3 Automatic Fare Collection System . . . 7
  - 1.2.4 Colored Petri Nets and Automatic Formal Analysis . . . 8
  - 1.2.5 Group Signatures Implementation and Performance Analysis 9
  - 1.2.6 Implementation and Performance Evaluation of Privacy-protecting Schemes . . . 10
- 1.3 Common Security Properties Requirements . . . 10
- 1.4 Outline of this Dissertation . . . 12
- 1.5 Contributions and Funding . . . 12

2 Background on Cryptographic Tools 13
- 2.1 Hash Chains . . . 13

2.2 Blind and Partially Blind Signature 14
2.2.1 Overview of Blind Signature 14
2.2.2 Overview of Partially Blind Signatures 15
2.2.3 Review the RSA-based Partially Blind Signature due to Chien et al. 16
2.3 Group Signatures 17
2.3.1 Overview of Group Signatures 18
2.3.2 Applications 20
2.3.3 Review Boneh et al. and Ateniese et al. Group Signature Schemes 21

II Contribution to Privacy-protecting Solutions for e-Payment and e-Ticketing 27

3 *μEasyPay*: Micropayment Scheme for Low Value Purchases 29
3.1 Introduction 30
3.2 Overview: Scenario and Security Model 32
3.2.1 Micropayment Scenario 32
3.2.2 Functional Model 33
3.2.3 Security Model 34
3.2.4 Unforgeability 36
3.2.5 Anonymity 37
3.2.6 Unlinkability 38
3.2.7 Untraceability 39
3.2.8 Microcoupon Reuse Avoidance 40
3.2.9 Overspending Protection 41
3.2.10 Fairness 42
3.3 Related Work 44
3.4 *μEasyPay*: Micropayment Scheme 46
3.4.1 Our Proposal in a Nutshell 46
3.4.2 Coin Structure 48
3.5 The Full Specification of the Micropayment 49
3.5.1 System Setup 49
3.5.2 Initial Request Protocol 50
3.5.3 Withdrawal Protocol 51
3.5.4 Spend Protocol 53
3.5.5 Deposit Protocol 53
3.5.6 Refund Protocol 56
3.6 Micropayment Application Example: Pay-per-use Location-Based Services 57

3.6.1 Overview about Location-Based Services and Their Security 57
3.6.2 Payment Methods to Access LBS 58
3.6.3 Legal Framework on Privacy and LBS 59
3.7 Conclusions 62

4 $\mathscr{MC}-2\mathscr{D}$: Electronic Multicoupons Scheme 63
4.1 Introduction 64
4.2 Overview: Scenarios and Security Model 66
4.2.1 The Multicoupon Scenario 66
4.2.2 Security Model 67
4.2.3 Unforgeability 69
4.2.4 Unlinkability 70
4.2.5 Anonymity of Customers 71
4.2.6 Coupon Reuse Avoidance 72
4.2.7 Anonymity Revocation of Misbehaving Customers 73
4.2.8 Disaffiliation of Merchants 74
4.2.9 Unsplittability Protection 75
4.3 Related Work 76
4.4 $\mathscr{MC}$-$2\mathscr{D}$: A Multicoupon Scheme 78
4.4.1 Our Proposal in a Nutshell 79
4.4.2 The Structure of a Multicoupon 80
4.5 The Full Specification of the Multicoupon Proposal 82
4.5.1 System Setup 82
4.5.2 Affiliation/Disaffiliation 83
4.5.3 Group Manager Registration 83
4.5.4 Issue Protocol 83
4.5.5 Multiredeem Protocol 85
4.5.6 Claim Protocol 87
4.5.7 Refund Protocol 87
4.6 Analytical Efficiency Comparison with Armknecht Scheme 88
4.7 Conclusions 89

5 Automatic Fare Collection Scheme 93
5.1 Introduction 93
5.2 Overview: Automatic Fare Collection Systems 95
5.2.1 Scenario 95
5.2.2 Security Model 96
5.2.3 Unforgeability 99
5.2.4 Non Repudiation of Origin 100
5.2.5 Anonymity 100
5.2.6 Unlinkability 100

5.2.7 Untraceability . . . 101
5.2.8 Overspending Avoidance . . . 101
5.2.9 Revocable Anonymity of Misbehaving Users . . . 102
5.3 Related Work . . . 102
5.4 General Automatic Fare Collection Scheme . . . 104
5.4.1 Proposal Overview . . . 104
5.4.2 Tickets Structure . . . 105
5.5 General Automatic Fare Collection Proposal Specification . . . 106
5.5.1 Initial System Setup . . . 106
5.5.2 User Registration . . . 107
5.5.3 System entrance . . . 108
5.5.4 System exit . . . 108
5.5.5 First User Claim . . . 111
5.5.6 Second User Claim . . . 111
5.5.7 First Provider Claim . . . 112
5.5.8 Second Provider Claim . . . 112
5.6 Apply General AFC Scheme to Time and Distance-Based Scenarios . 115
5.6.1 Colluding Attack: Exchange of Tickets between Malicious Users . . . 116
5.7 Enhanced Distance-Based AFC . . . 117
5.7.1 Linkability of Signatures . . . 118
5.7.2 AFC System Specification . . . 119
5.8 Conclusions . . . 122

III Security Analysis of Privacy-Protecting Proposals 123

6 Automatic Formal Analysis of Secure Protocols 125
6.1 Introduction . . . 125
6.2 Process Modeling Languages and Model Checkers . . . 127
6.3 Petri Net Definition . . . 128
6.4 Colored Petri Nets . . . 130
6.5 General Assumptions and Methodology . . . 131
6.6 Case Example I: FPH Contract Signing Scheme . . . 133
6.6.1 Overview on Contract Signing Schemes . . . 133
6.6.2 Ideal Features of a Contract Signing Scheme . . . 134
6.6.3 Description of the FPH Contract Signing Scheme . . . 135
6.6.4 ECS1 Contract Signing Scheme due to Micali et al. . . . 138
6.7 Modeling the FPH Scheme with CPN . . . 138
6.7.1 Query Functions . . . 143
6.8 Formal Evaluation of FPH Using the CPN Methodology . . . 145

6.8.1 Bao's First Attack . . . 145
6.8.2 Bao's Second Attack . . . 146
6.8.3 Bao's Third Attack . . . 148
6.8.4 Sornkhom's First Attack . . . 149
6.8.5 Sornkhom's Second Attack . . . 149
6.8.6 Full Formal Analysis of the FPH Scheme with CPN . . . 151
6.9 Case Example II: Multi-Party Contract Signing Scheme . . . 155
6.10 Conclusions . . . 158

7 Security Analysis of Proposed Privacy-Protecting Schemes 161
7.1 Formal and Automatic Security Analysis of the Micropayment Scheme 161
7.1.1 Unforgeability . . . 162
7.1.2 Anonymity of Customers . . . 163
7.1.3 Unlinkability . . . 164
7.1.4 Untraceability . . . 165
7.1.5 Microcoupon Reuse Avoidance . . . 165
7.1.6 Overspending Protection . . . 166
7.1.7 Fairness . . . 167
7.1.8 Automatic Formal Fairness Verification with Colored Petri Net 168
7.2 Formal Security Analysis of the Multicoupon Scheme . . . 176
7.2.1 Unforgeability . . . 176
7.2.2 Unlinkability . . . 178
7.2.3 Anonymity of Customers . . . 179
7.2.4 Coupon Reuse Avoidance . . . 180
7.2.5 Anonymity Revocation of Misbehaving Customers . . . 181
7.2.6 Merchant disaffiliation . . . 181
7.2.7 Unsplitting . . . 182
7.3 Security Analysis of the Automatic Fare Collection System . . . 184
7.3.1 Unforgeability . . . 184
7.3.2 Non Repudiation of Origin . . . 185
7.3.3 Anonymity . . . 185
7.3.4 Unlinkability . . . 186
7.3.5 Untraceability . . . 186
7.3.6 Overspending Avoidance . . . 187
7.3.7 Anonymity Revocation of Misbehaving Users . . . 187

IV Implementations and Performance Analysis 189

8 Considerations and Implementation Framework 191
8.1 Introduction . . . 191

8.2 Considerations on the Deployment of e-Commerce Solutions 192
8.2.1 Server Platform 193
8.2.2 Client Application Platform 196
8.2.3 Communication Technologies 197
8.2.4 Messages 198
8.3 Implementation Framework 199
8.3.1 Prototype Environment 199
8.3.2 Secure Storage of Credentials 201
8.3.3 Tools, Libraries and Frameworks 201
8.3.4 Selection of a Common Security Strength 203
8.3.5 Testing Devices 204
8.4 Conclusions 206

9 Group Signature: Comparing BBS and ACJT schemes 207
9.1 Introduction 207
9.2 Implementation 208
9.2.1 BBS: Selection of Pairing Parameters 209
9.2.2 ACJT: Selection of Parameters 213
9.2.3 Group Key Distribution 214
9.3 Performance Discussion 216
9.3.1 Test Scenario 216
9.3.2 Group Manager Computation Analysis 216
9.3.3 Group Member Computation Analysis 218
9.3.4 Storage Requirements Analysis 221
9.3.5 Summarizing Performance Results 222
9.4 Concluding Remarks 223

10 *μEasyPay*: Implementation and Performance Analysis 225
10.1 Introduction 225
10.2 Implementation 226
10.2.1 Partially Blind Signature Implementation 227
10.2.2 Customer Application and User Experience 227
10.2.3 Server Entities 230
10.2.4 Messages 230
10.3 Performance Evaluation 230
10.3.1 Test Scenario 230
10.3.2 Discussion of Results 231
10.4 Conclusion 235

11 $\mathscr{MC}-2\mathscr{D}$: Implementation and Performance Analysis 237
11.1 Introduction 237

11.2 Implementation . . . 238
11.2.1 Customer Application and User Experience . . . 238
11.3 Performance Comparision with the Armknecht's Scheme . . . 240
11.4 Performance Discussion . . . 241
11.4.1 Test Scenario . . . 241
11.4.2 Time response and Network Overhead . . . 243
11.4.3 Analyzing Influence Factors on Computational Performance 244
11.5 Conclusions . . . 247

12 AFC: Implementation and Performance Analysis 249
12.1 Introduction . . . 249
12.2 Implementation . . . 250
12.2.1 Client Side . . . 250
12.2.2 Server Entities . . . 251
12.2.3 Messages . . . 251
12.2.4 Tickets . . . 251
12.3 Performance Evaluation . . . 251
12.3.1 Test Scenario . . . 251
12.3.2 Discussion of Results . . . 253
12.3.3 Size of Tickets and Exchanged Messages . . . 257
12.4 Conclusions . . . 259

V Concluding Remarks 261

13 Conclusions and Future Work 263
13.1 *μEasyPay*: Contribution to Micropayments . . . 264
13.2 $\mathscr{MC}-2\mathscr{D}$: Contribution to Electronic Multicoupons . . . 264
13.3 AFC: Contribution to Automatic Fare Collection Systems . . . 266
13.4 Automatic Formal Analysis of Secure Protocols with Colored Petri Nets . . . 267
13.5 Implementation and Performance Evaluation of Group Signatures . . 268
13.6 Implementation and Performance Evaluation of Privacy-protecting Schemes . . . 268

14 List of Publications Supporting The Dissertation 271

Bibliography 275

List of Figures

3.1 Micropayment scenario. Entities and protocol flow among them. 47
3.2 Life cycle of coins. 48
3.3 Coin structure ($\mathbb{C}$). 48
3.4 `InitialReq` protocol flow. 51
3.5 `Withdrawal` protocol flow. 52
3.6 `Spend` protocol flow. 54
3.7 `Deposit` protocol flow. 55
3.8 `Refund` protocol flow. 56

4.1 Our proposed scenario. Entities relationship and protocol flow. 79
4.2 Life cycle of multicoupon. 80
4.3 $\mathbb{MC}^{2D}$ structure composed by $\mathbb{MC}_{\omega}$ and $\mathbb{MC}_{\mathscr{PBS}}$. 81
4.4 `Issue` protocol flow. 84
4.5 `Multiredeem` protocol flow. 86
4.6 `Claim` protocol flow. 88

5.1 General AFC scenario and relations among parties. 105
5.2 `URegistrationMG` protocol flow. 107
5.3 `URegistrationMP` protocol flow. 108
5.4 `SysEntrance` protocol flow. 109
5.5 `SysExit` protocol flow. 110
5.6 `UClaim1` protocol flow. 111
5.7 `UClaim2` protocol flow. 112
5.8 `PClaim1` protocol flow. 113
5.9 `PClaim2` protocol flow. 114
5.10 AFC scenario with separated directions and stations, with and without ξ parameter setting the direction of movement. 116
5.11 AFC scenario without separated directions and stations. Users choose the direction after entering the service. 117
5.12 Enhanced `SysEntrance` protocol flow for the new scenario. 120

5.13 Enhanced `SysExit` protocol flow for the new scenario. 121

6.1 A Colored Petri Net example built using CPNTools. 131
6.2 State space tool from the CPNTools software. 133
6.3 Marking tree generated by using the CPNTools. 133
6.4 Definition of tokens with type and value (color). 140
6.5 Top level scheme. 141
6.6 **A**'s entity level. 142
6.7 **B**'s entity level. 142
6.8 Process level of **T** party modeling the *cancel* protocol. 143
6.9 Search query functions developed in order to search for commits into the party's databases. 144
6.10 Bao's first attack query results, $\mathbf{A}_m$ database contents and session configuration. 146
6.11 Bao's second attack query results, $\mathbf{A}_m$ database contents and session configuration. 147
6.12 Bao's third attack query results, $\mathbf{B}_m$ database contents and session configuration. 148
6.13 Sornkhom's first attack query results, parties' database contents and session configuration. 150
6.14 Sornkhom's second attack query results, $\mathbf{A}_m$'s database contents and session configuration. 151
6.15 Query functions results over the model with $(\mathbf{A}_m, \mathbf{B}_m)$ session. 152
6.16 Top level CPN modeling the multi-party FPH scheme. 156
6.17 Entity level of **T** modeling resolution in the multi-party FPH scheme. . . 157
6.18 The perceived utility and the cost to model a secure scheme depending on its underlying complexity. 159

7.1 Top-level net modeling the micropayment scheme. 169
7.2 CPN modeling part of the customer entity. 170
7.3 Corrected `Deposit` protocol after step-by-step checking with CPN (left) vs. flawed `Deposit` protocol (right). 171
7.4 Corrected `Refund` protocol after step-by-step checking with CPN (left) vs. flawed `Refund` protocol (right). 172
7.5 Query `CustomerAdvantage`. 173
7.6 Queries do not detect any unfair marking in the micropayment including resolutions and support by $\mathscr{B}$. 173
7.7 Queries output some unfair markings in the micropayment without considering resolutions. 173

9.1 X.509 Certificate Profile. 214

9.2 Public Key Info. 215
9.3 ASN.1 Group Public Key. 215
9.4 Performance results related to the BBS group manager who performs the $KeyGen^G$ algorithm on the laptop. 217
9.5 Time to trace a signature to the corresponding signer. 217
9.6 Performance results related to a user who performs the $Sign^G$ and the $Verify^G$ algorithms from BBS (using types A, A1, D, E and F pairings) and ACJT schemes. 218
9.7 Elapsed time to perform costlier pairing operations on testing devices. . . 221

10.1 System workflow: interactions between customer application, bank and merchant servers. 226
10.2 Code snippets of the implemented $\mathscr{PBS}$ scheme. 228
10.3 Screenshots from the developed Android customer application. (a) DataReqWithdrawalActivity view before to select a targe merchant. (b) SpendActivity view before execute the `Spend` protocol, showing the current customer's localization. (c) SpendActivity view after an execution of the `Spend` protocol, with results localized on map. (d) SettingsActivity view to configure some parameters from the application. 229
10.4 Test scenario. 231
10.5 Time required to comple the `Withdrawal` protocol depending on the RSA modulus length. 232
10.6 `Withdrawal` protocol performance depending on the chain length for each smartphone (considering RSA modulus of 1024 bits). 233
10.7 `Spend` protocol performance depending on the number of coupons released at once for each smartphone. 234

11.1 Screenshots from the developed Android customer application. (a) *IssueNewMC* view. (b) *ManageMC* view. (c) *Settings* view. 239
11.2 Considered performance testing scenario. 242
11.3 Total message lenght for each protocol. 243
11.4 Total response time for each protocol (Desire with Wi-Fi). 244
11.5 `GMRegistration` protocol: processing and encoding/decoding in function of smartphone. 245
11.6 `Issue` protocol: processing and encoding/decoding in function of smartphone. 245
11.7 `Multiredeem` protocol: processing and encoding/decoding in function of smartphone. 246
11.8 Networking response time for each protocol. 246

12.1 Entrance and exit tickets with XML syntax. 252

12.2 Total time expended for each protocol version over both smartphones. . . 253
12.3 Time to run each scheme phase considering precomputation disabled. . . 254
12.4 Time to run each scheme phases considering precomputation enabled. . . 255
12.5 Time flow for both AFC schemes and smartphones. 258
12.6 Size of tickets and exchanged messages during `SysEntrance` and `SysExit` protocols (bytes). 259

List of Tables

3.1 Micropayment solutions - A comparative analysis 45
3.2 Notation used along the protocol description. 50

4.1 Multicoupons solutions - A comparative analysis 77
4.2 Notation used in the protocol description. 82
4.3 Multi-merchant solutions - An analytical evaluation comparison. 89

5.1 AFC Proposals - A comparative analysis. 103
5.2 Information included in the entrance ticket ($t_{in}{}^{*}$) 105
5.3 Information included in the exit ticket ($t_{out}{}^{*}$) 106
5.4 General notation used along the protocol description. 106
5.5 Information included in the entrance ticket ($t_{in}{}^{*}$) in case it has to be applied to the distance-based scenario with common entrance and exit stations. 115

6.1 The notation used to describe the Ferrer-Payeras-Huguet (FPH) scheme. 136
6.2 The *exchange protocol* between **A** and **B**. 136
6.3 The *cancel protocol* between **A** and **T**. 136
6.4 The *Finish protocol* between **B** and **T**. 137
6.5 Micali's ECS1 scheme definition. 138
6.6 Bao's first attack trace to the ECS1 scheme. 145
6.7 Bao's second attack trace to the ECS1 scheme. 147
6.8 Bao's third attack trace to the ECS1 scheme. 148
6.9 Sornkhom's second attack trace to the ECS1 scheme. 150
6.10 List of the markings corresponding to the three cases with contradictory evidences (where two markings correspond to the same case). 153
6.11 Scenarios without contradictory evidences. 154

7.1 List of likely unfair states output by CPNTools. 174

8.1 Server models solutions: a comparative analysis. 195
8.2 Frameworks for developing mobile apps: a comparative analysis. 197
8.3 Considered testing devices. 205

9.1 Embedding degree and theoretical lower bound size of q for each considered type of pairing (equivalent to 80 bits of security strength). 211
9.2 Groups order and length of a random element within each group collected using the jPBC library on input of curves generated by PBC. . . . 212
9.3 ACJT parameter selection. 213
9.4 Number of precomputable operations along the signature and verification algorithms. 219
9.5 Percentage of improvement if precomputation is enabled during the signature generation and verification on both Laptop and Desire. 220
9.6 Storage requirements for users and group manager. 222

11.1 Multi-merchant solutions - An implementation comparison (in seconds). 240
11.2 Network features of each communication path (estimated average). . . . 242

ACRONYMS

ICT Information Communications Technologies 3

ITU International Telecommunication Union 3

B2C Business to Customer 4

NSA National Security Agency 4

BBS Boneh-Boyen-Shachan 20

SDH Strong Diffie-Hellman 20

DDH Decisional Diffie-Hellman 24

LBS Location-Based Service 29

CPN Colored Petri Net 31

CA Certification Authority 35

TTP Trusted Third Party 42

NR non-repudiation 42

NFC Near Field Communications 46

PIR Private Information Retrieval 58

AFC Automatic Fare Collection 93

NRO non-repudiation of origin 100

ZKP Zero-Knowledge Proof 107

FPH Ferrer-Payeras-Huguet 126

UML Unified Modeling Language 127

AVISPA Automated Validation of Internet Security Protocols and Applications 128

SPIN Simple Promela Interpreter 128

PROMELA PROcess ModEling LAnguage 128

CRYPTYC Cryptographic Protocol Type Checker 128

SaaS Software-as-a-Service 194

PaaS Platform-as-a-Service 194

IaaS Infrastructure-as-a-Service 194

EC2 Elastic Cloud Computing 195

S3 Simple Storage Service 195

AWS Amazon Web Services 195

ASN.1 Abstract Syntax Notation One 198

XML eXtensible Markup Language 198

UI User Interface 200

API Application Programming Interface 201

JCE Java Cryptographic Extension 201

JAXB Java Architecture for XML Binding 202

PBC Pairing-Based Cryptography 202

jPBC Java-Pairing Based Cryptography 202

NIST National Institute of Standards and Technology 203

POM Project Object Model 203

IDE Integrated Development Environment 203

ECC Elliptic Curve Cryptography 210

PKC Public Key Certificate 214

POI Points Of Interest 226

RESUM

Avui dia, estam assistint a un increment important en el coneixement i en la introducció de les Tecnologies de la Informació i les Comunicacions (TIC) en la societat. De cada dia més i més gent acull les noves formes de consumir béns i serveis electrònics a través de l'ús de les noves infrastructures TIC, típicament relacionades en escenaris mòbils. El nombre de dispositius mòbils *sempre connectats* ha experimentat un gran increment degut a la promoció de subscripcions de banda ampla mòbil que permeten als usuaris accedir a la informació quan volen i allà on volen, fins i tot amb models de consum d'informació a temps real. És justament en aquest escenari de mobilitat on moltes àrees de negoci poden rebre excel·lents beneficis de l'increment de l'ús de les TIC. Un d'aquests camps és el comerç electrònic. De fet, d'acord amb un estudi recent de prospecció de mercat, les ventes relacionades amb el comerç electrònic (especialment en el mercat Business to Customer (B2C)) augmentaran globalment a un ritme d'entre el 10% i el 15% des del 2012 fins al 2017. A més, l'estudi també apunta que la gent estirà de cada vegada més acostumada a fer ús dels seus dispositius mòbils per accedir a serveis electrònics relacionats amb el comerç electrònic. No obstant, la gent també exigeix certs nivells de privacitat per a confiar més i estar més actius en l'ús del comerç electrònic.

Encara que actualment el comerç electrònic és un tema molt important i molta gent està ja familiaritzada amb ell, encara li resta un llarg camí per recórrer. Com s'ha dit anteriorment, una de les majors preocupacions dels consumidors i dels venedors és la falta de privacitat i confiança quan fan ús d'Internet, i especialment quan han de realitzar compres a la xarxa o han d'accedir a serveis electrònics. De fet, aquesta és la premisa que ha motivat la recerca recollida en aquesta Tesi. Llavors, l'objectiu d'aquesta Tesi és el de proposar noves i millors solucions que protegeixin la privacitat dels actors implicats per a tractar de resoldre algunes mancances del comerç electrònic i d'aquesta forma incrementar la privacitat, la seguretat i la confiança per accelerar encara més la utilització del comerç electrònic. Entre tots els temes que composen el comerç electrònic, en aquesta Tesi es tracten tres serveis que sofreixen aquesta falta de privacitat i confiança per part dels consumidors i dels venedors: el pagament a canvi de compres de petit valor, l'ús de cupons electrònics

que normalment associen algun tipus de descompte o regal i el tíquets electrònics per accedir a serveis a canvi d'una tarifa basada en l'ús que se li ha donat al servei.

Per tant, en la Tesi es proposen tres noves solucions per a protegir la privacitat dels tres serveis abans esmentats. Les solucions aportades s'han definit per mitjà de models funcionals i de seguretat formals i s'han provat que són segurs després de verificar-ne la seva seguretat d'acord amb les propietats requerides en cada cas. Encara que l'objectiu principal de la Tesi giri al voltant de la proposta teòrica de noves solucions que protegeixin la seguretat, també és important remarcar que les solucions també han de ser pràctiques i eficients per a poder implementar-les i després usar-les en dispositius mòbils actuals. Llavors, el treball que ha conduit a la realització d'aquesta Tesi també inclou el desenvolupament d'implementacions funcionals de les solucions proposades i l'anàlisi del rendiment obtingut sobre dispositius mòbils actuals.

ABSTRACT

Nowadays, we are witnessing an important increase in the knowledge and the introduction rate of Information Communications Technologies (ICT) in society. Day after day, there are more and more people embracing new means to consume electronic goods or services through the utilization of these new ICT infrastructures, usually related to mobile scenarios. The number of *always connected* mobile devices has experienced an important growth due to the release of mobile broadband subscriptions that allow users to access information where and when they want, even with real-time data consumption patterns. In this mobility scenario, there are many business areas that can receive excellent benefits from the increasing use of ICTs. One of these fields is electronic commerce (e-commerce). In fact, according to a recent forecast research, sales due to e-commerce (specially the Business to Customer (B2C) market), will rise globally at an annual rate of between 10% and 15% from 2012 to 2017. Besides, the study also points out that people are more likely to use their mobile devices to access e-commerce services. However, people also want to achieve certain levels of privacy to be more confident and active with e-commerce.

Despite the fact that e-commerce currently is an important topic and that many people are already familiarized with it, e-commerce still has a long way to go. As introduced before, one of the main concerns of customers and merchants is the lack of privacy and trust when they browse Internet, and especially when they have to do on-line purchases or have to access electronic services. Indeed, this is the underlying premise that motivates the research gathered in this dissertation. Therefore, the objective of this dissertation is to propose new and improved privacy-protecting solutions to address some unresolved deficiencies in the e-commerce field to increase privacy, security and trust that would accelerate even more the use of e-commerce by society. Among all topics within the e-commerce field, this dissertation deals with three services that may suffer this lack of privacy and trust from customers and merchants: payment of low value purchases, use of electronic coupons (typically associated to some discounts or gifts) and electronic ticketing to access services in exchange for a fare based on the given use.

Thus, the dissertation proposes three new privacy-protecting solutions that cover

the aforementioned e-commerce tools. Solutions are defined by formal functional and security models and proved secure by means of a security verification according to the required security properties. Although the main objective of this dissertation revolves around the theoretical proposal of new privacy-protecting solutions, it is also important to note that they should be proved practical and efficient to be implemented and afterward used by current mobile devices. So, the work that conducts to this dissertation addresses also the development of functional implementations of proposed privacy-protecting solutions and their performance evaluation on current mobile devices.

En primer lloc, agrair a tots els qui han estat d'una o altra forma al meu costat durant aquests anys, especialment la meva mare, al meu pare, la meva atl·lota, tots els familiars i amics que m'han recolzat i els companys de laboratori que han esdevingut uns amics més. Gràcies!

I en segon lloc, extendre el meu reconeixement als meus directors de tesi i a la resta de professors del grup d'investigació, especialment a na Xisca Hinarejos. També els meus agraïments a n'Arnau Vives i en Jordi Castellà de la URV i a n'Antònia Paniza de la UIB pel seu suport en temes legals. Gràcies!

Als meus padrins, sempre amb jo. Padrina *Cabeta* i Padrí *Roc*.

Part I

Introduction

CHAPTER 1

INTRODUCTION

Information Communications Technologiess (ICTs) taken-up by people in the world is growing year after year with impressive rates. According to the latest ICT Facts and Figures report [1] from the International Telecommunication Union (ITU) released in 2013, almost 40% of the world's population is already online and, in developed countries, this value raises to a 77% of the total population. It represents a growth rate of about a 300% from 2003. Another important and interesting fact is that there are almost as many mobile-cellular subscriptions as people in the world and that the penetration rates stand at a 96% globally. It is also representative how mobile-broadband subscriptions have climbed from 268 million in 2007 to 2100 million in 2013. It represents an average annual growth rate of 40% making the mobile broadband the most dynamic ICT market. Another related fact is the latest statistics about the number of smartphones in use worldwide which suggest that the 1000 million mark was passed at the end of 2012 [2]. It represents about a 46% growth according to the previous study released in 2011. Some of the major Internet firms contributed to it, such as Apple with its iPhone devices and Google with its Android Operation System. So, it is actually clear that users are willing to stay connected (even always connected), making use of new generation smartphones to access applications and services by means of mobile internet subscriptions.

This growth on the level of introduction of ICTs in the world results in people asking for new services and applications to make use of these new access technologies. One of the fields most benefited by these impressive growth rates is electronic

commerce (henceforth e-commerce), specially the Business to Customer (B2C) e-commerce, also known as retail market. In fact, according to predictions from the last forecast published by Forrester [3–5] which covers from 2012 to 2017, online retail sales will rise globally at an annual growth rate between 10% and 15%. It means that in 2017, sales due to e-commerce will reach $370,000 million in the U.S and in Europe it is projected to hit $247,100 million. The forecast also points out the increasing importance of mobile e-commerce in a sense that consumers are more likely to use their phones to access e-commerce services. Therefore, e-commerce nowadays represents an important area of business (and probably even more important in the future) with a huge potential revenue for merchants and great opportunities for customers to achieve a better offer.

Aside from the growth in ICTs adoption and the rise of e-commerce sales, an important concern of customers and merchants is their security and privacy when they are connected to the network. Recently, the revealing of secret U.S. and British government mass surveillance programs, code named PRISM [6, 7], lead by the National Security Agency (NSA), has been remarkable. Indeed, PRISM has been revealed as the major electronic surveillance program to monitor communications and data around the world to guarantee national security. Also, some revealed reports pointed out that major internet actors, such as Facebook, Google, Microsoft, Yahoo and so on, may have collaborated with the NSA, providing data (e-mails, video, photos, documents, search history and logs) about users to the agency [6, 7]. They have rejected these accusations, but the mistrust flies over Internet users.

Therefore, people are more concerned about their privacy when they browse the Internet and especially when they have to make on-line purchases or access on-line services [8–10]. In fact, sometimes the lack of availability of trusted methods negatively affects the customers' willingness to use mobile devices to do the same things they currently do with paper-based systems. The same can be stated for merchants who are concerned about their security when they are operating on-line selling goods or offering access to services. So, both parties involved in e-commerce demand new privacy-protecting methods to take advantage of recent developments in networking and the availability of mobile devices so to transform the way they buy or access services on-line.

1.1 Dissertation Objectives

As stated before, there is often a lack of trust on e-commerce that may impede and slow down the deployment of new business models. Therefore, new privacy-protecting electronic methods must be proposed to improve privacy and security of all involved parties in this business area. There are many topics within the e-commerce field that require privacy enhancements in order to improve the trust on

both sides. Along the work which has conducted to this dissertation, we have identified three services related to e-commerce that may suffer from this lack of trust between customers and merchants. These topics are the payment of low value purchases, the use of electronic coupons typically attaching some discounts or gifts and the electronic ticketing to access services in exchange of a fare based on the given use. All of these privacy-protecting services share some common facets, such as security and privacy requirements, management of money (in an implicit or explicit way) and its use by mobile devices.

The fundamental objective of this dissertation is the proposal of new and improved privacy-protecting schemes for these services and to prove whether these solutions are in fact secure according to security and privacy requirements. Moreover, the work also provides proof demonstrating that the solutions are also practical and implementable in a mobile scenario, considering the use of mobile devices, such as smartphones, tablets, etc. Therefore, this dissertation does not only cover the theoretical design and security analysis of these schemes, but it also wants to provide solutions for the real world. Thus, it implies the job of analyzing the practicability of these solutions by means of performance evaluations once they have been implemented.

1.2 Dissertation Achievements

The following topics have been addressed along this dissertation. They cover the complete cycle of production of a new proposal, i.e. detection of previous flaws or lacks in existing solutions, design of improved schemes, proofs of security, implementation and performance evaluation with current devices.

1.2.1 Micropayment: Paying Low Value Purchases

Nowadays, there are plenty of items that can be bought on-line. Most of them have a worth value that can be paid by common credit or debit cards without it harming a big part from the merchants' revenue due to commissions applied by card intermediaries. However, there a large number of goods and services that have a worth value as low as the very often applied commissions, so that they tend to be more expensive than the item just sold. We can cite some examples, such as intangible selling of goods (newspapers, product reviews, services related to localization, etc.), virtual gifts, in-app sales (sales of virtual items related to gaming, smiles in instant messaging applications, low value subscriptions, etc.) or electronic data (music, videos, documents, etc.). All of these examples involve low value transactions, so transactional costs related to them must never be larger than the value of the item itself. Otherwise, sales will not be profitable for either customers or merchants. Payment solutions involving these kind of exchanges are called micropayments [11–15].

They have different functional and security requirements to full payments involving higher amounts of money. Micropayments lay out an interesting trade-off between security measures and reduction of transactional costs because both requirements are usually opposed.

The contribution to the micropayment topic is the proposal of *μEasyPay*, an efficient and practical payment method to purchase low value items. Customers withdraw from a bank a micropayment coin composed by a configurable number of tokens (microcoupons in the terminology used in this dissertation) in such a way that issuing costs are completely amortized with a really efficient protocol to perform micropayments to merchants. The proposed solution provides anonymity to customers. This way, they cannot be identified by merchants since it is not required to provide any type of information related to their real identity. Moreover, merchants cannot trace activities run by customers, so it avoids the creation of profiles. *μEasyPay* not only protects customers, it also gives merchants the ability to detect and avoid malicious actions performed by misbehaving customers, such as microcoupon reuse or forging attempts.

Particular features of *μEasyPay* allow it to be utilized as a payment method to access Location-Based Service (LBS) providers. In fact, we prove that our solution fits not only privacy properties but also accomplishes all legal requirements stated by the European legal framework in this specific field.

1.2.2 Multicoupons: Obtaining Goods or Services Achieving Discounts or Gifts

Another interesting tool for the e-commerce is the use of coupons that allow customers to obtain goods or services, typically achieving some discounts or gifts. There are examples of success paper-based coupons, such as coupon booklets for restaurants (Ticket restaurant [16, 17]), hotels (Bancotel [18]), etc. By using these systems, both customers and merchants obtain benefits in such a way that the former achieve good discounts and the latter increase their sales. There are some interesting research results in the literature about the electronic equivalent of coupons for e-commerce [19–26]. These kind of schemes are typically referred to as multicoupon schemes because they carry multiple coupons in a single electronic booklet. They have some common security and privacy requirements from micropayments but they usually demand higher levels of protection since they usually must be able to manage large amounts of money. However, there are not real experiences on the deployment of these solutions. It can be due either to the lack of trust and privacy perceived by customers and merchants or because there do not yet exist practical solutions that can be deployed successfully on mobile devices.

Regardless of which are the reasons, the objective of this research line is to provide an electronic coupon solution to fulfill wishes from customers and merchants

referring to trust, privacy and practicability of electronic coupons. So, the contribution to this e-commerce topic is the design of $\mathscr{MC}-2\mathscr{D}$. This solution addresses some unresolved drawbacks of previous proposals and allows customers to redeem coupons from the same multicoupon at different merchants, so $\mathscr{MC}-2\mathscr{D}$ deals with a multi-merchant scenario. It is particularly novel because only one of the previous proposals considers a multi-merchant scenario [25]. However, that solution presents an important security problem and it is not practical. $\mathscr{MC}-2\mathscr{D}$ achieves untraceable and anonymous multicoupons, so nobody can determine who redeems coupons and where coupons are being spent, thanks to the use of group signatures and partial blind signatures. So, a high degree of privacy is provided to customers. Besides, merchants are also protected from possible customer misbehavior in such a way that if a customer tries to reuse or forge a coupon, the customer's anonymity can be revoked and her identity provided to the proper authorities. Moreover, merchants usually demand that customers must not be able to share coupons from a single multicoupon with other customers. Therefore, our solution also includes this wish by means of a method that discourages customers to detach and share coupons. All these security requirements will be properly defined and afterward proved by formal security analysis.

Apart from design of scheme, we also provide strong arguments to prove that our scheme outperforms the previous proposal that deals with the same scenario considered for this research line. Thus, it will be proved that $\mathscr{MC}-2\mathscr{D}$ is simpler and the complexity to implement it and deploy it is less than that previous solution.

1.2.3 Automatic Fare Collection System: Charging Based on the Use of Services

Electronic ticketing is another important topic within e-commerce. There are many applications and services based upon using tickets. Each of them has special security and functional requirements depending on the concrete scenario where it has to be applied. Because the electronic ticket topic is so huge, we have decided to reduce the research scope to focus our efforts on a concrete scenario composed by those services in which users are charged with a fare depending on the use they have made. Solutions that control these services are called Automatic Fare Collection (AFC) schemes [27–33]. Some typical examples of services managed by AFC solutions are public transportation (e.g. train, bus, subway, etc.), closed parking lots or access to public and sport facilities. We can distinguish two different scenarios depending on the parameter used to charge users. This way, there are *time-based* AFC and *distance-based* AFC. In former solutions, the fare is computed according to time users have used the service while in latter the fare has a relation to the covered distance. Regardless of the concrete scenario, AFC solutions should present strict functional requirements focused on fast issuing and validation of tickets, even

in peak hours. It is also important to provide AFC schemes with means to be used on mobile devices, due to the fact that users typically will be in mobility scenarios. Besides, security is an important concern because users want to be anonymous and they also demand that different sessions must not be traced. Similarly to the multicoupon scheme, the AFC solution also makes use of group signature benefits.

The purpose of this research line is the study of current AFC proposals to provide a more secure solution. In fact, a vulnerability has been found in a previous distance-based AFC proposal [34] that arises in some concrete scenarios. Taking advantage of this security flaw, malicious and colluding users collaborate with each other in some concrete scenarios, finally achieving a benefit with the fare to be paid. As a result, colluding users obtain a fare less than the real one according to the enjoyed service. Therefore, with the collaboration of researchers who have originally proposed the distance-based AFC scheme, the contribution to this research line consists on proposing an improved AFC solution to be applied to either time-based or distance-based scenarios avoiding colluding attacks. Besides, the scenario and the security model has been also formalized in the same way as μ*EasyPay* and $\mathscr{MC}-2\mathscr{D}$ solutions do. According to the security model, a security analysis is provided to prove that the AFC solution meets all security requirements.

1.2.4 Colored Petri Nets: Automatic Formal Analysis of Secure Protocols

Formal analysis of secure protocols, schemes and in general, validation of distributed systems, is often a challenging, tedious and time-consuming task. In the literature various methods exist to conduct formal security analysis [35–37], but only some of them are promising methods that allow researchers to automatize this process. A formal methodology emerges from all analyzed methods as probably the best one to automatically verify some properties and to prove whether analyzed protocols are in fact resistant to attacks and vulnerabilities. It is based on the use of Colored Petri Net (CPN) language [35]. CPN derives from the original Petri Net language to model distributed and concurrent systems. CPN, as original Petri Nets, has a strong underlying mathematical background, so it represents a formal method.

The main purpose with this research line is to experiment with CPN to try to provide some insights describing how to model secure protocols and how to perform an automatic verification of them. To do it, based on some previous works on a similar research line [38, 39], we propose a methodology derived from the utilization of CPN to verify the fairness property of a contract signing scheme [40]. First, the CPN model to analyze the two-party version of that contract signing scheme (version where only two signers are involved in the signing protocol) is proposed together with a method to detect a unfair situation in which some party achieves some advantage. Secondly, the model of the multi-party version of the same contract sign-

ing protocol (it involves more than two signers) is also provided but some difficulties will be described to highlight that not all schemes and protocols can be successfully modeled and analyzed with this formal tool. So, it is worthwhile pointing out that we provide information about the applicability of this method to give some insights allowing researchers to distinguish whether this method can be applied or not in advance. Finally, with the knowledge acquired working on this research line, we also present an additional result consisting on the fairness and non-repudiation (NR) formal analysis of *μEasyPay* micropayment scheme using the CPN methodology.

1.2.5 Group Signatures Implementation and Performance Comparison

Group signatures are an interesting cryptographic tool to provide anonymous and non-repudiation (NR) credentials to provide authenticity and anonymity to signers at the same time [41–48]. Summarizing, group signatures involve a group of signers where each of them holds a membership certificate composed by a group key pair. As in other public key cryptographic algorithms, group signatures define a single public key for all members belonging to the group and a unique private key for every signer. Then, a member who belongs to a group can sign messages on behalf of the group in such a way that the resulting signature is publicly verifiable but the identity of the signer who actually has signed the message is hidden within the group. However, anonymity can be revoked by a special operation, but it can only be used by a trusted third party, usually called the group manager.

The interest on group signatures comes from the fact that we had researched means to provide revocable anonymity to some proposed schemes in this dissertation. Aside from the need to understand some of their internal features to know how they actually work, the fundamental purpose with the implementation of a group signature scheme was to utilize it as a development box ready to use by privacy-protecting proposals. However, once a successful implementation has been achieved, we have realized that it could be of particular interest to also compare performance results with another group signature scheme based on different mathematical assumptions. In fact, the first selected group signature scheme is based on pairing operations and Elliptic Curve Cryptography (ECC) [46] while the second one is based on RSA [44]. This way, it is also achieved an additional and initially not considered objective: provide a comparison between two different group signature schemes in order to analyze their performance on mobile devices. Therefore, since we cannot find any similar study and due to the fact that group signatures are usually addressed to be cryptographic black-boxes, this research line could be of interest for other researchers who are dealing with pairing-based schemes to prove that they are actually implementable and practical to be used on mobile devices.

1.2.6 Implementation and Performance Evaluation of Privacy-protecting Schemes

In addition to the implementation and performance analysis of two group signature schemes, the final purpose of this dissertation in reference to the described privacy-protecting schemes, is to provide a complete, practical and functional implementation together with an analysis of performance using real testing scenario with mobile devices. In fact, *μEasyPay*, $\mathscr{MC}-2\mathscr{D}$ and AFC solutions have been implemented with Java language. Both client and server side have been developed. Client side covers customer application to be installed on mobile devices running the Android OS while server side runs on more powerful computers. As a result of a careful performance evaluation, we prove that all solutions proposed in this dissertation are in fact practical and really efficient and scalable.

1.3 Common Security Properties Requirements

Despite the fact that contributions to privacy-protecting solutions for the e-commerce field have their concrete and special security requirements, all solutions proposed in this dissertation share a basic common security framework. It usually comes from the fact that proposed solutions deal somehow with elements that carry or exchange items with some monetary value. Therefore, involved parties in these schemes want to be protected from malicious behaviors. We then provide general and informal definitions based on specific definitions applied to particular previous solutions dealing with similar scenarios with common security requirements [11–13, 19–25, 27]. This way, we generalize unforgeability, reuse avoidance, anonymity, unlinkability and untraceability definitions to fit all proposed solutions.

First, consider an electronic document as a general e-commerce element that gives to the owner entity who holds it, typically a user or customer, certain rights to achieve an item or service from a merchant or provider. So, as stated before, this dissertation presents three different schemes for e-commerce that manage different types of electronic documents. This way, micropayment defines a coin of microcoupons as an electronic document owned by a customer to buy some items or services from a merchant. Similarly, multicoupon scheme considers a multicoupon of coupons to redeem them in exchange of some items or services from merchants. Finally, automatic fare collection system takes into account tickets to allow users to utilize a service in exchange of a fare.

The first security property to consider is the unforgeability. Since all solutions manage electronic documents with some implicit or explicit monetary value, honest parties want to be sure that electronic documents cannot be forged.

***Definition* 1.3.1** *(Unforgeability). An unforgeable privacy-protecting solution avoids*

an electronic document being accepted by verifiers if it has been modified in such a way that the owner has tried to achieve some dishonest benefit of this action.

There are schemes in which electronic documents can be reused up to a limited number of times (*reusable* schemes). It is usually a parameter previously agreed between the entity who issues the electronic document and receiver of this document. However, there are other solutions requiring this electronic document not be used more than once (*non-reusable* schemes). Along this dissertation, all proposed schemes consider electronic documents can only be used once in such a way that using them more than once is susceptible of punishment.

***Definition* 1.3.2** *(Reuse avoidance). A non-reusable privacy-protecting solution must avoid, or at least detect, dishonest parties trying to utilize electronic documents more than once.*

In general, electronic document owners are more likely to be anonymous in such a way that merchants and providers cannot recognize their real identities. However, there are two different types of anonymity considered along this dissertation: *non-revocable* and *revocable* anonymity.

***Definition* 1.3.3** *(Non-revocable anonymity). An anonymous privacy-protecting solution respects the secrecy of the identity of parties and guarantees that the owner who rightfully holds the electronic document and uses it cannot be identified.*

***Definition* 1.3.4** *(Revocable anonymity). An anonymous privacy-protecting solution with revocable anonymity follows Definition 1.3.3 on condition that owners holding and using the electronic document behaves correctly, otherwise their anonymity could be revoked by means of a trapdoor function.*

Besides, electronic document owners want their actions within an e-commerce solution not to be either linked or traced in such a way that nobody must be able to build profiles with all actions performed within the e-commerce solution.

***Definition* 1.3.5** *(Unlinkability). An unlinkable privacy-protecting solution provides the impossibility to attribute different actions performed in the system to the same party regardless it remains anonymous, even with a collusion of misbehaving parties.*

***Definition* 1.3.6** *(Untraceability). An untraceable privacy-protecting solution provides the indistinguishability of past and future actions in the system related to the identity of honest parties with the knowledge of the current transaction, even with a collusion of misbehaving parties.*

Then, at the time to formally define every scenario and security model for each proposal, these common security requirements will be further formalized and applied to the concrete scenario.

1.4 Outline of this Dissertation

The dissertation has been organized by means of five Parts covering a total number of 13 Chapters.

Part I covers this introductory Chapter together with Chapter 2 in which we review the most important cryptographic tools used along this work: hash chain procedure, partially blind signature and group signature.

Part II includes security models, related work and descriptions of three privacy-protecting schemes for e-commerce. In fact, Chapter 3 presents *μEasyPay*, an efficient micropayment scheme for low-value purchases; Chapter 4 describes $\mathscr{MC}-2\mathscr{D}$, a multicoupon scheme to achieve discounts or gifts; a joint work with other researchers appears in Chapter 5 to improve a previous AFC proposal.

Part III contains information related to security analysis of properties. In Chapter 6 we describe the proposed CPN methodology to automatically analyze some security requirements of secure protocols and in Chapter 7 we present a careful and formal security analysis of all privacy-protecting schemes proposed in Part II.

Part IV covers all implementations and performance analysis of above schemes. Chapter 8 describes the way how we have implemented and conducted performance analysis of our privacy-protecting schemes. Then, in Chapters 9, 10, 11 and 12 we describe implementations and performance analysis of group signatures, *μEasyPay*, $\mathscr{MC}-2\mathscr{D}$ and AFC e-commerce schemes, resp.

Finally, Part V include the Chapter 13 in which we conclude the dissertation.

1.5 Contributions and Funding

This dissertation is backed by a list of contributions to international conferences, international journals and book, as well as it is also worthy of mention the active participation in national conferences such as JITEL (*Jornadas de Ingeniería Telemática*), RESCI (*Reunión Española sobre Criptología y Seguridad de la Información*) and RSME conference (*Congreso de la Real Sociedad Matemática Española*). The complete list can be found in Chapter 14.

This work has been partially supported by MEC (Ministerio de Educación, Cultura y Deporte) and FEDER (Fondo Europeo de Desarrollo Regional) under project "Advanced Research on Information Security and Privacy" ARES-CONSOLIDER INGENIO [CSD2007-004].

CHAPTER 2

BACKGROUND ON CRYPTOGRAPHIC TOOLS

The privacy-protecting schemes for e-commerce to be presented along this dissertation rely on some key cryptographic tools. They provide some of the security requirements and give us some important functionalities. So, in this Chapter we provide a summary on these cryptographic algorithms, i.e. the hash chain procedure (used in Chapters 3 and 4), the blind and partially blind signature (used in Chapters 3 and 4) and finally, the group signature (used in Chapters 4 and 5).

2.1 Hash Chains

A hash chain $(w_N \cdots w_0)$ is defined as a set of values where w_N is a random value used as the seed of the hash chain, and the rest of values are generated applying iteratively a cryptographic hash function $\mathcal{H}$ over w_N up to the value w_0, called root, i.e. $\mathcal{H}(w_N) = w_{N-1}, \mathcal{H}(w_{N-1}) = w_{N-2}, ..., \mathcal{H}(w_1) = w_0$. Then, the verification of a chained element w_i $(N \geq i \geq 1)$ is performed applying i times the hash function $\mathcal{H}$ over it, checking whether the relation $\mathcal{H}^i(w_i) \stackrel{?}{\equiv} w_0$ holds. The verifier should know the root value w_0.

The key feature of hash chains is that providing a value w_i, it is not feasible to find another w_j where $j > i$ so that $\mathcal{H}^{j-i}(w_j) = w_i$.

2.2 Blind and Partially Blind Signature

2.2.1 Overview of Blind Signature

The concept of blind signatures were introduced by Chaum [49] as a key tool for building various electronic cash instruments to protect users to be traced back by banks when they spend money. Thus, this tool avoids banks know where a user spends her electronic currency. The security model of blind signatures was formalized by [50, 51] together with seminal blind signature proposals. Informally, blind signatures allow a user ($\mathcal{U}$) to obtain signatures (σ) from a signer ($\mathcal{S}$) on any message (m) in such a way the signer learns nothing about the message that is being signed.

Following the model of blind signatures presented in [51], such scheme can be defined by using two Interactive Turing machines ($\mathcal{S}$ and $\mathcal{U}$) running an interactive protocol ($Sign$) and two algorithms ($KeyGen$ and $Verif$):

- $KeyGen(1^k)$. It is a probabilistic polynomial-time algorithm that takes as input a security parameter (1^k) and outputs a pair of public and secret keys (pk, sk).
- $Sign_{sk,m}(pk)$. $\mathcal{U}$ and $\mathcal{S}$ engage in this interactive protocol of some polynomial (in 1^k) number of rounds. Both have a common input pk (the public key generated during $KeyGen$). $\mathcal{S}$ takes as private input sk (the secret key) and $\mathcal{U}$ has private input m, being the message to be signed. At the end of this protocol, $\mathcal{S}$ outputs either *completed* or *not-completed* and $\mathcal{U}$ outputs similarly either *fail* or σ.
- $Verif(pk, m, \sigma)$. It is a deterministic polynomial-time algorithm, which outputs either *accept* or *reject* after verify whether the σ on the message m is valid.

Security of blind signatures is formalized in [51], using concepts formulated in the original paper of Chaum [49] and from the paper of Poincheval and Stern [50]. Informally, a blind signature scheme is secure if it satisfies the *blindness* and the *unforgeability* properties.

BLINDNESS. This property follows from [49] and it means that a signer cannot distinguish, except with negligible probability, the order in which she issued signatures.

UNFORGEABILITY. This property follows from [50] (where it is called *one-more forgery*) and it means that after getting l signatures, it is unfeasible for $\mathcal{U}$ to compute $l+1$ signatures.

2.2.2 Overview of Partially Blind Signatures

One shortcoming of blind signatures is that, since the signer's view of the message to be signed is completely blocked, the signer has no control over the attributes except for those bound by the public key. A simple example of this shortcoming can be seen in a simple electronic cash system where a bank issues a blind signature as an electronic coin. Since the bank cannot set the value on any blindly issued coin, it has to use different public keys for different coin values.

In order to address such issues, the notion of partially blind signatures ($\mathcal{PBS}$ for short) was introduced in [52], and the formal security definition and a secure partially blind signature scheme in the random oracle model where presented in [53]. A partially blind signature allows the signer to explicitly include some information in the blind signature under some previous agreement with the user. This concept is a generalization of blind signatures since the blind signatures are a special case of partially blind signatures where the common information is a null string [54]. Following the above example, in case that simple electronic cash system uses a partially blind signature, it can use only one public key to sign issued coins. Moreover, partially blind signatures allow to include more information to the resulting signature, such as an expiration date.

Following the security definitions of partially blind signatures described by authors in [53], a partially blind signature can be defined as a tuple $\mathcal{PBS} = \{KeyGen^{\mathcal{PBS}}, Sign^{\mathcal{PBS}}, Verif^{\mathcal{PBS}}\}$ where $Sign^{\mathcal{PBS}}$ is being run by two interactive Turing machines (the signer $\mathcal{S}$ and the user $\mathcal{U}$):

- $KeyGen^{\mathcal{PBS}}\left(1^k\right)$. This probabilistic polynomial-time algorithm takes the security parameter $\left(1^k\right)$ and outputs a public and secret key pair $\left(pk, sk\right)$.
- $Sign^{\mathcal{PBS}}_{sk,m}\left(pk, info\right)$. It is a probabilistic and interactive protocol between $\mathcal{S}$ and $\mathcal{U}$ where $\mathcal{S}$ issues a partially blind signature on a message m provided by $\mathcal{U}$ in such a way $\mathcal{S}$ cannot know what are the contents of m. They have a common input pk (the public key generated during the $KeyGen^{\mathcal{PBS}}$) and a pre-agreed common information ($info$) between $\mathcal{S}$ and $\mathcal{U}$. $\mathcal{S}$ has private input sk (the secret signing key). $\mathcal{U}$ has private input m, which is the message to be partially blind signed. Some authors and proposals [55] divide this algorithm in three different procedures:
  - $Request^{\mathcal{PBS}}$. During this process, $\mathcal{U}$ hides m and outputs a new string s which has to be signed by $\mathcal{S}$ together with $info$.
  - $DoSign^{\mathcal{PBS}}$. On a string s and the signer's secret signing key sk, it outputs a signature $\hat{\sigma}$ that has to be unblinded by $\mathcal{U}$.
  - $Extract^{\mathcal{PBS}}$. On a signature $\hat{\sigma}$, she unblinds it and obtains the final partially blind signature σ.

- $Verif^{\mathcal{PBS}}(m, info, pk, \sigma)$. This deterministic polynomial-time algorithm takes $(pk, info, m, \sigma)$ and outputs either *accept* or *reject*.

Security of partially blind signature schemes is described in terms of three main requirements [53, 56]: *completeness*, *partial blindness* and *unforgeability*.

COMPLETENESS. If $\mathcal{S}$ and $\mathcal{U}$ honestly follow the signature issuing protocol $(Sign^{\mathcal{PBS}})$ with common info input $(pk, info)$, then, with probability of at least $1 - \frac{1}{k^c}$ for sufficiently large k and some constant c, $\mathcal{S}$ outputs $completed$, and $\mathcal{U}$ outputs (m, σ) that satisfies $Verif^{\mathcal{PBS}}(m, info, pk, \sigma) = accept$. The message-signature tuple $(info, m, \sigma)$ is valid with regard to pk if it leads $Verif^{\mathcal{PBS}}$ to *accept*.

PARTIAL BLINDNESS. Informally speaking, partial blindness requirement ensures that the scheme must satisfy the following two properties [53, 56]:

- The signer assures that an issued signature contains the information that it desires, and none can remove the embedded information from the signature.
- For the same embedded information, the signer cannot link a signature to the instance of the signing protocol that produces the corresponding partially blind signature.

We address the reader to [53] for a full formal definition of partial blindness.

UNFORGEABILITY. Intuitively, a partially blind signature is called unforgeable if for every $info$, and some integer l, there is no probabilistic polynomial-time adversary $\mathcal{A}$ that can compute, after l interactions with $\mathcal{S}$ using $Sign^{\mathcal{PBS}}_{sk,m}(pk, info)$, $l+1$ signatures with non-negligible probability. As well as in partial blindness requirement, the complete formal definition of unforgeability can be shown in [53].

2.2.3 Review the RSA-based Partially Blind Signature due to Chien et al.

There are many partially blind signature proposals, each of them based on different cryptographic assumptions and mathematical problems. For example, there are schemes based on pairings [54, 56] and other ones based on RSA [55]. We have selected a simple but powerful and secure partially blind signature based on the RSA factorization problem, proposed by Chien et al. [55] due to the fact that it seems one of the most simple schemes to implement in order to create a software library to be used by the implementation of the schemes that will be presented in this dissertation. Following the above formal definition of each algorithm, the RSA-based partially blind signature is defined as following:

- $KeyGen^{\mathscr{PBS}}$. $\mathscr{S}$ randomly chooses two large primes (p,q), and computes $n = p \cdot q$ and $\phi(n) = (p-1)(q-1)$. Next, $\mathscr{S}$ calculates d, such that $e \cdot d = 1 mod \phi(n)$, where $e = 3$. $\mathscr{S}$ outputs the key pair $(pk, sk) = [(e,n),(d,p,q)]$, where pk is the public key and sk is stored in a secret way by $\mathscr{S}$. Finally, a secure one-way cryptographic hash function, such as SHA-1, is published.

- $Request^{\mathscr{PBS}}$. $\mathscr{U}$ selects the message m to be partially signed and the corresponding pre-agreed common information $info$. $\mathscr{U}$ chooses two random numbers $r, u \in \mathbb{Z}_n^*$ and computes $\alpha = r^e \mathscr{H}(m)(u^2+1) \mod n$ and sends the tuple $(\alpha, info)$ to $\mathscr{S}$. After verifying $info$, $\mathscr{S}$ picks a random positive integer $x \in \mathbb{Z}_n^*$ and $x < n$ and send it to $\mathscr{U}$. Then, $\mathscr{U}$ randomly selects another integer $r' \in \mathbb{Z}_n^*$ and computes $b = r \cdot r'$. Finally, $\mathscr{U}$ computes $\beta = b^e(u-x) \mod n$ and sends β to $\mathscr{S}$.

- $DoSign^{\mathscr{PBS}}$. $\mathscr{S}$ computes $\beta^{-1} \mod n$ and $t = \mathscr{H}(a)^d \left(\alpha\left(x^2+1\right)\beta^{-2}\right)^{2d} \mod n$. Then, $\mathscr{S}$ returns $\hat{\sigma} = \left(\beta^{-1}, t\right)$ to $\mathscr{U}$.

- $Extract^{\mathscr{PBS}}$. Upon receiving $\hat{\sigma}$, $\mathscr{U}$ obtains the final partially blind signature $\sigma = (info, c, s)$ unblinding the received pair (β^{-1}, t) and computing $c = (ux+1) \cdot \beta^{-1} \cdot b^e = \frac{ux+1}{u-x} \mod n$ and $s = tr^2 r'^4 \mod n$.

- $Verif^{\mathscr{PBS}}$. Anyone can verify the partially blind signature σ using $(m, info, \sigma, pk)$ by checking whether $s^e \equiv \mathscr{H}(info)\mathscr{H}(m)^2\left(c^2+1\right)^2 \mod n$.

2.3 Group Signatures

Finding authentication methods has been a central problem in cryptography. A great number of authentication protocols and cryptographic tools have been created to establish authenticated channels. The group signature schemes are a useful anonymous non-repudiable multiuse credential primitive that was introduced in 1991 by Chaum and Van Heyst [41] that can be used to provide authenticity and anonymity to signers at the same time. So, a group signature is a basic privacy mechanism. Informally, the group signature cryptographic tool involves a group of signers, each holding a membership certificate. Any member of this group can sign messages that are publicly verifiable while hiding the identity of the actual signer within the group. Thus, the resulting signature keeps the signer's identity secret. However, a third party, in the case of any dispute or abuse, can trace the signature, undoing its anonymity, using a special trapdoor. So, only that third party can open an individual signature revealing the identity of its originator.

2.3.1 Overview of Group Signatures

Authors in [42] proposed a formal security model of group signature schemes defining the involved entities. Thus, three types of parties are considered: the group manager, the signer and the verifier. First, the group manager is the entity in charge of managing and deploying the parameters of the group signature scheme, generating keys and adding new members to the group. It can be also responsible of opening signatures and revealing the identity of the signer who had computed a group signature. However, in the literature on group signatures [43], it is common to assume that the latter task is assumed by an opening manager different from the group manager. Secondly, the signer is a user who acquires a group key pair (therefore, she is a member of a group of users) and she uses the scheme to group sign a message of arbitrary length. Finally, the verifier is another user who belongs to (or does not belong to) the same group as the signer and he is in charge of verify whether the signature is valid and made by a user belonging to the claimed group.

According to [42], a group signature scheme $\mathcal{GS} = \{KeyGen^G, Sign^G, Verify^G, Open^G\}$ is defined by the following four polynomial-time algorithms:

- $KeyGen^G(1^n, 1^k)$. This randomized algorithm, performed by the group manager, takes as inputs both the number of members of the group (n) and a security parameter (k), where ($k, n \in \mathbb{N}$). It outputs a group public key (pk^G), a private key of the group manager ($sk^G_{\mathcal{G}}$) and n user private keys $\left(sk^G[n] = sk^G_1, \ldots, sk^G_u, \ldots, sk^G_n\right)$.

- $Sign^G_u(pk^G, sk^G_u, m)$. This randomized algorithm is used by the signer u. Given a group public key pk^G, a user private key sk^G_u and a message m of an arbitrary length, this procedure outputs a group signature σ ($u \in [n]$).

- $Verify^G(pk^G, m, \sigma)$. This deterministic algorithm is executed by a verifier (who can either belong the same group of the signer or not). Given a group public key pk^G, a message m ($m \in \{0,1\}^*$) and a group signature (σ), it verifies whether σ is a valid group signature on m (i.e. return either 1 or 0).

- $Open^G_{\mathcal{G}}(pk^G, sk^G_{\mathcal{G}}, m, \sigma)$. This deterministic algorithm is used by the group manager to trace a signature to the identity of the signer. It takes a group public key $\left(pk^G\right)$, the group manager's private key $\left(sk^G_{\mathcal{G}}\right)$, a message ($m$) and a signature ($\sigma$), and it recovers and outputs either the identity of the signer (u, being $u \in [n]$) who had computed originally σ or the symbol $\perp$ to indicate failure.

Finally, [42] proposes four key security properties that a group signature scheme must satisfy for the setting in which the group is static (i.e. a group where the number

and identities of the members are decided at the time the group is set up and new members cannot be added later): *correctness*, *compactness*, *full-anonymity* and *full-traceability*. We provide a summary of these properties in an informal way. The complete formal definition and analysis can be found in [42].

CORRECTNESS. It ensures that honestly-generated signatures verify and trace correctly. So, a scheme should satisfy that for all $k, n \in \mathbb{N}$, all $(pk^G, sk^G_{\mathcal{G}}, sk^G[])$, all $u \in [n]$ and all $m \in \{0,1\}^*$ we have $Verify^G(pk^G, m, \sigma) = 1$ and $Open^G_{\mathcal{G}}(pk^G, sk^G_{\mathcal{G}}, m, \sigma) = u$. The former says that true signatures are always *valid* and the latter asks that the opening algorithm correctly recovers the identity of the signer who had computed a true signature.

COMPACTNESS. It is preferable that sizes of keys and signatures in a group signature scheme do not grow proportionally to the number of members (n). The upper bound length of the elements generated by a group signature must be $log(n)$.

FULL-ANONYMITY. This property ensures that no signature reveal the identity of the signer. Informally, anonymity requires that an adversary not in possession of the group manager's secret key and even colluding with other group members, find it hard to recover the identity of the signer from its signature.

FULL-TRACEABILITY. In case of misbehavior, the signer anonymity can be revoked by the group manager. Informally speaking, it is required that no colluding set of group members (even consisting of the entire group, and even being in possession of the secret key for opening signatures) can create signatures that cannot be opened, or signatures that cannot be traced back to some members of the forging coalition.

Authors in [43], extend the work of [42] in a sense that improves the security model and considers dynamic groups of signers rather than static ones. Among other things, authors add a $Join^G$ algorithm to manage how signers obtain their group credentials in dynamic groups:

- $Join^G$. It is an interactive and probabilistic algorithm used by both the group manager and the user who wants to start belonging to a group of signers. The future signer provides group manager with her identity together with her personal credentials (such as her digital certificate). If the algorithm ends properly, the user u achieves its private signing key $\left(sk^G_u\right)$ together with the corresponding group public key $\left(pk^G\right)$ (also called the membership certificate). It is supposed that the communication takes place over a secure (i.e. private and authenticated) channels. From now, the signer u belongs the group and

the group manager has the link between the last issued signing key and the corresponding user's identity.

Generally, in schemes considering dynamic groups, it is during the $Join^G$ algorithm, rather than during the $KeyGen$ algorithm, when the secret signing key of the user $\left(sk_u^G\right)$ is computed and returned to her.

Despite the formal definition and the number of group signature proposals, the problem on formulating efficient group signature schemes has been a research target. Ateniese et al. [44] presented the ACJT group signature based on the Strong-RSA assumption. However, this scheme (and other ones based on the Strong-RSA assumption) generates very long signatures, so new proposals had been proposed in the literature to reduce them [45–48]. Boneh et al. [46] presented the first practical pairing-based group signature scheme (Boneh-Boyen-Shachan (BBS) scheme) using the Strong Diffie-Hellman (SDH) assumption [57]. Using carefully selected security parameters, the specified scheme has approximately the same level of security as a regular RSA signature of the same length.

2.3.2 Applications

Group signatures have been showed to be extremely useful in various applications such as anonymous credentials, e-cash, e-vote, e-ticketing, e-auction and identity management. Some of these applications make use of mobile devices.

Some examples of group signatures application fall in the field of anonymous remote attestation. This enables remote attestation of virtual machine instances while preserving privacy under the user's control [58–60]. In these applications, during attestation process to a remote party (e.g. a bank) the group signatures are sent to this party. The signatures prove that the attestation was issued by a valid machine or user, but hides which machine it comes from.

Fujii et al. [61] proposed a protocol for subscription services to achieve anonymity by using a group signature. The subscriber (member of a group) has to register at a service manager, and then he can request a service at the distributor. Some other papers like [62,63] present public auction schemes based on group signature systems where the privacy of bidders is protected.

Other application examples enhance users' privacy even when the user is not a signer. In this case, the issuer of certificates uses a group signature to sign certificates. For example, car drivers can hide in which country they obtained the drivers license if this document has a group signature.

Regarding to e-cash, some proposals [64–66] make use of group signatures to provide anonymity to the customers but allowing the revocation of their anonymity if they misbehave.

2.3.3 Review Boneh et al. and Ateniese et al. Group Signature Schemes

Let us review in this Section the specification of both selected group signature schemes: the BBS pairing-based group signature and the ACJT state-of-the-art non-pairing group signature. As said before, we have chosen the ACJT scheme to compare with the BBS scheme because the former is based on different and more classical mathematical assumptions rather than the latter, which is based on bilinear maps, although it is known that schemes based on the Strong-RSA assumption produces longer signatures than pairing-based schemes. Note that along the descriptions of both schemes, we highlight, using grey letters, some operations that can be precomputed before to perform a group signature and its corresponding verification.

BBS Group Signature Scheme Based on Bilinear Maps

As pointed out before, the BBS scheme is based on bilinear maps, so let us make a brief review of few concepts about them. We follow the notation used in [46]:

- $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_T$ are three multiplicative cyclic groups, of prime order p,
- g_1 is a generator of $\mathcal{G}_1$ and g_2 is a generator of $\mathcal{G}_2$,
- ψ is a computable isomorphism $\psi : \mathcal{G}_2 \rightarrow \mathcal{G}_1$, with $\psi\left(g_2\right) = g_1$, and
- e is a computable map $e : \mathcal{G}_1 \times \mathcal{G}_2 \rightarrow \mathcal{G}_T$ with the following properties:
  - bilinearity: $\forall (u, v) \in \mathcal{G}_1 \times \mathcal{G}_2$ and $a, b \in \mathbb{Z}_p$, $e\left(u^a, v^b\right) = e\left(u, v\right)^{ab}$
  - non-degeneracy: $e\left(g_1, g_2\right) \neq 1$

The security of the signature scheme relies on the SDH and the Linear assumptions in groups with a bilinear map [57].

Following the notation of §2.3.1, the BBS scheme is defined by $KeyGen^G$, $Sign^G$, $Verify^G$ and $Open^G$ algorithms.

$KeyGen^G$. The algorithm, executed by the group manager, proceeds as follows:

1. Picks $h \xleftarrow{R} \mathcal{G}_1 \setminus \{1_{\mathcal{G}_1}\}$ ($\xleftarrow{R}$ means a random selection)
2. Picks $\xi_1, \xi_2 \xleftarrow{R} \mathbb{Z}_p^*$. The private key of the group manager is $gmsk = (\xi_1, \xi_2)$

3. Set $u, v \in \mathcal{G}_1$ such that $u^{\xi_1} = v^{\xi_2} = h$
4. Select $\gamma \xleftarrow{R} \mathbb{Z}_p^*$ and set $\omega = g_2^{\gamma}$
5. The group public key is $gpk = \left(g_1, g_2, h, u, v, \omega\right)$, where $g_1, h, u, v \in \mathcal{G}_1$ and $g_2, \omega \in \mathcal{G}_2$
6. For each user i, $1 \leq i \leq n$, the group manager generates a SDH ([46]) tuple (A_i, x_i):

$$x_i \xleftarrow{R} \mathbb{Z}_p^* \quad A_i \leftarrow g_1^{\frac{1}{(\gamma + x_i)}}$$

7. The private key of each user i is $gsk[i] = (A_i, x_i)$, where $A_i \in \mathcal{G}_1$ and $x_i \in \mathbb{Z}_p^*$

When a user asks for a key pair to the group manager (join the group), she must be identified and authenticated with her own credentials, such as her digital certificate. Then, the group manager selects and sends to the user an unassigned private key $(sk = (A_i, x_i))$ within the set $gsk[n]$, together with the corresponding group public key, and marks the assigned signing key as used. This way, a user becomes a member of the group and each group member has her own private key. Thus, the group manager could identify each group member uniquely.

Sign^G. The signer can precompute some parameters before to sign the message M $(M \in \{0,1\}^*)$. The procedure for a user i with private key (A_i, x_i) follows:

1. Picks $\alpha, \beta \xleftarrow{R} \mathbb{Z}_p$
2. Picks $r_\alpha, r_\beta, r_{x_i}, r_{\delta_1}, r_{\delta_2} \xleftarrow{R} \mathbb{Z}_p$
3. Calculate $T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5$

$$T_1 \leftarrow u^{\alpha} \qquad T_2 \leftarrow v^{\beta} \qquad T_3 \leftarrow A_i h^{\alpha+\beta}$$

$$\delta_1 \leftarrow x_i \alpha \qquad \delta_2 \leftarrow x_i \beta$$

$$R_1 \leftarrow u^{r_\alpha} \qquad R_2 \leftarrow v^{r_\beta}$$

$$R_3 \leftarrow e(T_3, g_2)^{r_{x_i}} \cdot e(h, \omega)^{-r_\alpha - r_\beta} \cdot e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}}$$

$$R_4 \leftarrow T_1^{r_{x_i}} \cdot u^{-r_{\delta_1}} \qquad R_5 \leftarrow T_2^{r_{x_i}} \cdot v^{-r_{\delta_2}}$$

4. Using a hash function $\mathscr{H}: \{0,1\}^* \rightarrow \mathbb{Z}_p$, the user i obtains the challenge c

$$c = \mathscr{H}(M, T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5) \in \mathbb{Z}_p$$

5. Using the challenge c the user i obtains $s_\alpha, s_\beta, s_{x_i}, s_{\delta_1}$, and s_{δ_2}:

$$s_\alpha = r_\alpha + c\alpha \quad s_\beta = r_\beta + c\beta \quad s_{x_i} = r_{x_i} + cx_i$$
$$s_{\delta_1} = r_{\delta_1} + c\delta_1 \quad s_{\delta_2} = r_{\delta_2} + c\delta_2$$

6. The signature of knowledge is $\sigma = \left(T_1, T_2, T_3, c, s_\alpha, s_\beta, s_{x_i}, s_{\delta_1}, s_{\delta_2}\right)$, where $T_1, T_2, T_3 \in \mathscr{G}_1$ and $c, s_\alpha, s_\beta, s_{x_i}, s_{\delta_1}, s_{\delta_2} \in \mathbb{Z}_p$

Verify^G. This algorithm also allows the precomputation of some operations, as the *Sign*^{G} algorithm does. The verification algorithm follows these steps:

1. Re-derive R_1, R_2, R_3, R_4, R_5

$$\widetilde{R}_3 \leftarrow e\left(T_3, g_2\right)^{s_{x_i}} \cdot e(h,\omega)^{-s_\alpha - s_\beta} \cdot e\left(h, g_2\right)^{-s_{\delta_1} - s_{\delta_2}} \cdot \left(\frac{e(T_3,\omega)}{e(g_1,g_2)}\right)^c$$

$$\widetilde{R}_1 \leftarrow \frac{u^{s_\alpha}}{T_1^c} \qquad \widetilde{R}_2 \leftarrow \frac{v^{s_\beta}}{T_2^c} \qquad \widetilde{R}_4 \leftarrow \frac{T_1^{s_{x_i}}}{u^{s_{\delta_1}}} \qquad \widetilde{R}_5 \leftarrow \frac{T_2^{s_{x_i}}}{v^{s_{\delta_2}}}$$

2. Check

$$c \stackrel{?}{\equiv} \mathscr{H}\left(M, T_1, T_2, T_3, \widetilde{R}_1, \widetilde{R}_2, \widetilde{R}_3, \widetilde{R}_4, \widetilde{R}_5\right)$$

   If equality holds, σ is accepted, otherwise is rejected.

Open^G. The group manager can reveal the identity of the signer who had computed σ as follows:

1. Verify that σ is a valid signature on M using *Verify*^{G}.

2. Considering T_1, T_2, T_3, the group manager can recover A_i using his private key $gmsk = (\xi_1, \xi_2)$:

$$A_i \leftarrow \frac{T_3}{\left(T_1^{\xi_1} \cdot T_2^{\xi_2}\right)}$$

because $T_1^{\xi_1} = h^{\alpha}$ and $T_2^{\xi_2} = h^{\beta}$ (see the above $Sign^G$ procedure)

3. The group manager can recover the user identity linked to the element A_i.

ACJT Group Signature Scheme based on Strong-RSA

Instead of relying its security on bilinear maps, the ACJT group signature scheme [44] is proven secure under the Strong-RSA and the Decisional Diffie-Hellman (DDH) assumptions in cyclic groups of prime order. It is claimed that proposals based on these kind of assumptions compute signatures longer and they are less efficient than schemes based on bilinear maps, such as the previously reviewed BBS scheme.

The ACJT scheme considers some security parameters and several lengths satisfying some conditions. Let $\epsilon > 1$ (it controls the tightness of the statistical zero-knowledgeness), k (it is the output length of a collision-resistant hash function) and l_p (it sets the size of the modulus to use) three security parameters. Moreover, $\lambda_1, \lambda_2, \gamma_1$ and γ_2 denote lengths satisfying $\lambda_1 > \epsilon(\lambda_2 + k) + 2$, $\lambda_2 > 4l_p$, $\gamma_1 > \epsilon(\gamma_2 + k) + 2$ and $\gamma_2 > \lambda_1 + 2$. It also defines two ranges as $\Lambda = \left]2^{\lambda_1} - 2^{\lambda_2}, 2^{\lambda_1} + 2^{\lambda_2}\right[$ and $\Gamma = \left]2^{\gamma_1} - 2^{\gamma_2}, 2^{\gamma_1} + 2^{\gamma_2}\right[$. Finally, it defines $\mathscr{H}$ as a collision-resistant hash function $\mathscr{H}: \{0,1\}^* \rightarrow \{0,1\}^k$.

$KeyGen^G$. The algorithm proceeds as follows:

1. Select random secret l_p primes p', q' such that $p = 2p' + 1$ and $q = 2q' + 1$ are prime (safe prime), and set the modulus $n = pq$.

2. Choose random $a, a_0, g, h \xleftarrow{R} QR(n)$ (of order $p'q'$).

3. Choose a random secret element $x \xleftarrow{R} \mathbb{Z}_{p'q'}$ and set $y = g^x \mod n$.

4. The group public key is $gpk = (n, a, a_0, y, g, h)$.

5. The corresponding group manager secret key is $gmsk = \left(p', q', x\right)$

$Join^G$. Anyway, using this algorithm, the user i obtains from the group manager the new membership certificate $[A_i, e_i]$, where $A_i = (C_2 a_0)^{\frac{1}{e_i}} \mod n = \left(a^{2^{\lambda_1} + (\alpha_i \hat{x}_i + \beta_i \mod 2^{\lambda_2})} a_0\right)^{\frac{1}{e_i}} \mod n$ and $e_i \xleftarrow{R} \Gamma$.

1. User i generates a secret exponent $\hat{x_i} \xleftarrow{R}]0, 2^{\lambda_2}[$, a random integer $\hat{r} \xleftarrow{R}]0, n^2[$ and sends $C_1 = g^{\hat{x_i}} h^{\hat{r}} \mod n$ to the group manager. She proves his knowledge of the representation of C_1 w.r.t. bases g and h.

2. The group manager checks that $C_1 \in QR(n)$. If this is the case, the group manager selects $\alpha_i, \beta_i \xleftarrow{R}]0, 2^{\lambda_2}[$ and sends (α_i, β_i) to the user i.

3. The user i computes $x_i = 2^{\lambda_1} + (\alpha_i \hat{x_i} + \beta_i \mod 2^{\lambda_2})$ and sends to the group manager $C_2 = a^{x_i} \mod n$. The user also proves to the group manager:

   a) that the discrete log of C_2 w.r.t. base a lies in Λ, and

   b) that the user's membership secret $x_i = log_a C_2$ is correctly computed from C_1, α_i and β_i.

4. The group manager checks that $C_2 \in QR(n)$. If this is the case and all the above proofs were correct, the group manager selects a random prime $e_i \xleftarrow{R} \Gamma$ and computes $A_i := (C_2 a_0)^{\frac{1}{e_i}} \mod n$. Finally, the group manager sends to the user i the new membership certificate $[A_i, e_i]$.

5. The user i verifies that $a^{x_i} a_0 \equiv A_i^{e_i} \pmod n$

$Sign^G$. Using the membership certificate (A_i, e_i), the member i can generate a group signature on a message M (note that there are some precomputable operations):

1. Generate a random value $w \xleftarrow{R} \{0,1\}^{2l_p}$ and compute:

$$T_1 = A_i y^w \mod n \qquad T_2 = g^w \mod n$$

$$T_3 = g^{e_i} h^w \mod n$$

2. Randomly choose:

$$r_1 \xleftarrow{R} \pm\{0,1\}^{\epsilon(\gamma_2+k)} \qquad r_2 \xleftarrow{R} \pm\{0,1\}^{\epsilon(\lambda_2+k)}$$

$$r_3 \xleftarrow{R} \pm\{0,1\}^{\epsilon(\gamma_2+2l_p+k+1)} \qquad r_4 \xleftarrow{R} \pm\{0,1\}^{\epsilon(2l_p+k)}$$

3. Compute d_1, d_2, d_3, d_4, c, s_1, s_2, s_3 and s_4:

$$d_1 = \frac{T_1^{r_1}}{(a^{r_2} y^{r_3})} \mod n \qquad d_2 = \frac{T_2^{r_1}}{g^{r_3}} \mod n$$

$$d_3 = g^{r_4} \mod n \qquad d_4 = g^{r_1} h^{r_4} \mod n$$

$$c = \mathcal{H}\left(g, h, y, a_0, a, T_1, T_2, T_3, d_1, d_2, d_3, d_4, M\right)$$

$$s_1 = r_1 - c\left(e_i - 2^{\gamma_1}\right) \qquad s_2 = r_2 - c\left(x_i - 2^{\lambda_1}\right)$$
$$s_3 = r_3 - c\left(r_3 - c e_i w\right) \qquad s_4 = r_4 - c w \text{ (all in } \mathbb{Z})$$

4. Output $\sigma = (c, s_1, s_2, s_3, s_4, T_1, T_2, T_3)$.

Verify^G. A verifier can check whether the signature σ on the message M is valid as follows (note that some exponentiations can be precomputed):

1. Compute:

$$\begin{aligned} c' = \mathcal{H}(g, h, y, a_0, a, T_1, T_2, T_3, \\ a_0^c T_1^{s_1 - c2^{\gamma_1}} / \left(a^{s_2 - c2^{\lambda_1}} y^{s_3}\right) \mod n \\ T_2^{s_1 - c2^{\gamma_1}} / g^{s_3} \mod n, \\ T_3^c g^{s_1 - c2^{\gamma_1}} h^{s_4} \mod n, \\ M) \end{aligned}$$

2. Accept the signature if and only if $c = c'$, $s_1 \in \pm\{0,1\}^{\epsilon(\gamma_2 + k)+1}$, $s_2 \in \pm\{0,1\}^{\epsilon(\lambda_2 + k)+1}$, $s_3 \in \pm\{0,1\}^{\epsilon(\gamma_1 + 2l_p + k + 1)+1}$ and $s_4 \in \pm\{0,1\}^{\epsilon(2l_p + k)+1}$.

Open^G. The group manager can identify the member who had computed the signature σ following these steps:

1. Check the signature's validity applying the *Verify*^{G} algorithm.
2. Recover A_i as $A_i = \frac{T_1}{T_2^x} \mod n$.
3. Prove that $log_g y = log_{T_2}\left(\frac{T_1}{A_i} \mod n\right)$.

Part II

Contribution to Privacy-protecting Solutions for e-Payment and e-Ticketing

CHAPTER 3

μEasyPay: MICROPAYMENT SCHEME FOR LOW VALUE PURCHASES

Electronic commerce (e-commerce) has emerged the last years as to become an important market involving large amounts of money. Because of this, new security requirements and privacy challenges are also appearing. Nowadays, customers can purchase items of any value merchants are willing to sell. However, an interesting emerging trend is the purchase of intangible goods or services involving low-value transactions. This kind of payments are called micropayments. They have unique functional and security requirements because the low-value of the involved items. So, a micropayment scheme must set a trade-off between security measures and the provided efficiency due to the fact that they are usually opposed. In this Chapter we propose an efficient and secure micropayment scheme that protects the privacy of customers and also presents means to detect and avoid malicious behaviors. Moreover, we prove that our micropayment scheme can be successfully applied to Location-Based Services (LBSs) subject to payment due to its privacy properties. In addition, we will demonstrate afterward that the scheme is suitable to be used by mobile devices even if they have limited resources, thanks to the fact that the micropayment proposal is so lightweight.

3.1 Introduction

Electronic commerce (e-commerce) introduce new requirements and challenges to on-line applications. Payment is one of the main stages of a commercial transaction and both merchants and customers require new methods to sell and pay in a secure and efficient way. Some purchases (intangible selling of goods such as information: newspapers, product reviews, location-based services, etc.; virtual gifts; or electronic data: music, videos, etc.) involve low-value transactions, so the operational cost needs to be as low as possible in order to be profitable for merchants and customers. These kind of payment schemes that emphasize the ability to make payments of small amounts are called micropayments [11–15]. They have unique functional and security requirements compared to other payment systems designed to manage larger amounts of money that tend to be costly. On one hand, security properties are a primary concern for the development of micropayment systems to control or to avoid financial risks for merchants and to ensure the required privacy for customers. On the other hand, efficiency and the cost of individual transactions are critical factors for the development of these systems. However, efficiency and security are usually opposed requirements, so micropayment protocols must provide a trade-off between these requirements.

Several micropayment schemes have been suggested in the literature. In general, they rely in a general scenario composed by customers, merchants and a bank, in which customers withdraw coins from the bank to purchase items from merchants. Then, merchants are allowed to ask for a money deposit in exchange for the received coins from customers. However, we can differentiate more micropayment scenarios in such a way that they can be classified depending on the following: the given use to the coins, how the coin validation is and the provided control to the funds. Then, a micropayment can use either generic coins to pay different merchants [11] or specific coins to be used with a single merchant (or even a single product) [13]. Moreover, depending on how the system validates the coins, we can distinguish the on-line scenario, in which the bank is involved in every payment transaction (i.e. the coin is validated by the bank at the time of payment) and the off-line scenario, where the bank does not take part during the payment (i.e. the coin will be verified by the bank after the payment). Finally, regarding to the control of funds, we have found credit-based and debit-based scenarios depending on when the bank charges customers. In a credit-based scenario [13], the customer is charged when the merchant deposits the spent coin, while in a debit-based scenario [15], the bank diminishes customer's bank account when the coin is withdrawn.

In this Chapter we propose an off-line, merchant specific and debit-based micropayment scheme (that we call *μEasyPay*) to provide an effective and efficient

solution for the micropayment problem. The solution makes use of specific coins to be used to pay for items purchased in a single merchant. The coin withdrawn process diminishes the balance of customer's bank account, so we consider it a debit-based system. Besides, coins can be fully configured as regards to the number and the attached value of microcoupons they have to hold, according to the pre-agreed information between customer and merchant. Customers can use as many microcoupons in a single payment transaction as to meet the worth price of the item they want to purchase. It really improves the efficiency of the micropayment transaction and provides flexibility to pay items of different values. These coins are anonymous, unlinkable and untraceable, so customers are protected from the disclosure of their real identity when they purchase items from merchants. Moreover, customers cannot be traced by merchants or the bank, meaning that customers are protected from being profiled, e.g. for advertisement or malicious purposes. The scheme also protects merchants and bank from misbehaving customers. Therefore, the scheme detects and avoids the reuse of microcoupons and also protects from forging. Moreover, we fully define a solid security model for the micropayment scenario we are dealing on as it has not yet been very well described in previous works in this field. We adapt the definition of fairness described in [67, 68] to the terms of micropayment, because the general fairness definition does not fit the concrete and special requirements requested by these type of payments. Our fairness definition includes the special trade-off between security requirements and efficiency features keeping in mind that parties should assume some controlled and limited risk in exchange for an improvement on the performance. These security requirements will be proved in Chapter 7 by a formal analysis and fairness property will be proved also by an automatic formal methodology (based on the use of Colored Petri Nets (CPNs)).

One of the services in which the micropayment scheme can be applied is the Location-Based Services (LBSs). These type of location-aware and context-aware services make use of the user's location to provide value-added services related to the location where they are located. If users receive a better service related to their surroundings and also an improved security, they may agree to pay for these services. This way, LBS providers can also increase their revenue. Therefore, we state that our micropayment scheme meets the required functionalities for this scenario, such as efficiency and scalability, as well as the legal requirements, usually related to security and privacy. Regarding to legal requirements, we have reviewed the applicable legal framework for LBSs and we show how special features from our proposal meet it. Concerning the performance, we will provide a real performance analysis with actual and commercial mobile devices in Chapter 10 to prove that our micropayment solution meets also the efficiency requirements in a production scenario.

The Chapter is organized as follows. In Section 3.2 we provide an overview

about the scenario in which our micropayment proposal relies, describing the functional model along with the required security model and the security requirements pursued by our micropayment scheme. In Section 3.3 we review the related work in this field. Section 3.4 summarizes the micropayment proposal with a brief description of each involved protocol. Besides, this Section gives a description about the structure of the coin and the life cycle to control when each party can execute a concrete protocol. Section 3.5 contains the full description of the proposal. As a case example in which the proposed micropayment can be used, in Section 3.6 we show how our proposal fits both functional and legal requirements as to be applied to a LBS. Finally, 3.7 concludes the Chapter. Besides the contents of this Chapter, by convenience, we have moved aside the security analysis to the Chapter 7 and the full scheme implementation and performance analysis to the Chapter 10.

3.2 Overview: Scenario and Security Model

In this Section we describe the typical scenarios of a micropayment scheme, taking into account the involved entities and the common protocols among them [69–72]. Moreover, we list the functional features that an ideal micropayment system must accomplish and we also define the security model with the set of considered security requirements.

3.2.1 Micropayment Scenario

All the micropayment scenarios share, at least, the following three common entities: the customer (or payer), the merchant (or payee) and the bank (or broker). Hence, a micropayment system allows a customer to acquire something from the merchant while the bank is the entity who is in charge of issuing coins to customers and deposit back money to merchants.

In the literature, the micropayment scenario is often classified depending on the given use to the coins, based on when the coin validation is performed and based on the provided control to the funds. Regarding to the use given to the coins, we can define two different scenarios depending whether the coin is specific for a single-merchant (or for a specific product) or it is a generic coin to use at different merchants (thus, it is bound neither to a single merchant nor to a single product). Besides, a micropayment scheme could be defined to use either on-line or off-line validations depending on when merchants are allowed to (or are committed to) ask for the validity of the received coins from customers. If the bank is involved in every payment transaction, the system is called on-line due to the fact each payment is verified by the bank. Otherwise, the system is called off-line because the merchant does not perform an immediate validation with the bank at the moment of receiving

coins. Finally, a micropayment could be credit-based or debit-based. In the former scenario, the customer will be charged by the bank when the merchant deposits the received coin. Instead, in the latter scenario, the bank decreases the customer's funds from her bank account at the moment of the withdrawn of the coin.

The choice on the type of scenario depends mainly on the wished efficiency, the security and the role of the micropayment itself during the service provisioning in which it is involved. Therefore, it is claimed that a scenario with specific coins is more efficient even though it reduces the perceived flexibility by customers. Indeed, a micropayment in which the bank is not involved at every payment transaction is obviously more efficient and scalable. Because we want to provide an efficient micropayment suitable to purchase low-value electronic data (and probably data coming from nearly to real-time services), the scenario we are dealing in this Chapter takes into account features from each discussed scenario. Thus, we consider an off-line and debit-based micropayment scenario with specific coins to be used to pay single merchants.

3.2.2 Functional Model

A set of functional features are required for micropayment schemes [69, 72]. These schemes should minimize the *transactional costs* of each microtransaction in order to be a small fraction of the amount transferred during the payment. This fact requires adjustment in the following fields.

Type of coin. Generally, in terms of efficiency and performance, it is preferred a micropayment scheme using specific coins rather than generic coins.

Reduce the number of interactions during the payment. The interactions among parties are one of the components in any exchange that should be reduced as it consumes resources (e.g. time, bandwidth, storage, etc.). In order to reduce them, a micropayment scheme must be able to perform off-line payments, in which the involvement and the mediation of the bank is not required.

Decrease the volume of data. Along with the number of interactions, the volume of data transferred is required to be minimized since we want to preserve the resources, such as the bandwidth or the storage.

Minimize computation costs. The computational complexity should be minimized, so it is required that a micropayment scheme should not rely on expensive asymmetric cryptography, at least during the payment phase, i.e when the customer transfers coins to the merchant. For instance, a hash function is about 100 times faster than a RSA signature verification and about 10000 times faster than RSA signature gener-

ation [13]. So its use should be limited or amortized due to its processing time and certificate management costs.

Keep away from using tamper-proof devices. The micropayment scheme should not rely on tamper-proof devices since they are really costly devices compared with general purpose devices and they could not be either attainable or available to all customers.

***Definition* 3.2.1** *(Efficiency). The micropayment scheme is efficient if it accomplishes all of the following functional requirements: the exchange during the micropayment has a reduced number of interactions between customer and merchant; the amount of data transferred within the payment execution is reduced; the computational costs are low; the scheme does no need the use of tamper-proof devices; and the scheme makes us of specific coins.*

Functional features deal deeply with efficiency and cost minimization but they are usually opposed to the security requirements that must be achieved by general payment schemes. Therefore, a micropayment differs from a normal payment in the fact that former method involves low-value transactions while the latter often is used in scenarios in which goods or services have higher value. Therefore, as stated by the above list of functional requirements, one of the objectives to improve efficiency is to reduce the amount of cryptography, specially asymmetric one. However, this fact could impact on the scheme's overall security.

***Remark* 3.2.1** *In a micropayment scheme, since it involves purchases of low-value items, the security measures must be relaxed (but keeping them controlled) in order to minimize costs, meaning that a micropayment scheme must be a trade-off between security and efficiency.*

3.2.3 Security Model

We describe the security model beginning from the definition of the algorithms and protocols that composes the micropayment scheme for the scenario we are dealing on, along with the parties involved in them. Then, we describe and define the security properties a micropayment scheme must accomplish. They are defined following the adversarial model where and adversary is a probabilistic polynomial algorithm (p.p.t. algorithm hereinafter) that can take the role of either malicious customers or malicious merchants in order to corrupt the protocols. Our security model also considers the fact that the adversary can also corrupt the bank to acquire more information about the customer. Finally, we show that security requirements are generally opposed to some of the functional requirements described in §3.2.2.

***Definition* 3.2.2** *(Micropayment Scheme). An off-line, merchant-specific and debit-based micropayment scheme (μEasyPay) involving a customer (or payer) ($\mathscr{C} \in \{\mathscr{C}_0 \cdots \mathscr{C}_n\}$), a merchant (or payee) ($\mathscr{M} \in \{\mathscr{M}_0 \cdots \mathscr{M}_n\}$) and a bank (or broker) ($\mathscr{B}$), consists on a set of algorithms and interactive protocols such as {*`Setup`*, *`BRegistration`*, *`InitialReq`*, *`Withdrawal`*, *`Spend`*, *`Deposit`* and *`Refund`*}.*

- `Setup`. It is a probabilistic algorithm performed by all the involved parties considering an input security parameter (1^K). As a result, all parties output a key pair $(kp_{\mathscr{P}} = sk_{\mathscr{P}}, pk_{\mathscr{P}})$ from a public key cryptographic algorithm (such as RSA) together with a digital certificate from a trusted Certification Authority (CA) certifying the public key.

- `BRegistration`. It is a deterministic and interactive protocol executed by merchants and customers being interested to spend funds certified by $\mathscr{B}$. Such parties take as input their own digital certificate from a recognized CA and $\mathscr{B}$ proceeds to open an account (`acc`) for them, linking the provided digital certificate to it. In addition, $\mathscr{C}$ must pay some money in at the moment to open the account.

- `InitialReq`. It is a probabilistic and interactive protocol between $\mathscr{C}$ and $\mathscr{M}$. $\mathscr{C}$ declares her interest on the services offered by $\mathscr{M}$, agreeing on a common information data. The common information element contains a list of terms and conditions about this long-term relationship between both parties, along with a set of timestamps to indicate when each protocol and algorithm can be executed by parties. It also defines the value that should have each coin. As a result, $\mathscr{C}$ obtains an identifier to be used afterwards by her to run the `Withdrawal` protocol.

- `Withdrawal`. It is a probabilistic and interactive protocol between $\mathscr{C}$ and $\mathscr{B}$. $\mathscr{B}$ takes as input its private key ($sk_{\mathscr{B}}$) while $\mathscr{C}$ uses the corresponding public key $(pk_{\mathscr{B}})$ and the pre-agreed common information. At the end of the protocol, according to the common information and in case she has enough funds at her bank account ($\mathtt{acc}_{\mathscr{C}}$), $\mathscr{C}$ receives a coin ($\mathbb{C}$) with a number of microcoupons ($\mathtt{mc} \in \mathbb{C}$). As a consequence, her bank account is diminished according to the pre-agreed value. The identifier of $\mathbb{C}$ is not visible to $\mathscr{B}$ as the protocol involves a partially blind signature. If something fails, $\mathscr{B}$ stops the protocol and returns *fail*. $\mathscr{B}$ outputs its view $V_{\mathscr{B}}^{\mathtt{Withdrawal}}$ of the protocol.

- `Spend`. It is a deterministic, interactive and off-line protocol ($\mathscr{B}$ does not take part in it) involving $\mathscr{C}$ and $\mathscr{M}$. $\mathscr{C}$ uses the withdrawn coin ($\mathbb{C}$) while $\mathscr{M}$ takes as input the list of microcoupons already spent from $\mathbb{C}$ (if any). $\mathscr{C}$ spends a set

of microcoupons ($\mathcal{L}_S \in \mathbb{C}$) in exchange of the requested item. Therefore, $\mathcal{M}$ obtains a list of microcoupons from $\mathcal{C}$ ($\mathcal{L}_S \in \mathbb{C}$). If something fails, $\mathcal{M}$ stops the protocol and returns *fail*. $\mathcal{M}$ outputs its view $V_{\mathcal{M}}^{\texttt{Spend}}$ of the protocol.

- `Deposit`. It is a deterministic and interactive protocol between $\mathcal{M}$ and $\mathcal{B}$. $\mathcal{M}$ takes as inputs the list of received microcoupons from customers ($\mathcal{L}_S \in \mathbb{C}$), the corresponding coin ($\mathbb{C}$) and its private key ($sk_{\mathcal{M}}$). $\mathcal{B}$ uses his public key $(pk_{\mathcal{B}})$ and the list of already deposited coupons (if any) from the same coin ($\mathbb{C}$). Through this protocol, $\mathcal{B}$ allows $\mathcal{M}$ to deposit the value of those spent coupons ($\mathcal{L}_S$). If something goes wrong, $\mathcal{B}$ stops the protocol and returns *fail*. $\mathcal{B}$ outputs its view $V_{\mathcal{B}}^{\texttt{Deposit}}$ of the protocol.

- `Refund`. It is a deterministic and interactive protocol involving $\mathcal{C}$ and $\mathcal{B}$. $\mathcal{C}$ uses her not yet spent microcoupons $(\mathcal{L}_{\overline{S}})$, the coin itself ($\mathbb{C}$) and her own private key ($sk_{\mathcal{C}}$). $\mathcal{B}$ takes as inputs his public key $(pk_{\mathcal{B}})$ and the $\mathcal{C}$'s public key $(pk_{\mathcal{C}})$ together with the list of microcoupons already spent by $\mathcal{C}$ and already deposited by $\mathcal{M}$ ($\mathcal{L}_S$). $\mathcal{B}$ returns *fail* if something goes wrong. $\mathcal{B}$ outputs its view $V_{\mathcal{B}}^{\texttt{Refund}}$ of the protocol.

***Definition* 3.2.3** *(Correctness). If an honest $\mathcal{C}$ runs the `Withdrawal` protocol with an honest $\mathcal{B}$, then $\mathcal{C}$ always obtains a signed and properly withdrawn coin composed by the number and value of the microcoupons previously agreed between $\mathcal{C}$ and $\mathcal{M}$ during the `InitialReq` protocol. In addition, the $\mathcal{C}$'s bank account is diminished according to the value of the coin. If an honest $\mathcal{C}$ runs the `Spend` protocol with an honest $\mathcal{M}$ transferring some valid microcoupons from a valid coin, then $\mathcal{M}$ always accepts them. Moreover, if an honest $\mathcal{M}$ executes the `Deposit` protocol with an honest $\mathcal{B}$ using a list of valid microcoupons, then $\mathcal{B}$ always accepts them if all microcoupons are valid. Similarly, if an honest $\mathcal{C}$ runs the `Refund` protocol with an honest $\mathcal{B}$ using a list of valid and not yet spent microcoupons, then $\mathcal{B}$ always accept them and $\mathcal{C}$ is rewarded with the corresponding value in her bank account.*

Below, the general security requirements for privacy-protecting solutions already pointed out in §1.3 will be defined in a formal way, together with other specific requirements for micropayments: unforgeability, anonymity, unlinkability, untraceability, microcoupon reuse avoidance, overspending protection and fairness.

3.2.4 Unforgeability

A micropayment scheme uses coins with a monetary value attached to them, so the system must be unforgeable. Two types of unforgeability games can be defined

depending on the party who tries to forge: either a malicious customer or a malicious merchant.

Consider an unforgeability game 0, in which an adversary $\mathscr{A}_0$ as a p.p.t. Turing Machine acts in the system taking the role of a malicious customer (or a coalition of them) with all the public parameters, so:

- $\mathscr{A}_0$ runs the `Withdrawal` protocol with the honest bank.
- $\mathscr{A}_0$ runs the `Spend` protocol with an honest merchant.

At some point of the game 0, after acquiring a valid coin ($\mathbb{C}$) from the bank, $\mathscr{A}_0$ tries to spend a microcoupon (mc) not belonging to the coin ($\mathtt{mc} \notin \mathbb{C}$). $\mathscr{A}_0$ could try to forge either the coin or the list of microcoupons by different ways: modify the number of microcoupons within the coin; increase the value attached to microcoupons; generate more coupons beyond the coin limits; or create coins. The objective is the same in all cases: try to spend more coupons during the `Spend` protocol than the number (or value) of microcoupon rightfully withdrawn by using the `Withdrawal` protocol. So, it is required that an adversary $\mathscr{A}_0$ playing the unforgeable game 0, has negligible probability on be accepted by an honest merchant by means of the use of the `Spend` protocol.

Similarly, let us define an adversary $\mathscr{A}_1$ as a p.p.t. Turing Machine playing an unforgeability game 1, acting as a malicious merchant, so:

- $\mathscr{A}_1$ can execute the `Spend` protocol with an honest customer.
- $\mathscr{A}_1$ can execute the `Deposit` protocol with the honest bank.

At some point of the game 1, $\mathscr{A}_1$ could try to deposit more coupons than the number he had received from the honest customer. Then, it is required that the probability for an adversary $\mathscr{A}_1$ to successfully deposit more microcoupons than the number he had received from an honest customer is negligible.

***Definition* 3.2.4** *(Unforgeability). The micropayment scheme is unforgeable if no p.p.t. adversary $\mathscr{A}$ ($\mathscr{A}_0, \mathscr{A}_1$) can win none of the unforgeability games with non-negligible probability (in* `K`*).*

3.2.5 Anonymity

Anonymity deals with the secrecy of the customer identity in the system in such a way that the customer does not want to be identified when she executes actions with the scheme. In case of micropayments, however, there is involved the bank entity, who is in charge of the storage and management of the customer's bank accounts

(thus, their funds). So, the customer has to authenticate herself with her real identity to manage her funds and withdrawn a coin by means of the use of the `Withdrawal` protocol. Therefore, anonymity in the scenario we are dealing on is referenced to the `Spend` protocol among customers and merchants. Hence, the customer who uses the `Spend` protocol wants to be anonymous since she does not want to either reveal any information or to allow merchants to collect any data related to her identity that could be used by them to infer her real identity.

Formally, let us define an adversary $\mathscr{A}$ as a p.p.t Turing Machine playing the anonymity game 0 and taking the role of a malicious merchant or a coalition of them. In this game, $\mathscr{A}$ has all the parameters to play with the scheme, so:

- $\mathscr{A}$ can execute the `Spend` protocol with an honest customer.

Suddenly, $\mathscr{A}$ tries to discover the identity of the customer exploring all the data exchanged within the `Spend` protocol (allocated into the adversary's view $\mathcal{V}_{\mathscr{A}}^{\texttt{Spend}}$). Then, it is required that $\mathscr{A}$ has negligible probability to obtain the real identity of the customer.

***Definition* 3.2.5** *(Anonimity). The micropayment scheme provides anonymity to an honest customer if there is no p.p.t. adversary $\mathscr{A}$ able to win the anonimity game 0, obtaining her real identity during the* `Spend` *protocol, with non-negligible probability (in* `K`*).*

3.2.6 Unlinkability

In general, unlinkability deals with the impossibility to attribute different actions performed by the same customer while this customer remains anonymous. It means that an adversary neither can link different operations nor he can know the identity of the customer who has performed those actions. In the micropayment scenario we are dealing on, we can differentiate two different unlinkability instances. On one hand, it must not be possible for the merchant to link two different executions of the `Spend` protocol made by the same customer using different coins ($\mathbb{C}_1$ and $\mathbb{C}_2$). On the other hand, it must be also unfeasible for the merchant to link a `Spend` procedure to the `Withdrawal` instance in which the spent coin had been issued.

Regarding to the first unlinkability case, consider an adversary $\mathscr{A}_0$ be a p.p.t. Turing Machine taking the role of a malicious merchant or a coalition of them. Besides, let us consider two different customers: $\mathscr{C}_1$ and $\mathscr{C}_2$. Each of them holds a coin $\mathbb{C}_1$ and $\mathbb{C}_2$, resp. As usual, let us consider a game 0 in which:

- $\mathscr{A}_0$ can execute the `Spend` protocol with either $\mathscr{C}_1$ or $\mathscr{C}_2$.

Then, at some point of the game, $\mathscr{A}_0$ receives a `Spend` protocol request from the customer b $\left(\text{being } b \in \{0,1\}\right)$ and $\mathscr{A}_0$ stores exchanged data into a protocol view $\mathcal{V}_{\mathscr{A}_0}^{\texttt{Spend}_b}$. $\mathscr{A}$ outputs randomly $b = \{0,1\}$. Suddenly, $\mathscr{A}_0$ receives another call to the `Spend` protocol from the customer b' $\left(\text{being } b' \in \{0,1\}\right)$ and also stores the protocol transcript into another view $\mathcal{V}_{\mathscr{A}_0}^{\texttt{Spend}_{b'}}$. As before, $\mathscr{A}_0$ outputs $b' = \{0,1\}$. Then, it is required that the advantage for $\mathscr{A}_0$ in the success probability that $\mathscr{C}_b \equiv \mathscr{C}_{b'}$ must be negligible. It means that $\mathscr{A}_0$ cannot win the unlinkability game 0 with a better probability to guessing between two values.

As a result, $\mathscr{A}$ has a view $\mathcal{V}_{\mathscr{A}}^{\texttt{Spend}}$ containing all the data exchanged during that spend transaction.

Concerning the second unlinkability instance, consider an adversary $\mathscr{A}_1$ be a p.p.t. Turing Machine involved in the unlinkability game 1, acting as a malicious merchant, or a coalition of them, and with the collaboration of a corrupted bank. This latter can provide $\mathscr{A}_1$ with views containing transcripts of previous `Withdrawn` protocol executions. Supposing that $\mathscr{A}_1$ has all the public parameters to play with the system, consider that:

- $\mathscr{A}_1$ can execute the `Spend` protocol with honest customers.

Consider $\mathscr{A}_1$ has two views available $\left(\mathcal{V}_{\mathscr{B}}^{\texttt{Withdrawal}_0} \text{ and } \mathcal{V}_{\mathscr{B}}^{\texttt{Withdrawal}_1}\right)$ provided by the corrupted $\mathscr{B}$. At some point of the game, $\mathscr{A}_1$ outputs a bit $b = \{0,1\}$. Then, $\mathscr{A}_1$ executes a `Spend` protocol with the honest customer b who owns the coin contained in $\mathcal{V}_{\mathscr{B}}^{\texttt{Withdrawal}_b}$ and outputs another bit $b' = \{0,1\}$. Similarly to the game 0, it is required that the advantage for $\mathscr{A}_1$ in the success probability that $\mathscr{C}_b \equiv \mathscr{C}_{b'}$ must be negligible.

***Definition* 3.2.6** *(Unlinkability). The micropayment scheme is 2N-unlinkable if no p.p.t. adversary $\mathscr{A}(\mathscr{A}_0, \mathscr{A}_1)$ can win any unlinkability game with non-negligible linkability advantage over a random guess of a coin (in* `K`*).*

3.2.7 Untraceability

Related to the unlinkability property, we can define the untraceability security requirement as the indistinguishability of the past and future payment transactions of the identity of the customer with the knowledge of the current transaction. It means that no adversary (or a coalition of them) must be able to identify the customer by linking transactions she had performed or she will perform by using the micropayment scheme. So, it is similar to make a profile of past, current and future actions with relation to the customer's identity (profiling). Therefore, in general,

untraceability implies the accomplishment of both anonymity and 2N-unlinkability properties, as we have defined them above.

Based on the unlinkability game 0 from §3.2.6, let us define the untraceability game, in which $\mathscr{A}$ cannot state whether $\mathscr{C}_b \equiv \mathscr{C}_{b'}$ with non-negligible advantage over random guess as well as $\mathscr{A}$ cannot state about whether $\mathscr{C}_b \equiv \mathscr{C}_{b'}$ is actually $\mathscr{C}_1$ or $\mathscr{C}_2$, with non-negligible advantage over random guess.

***Definition* 3.2.7** *(Untraceability). The micropayment scheme protects customers from traceability while using the* `Spend` *protocol if there is no p.p.t. adversary $\mathscr{A}$ able to win the untraceability game with non-negligible probability (in* `K`*).*

It is similar to say that $\mathscr{A}$ can break neither the anonymity of the system w.r.t. the Definition 3.2.5 nor the 2N-unlinkable property w.r.t. the Definition 3.2.6 with non-negligible probability (in `K`).

3.2.8 Microcoupon Reuse Avoidance

Typically, in the literature related to micropayments, proposals define the double-spending detection and/or avoidance as the property to detect and/or avoid a customer spends more than once the same coin (or microcoupon). However, in the security model we follow in this Chapter, we generalize the sense attached to this security property in order to include also the fact that a merchant could try to request a deposit of the same coin (or microcoupon) more than once. Therefore, the micropayment scheme must avoid, or at least detect, that neither customer nor merchant are allowed using a microcoupon already used during a previous transaction. Thus, this fact affects both customers and merchants in a sense that neither a customer can spend the same microcoupon more than once nor a merchant can deposit the same microcoupon more than once.

Formally, let us consider an adversary $\mathscr{A}_0$ running a microcoupon reuse game 0 with the role of a misbehaving customer or a coalition of them. $\mathscr{A}_0$ has all the public parameters to be involved in the system, so:

- $\mathscr{A}_0$ can execute the `Withdrawal` protocol with the honest bank.
- $\mathscr{A}_0$ can execute the `Spend` protocol with an honest merchant.

During the game 0, $\mathscr{A}_0$ runs the `Spend` protocol using a microcoupon already spent with the merchant. Then, it is required that the probability for $\mathscr{A}_0$ to successfully conclude the `Spend` protocol without the merchant detecting that $\mathscr{A}_0$ has used an already spent microcoupon is negligible (in `K`).

Similarly, consider another adversary $\mathscr{A}_1$ playing the microcoupon reuse game 1 and taking the role of a malicious merchant or a coalition of them. As before, $\mathscr{A}_1$ has all the required parameters to participate in a protocol execution, so:

- $\mathscr{A}_1$ can execute the `Spend` protocol with an honest customer.
- $\mathscr{A}_1$ can execute the `Deposit` protocol with the honest bank.

During the game 1, $\mathscr{A}_1$ tries to cheat the bank running the `Deposit` protocol with a microcoupon already deposited. Then, as before, it is required that the probability for $\mathscr{A}_1$ to deposit again an already deposited microcoupon without the bank detecting it is negligible (in K).

***Definition* 3.2.8** *(Microcoupon reuse detection and avoidance). The micropayment scheme detects and avoids the microcoupon reuse if there is no p.p.t. adversary $\mathscr{A}$ $(\mathscr{A}_0, \mathscr{A}_1)$ able to win any micropayment reuse game using the same microcoupon without the verifier (for instance, either the merchant or the bank) detecting it with non-negligible probability (in* K*).*

3.2.9 Overspending Protection

Any payment method must guarantee that the electronic coins have been created with real funds from customer's bank accounts at the moment of withdrawal (debit-based scenario) or it must assure somehow that the bank account to be charged will have sufficient funds at the deposit time to pay the merchant (credit-based scenario). In the former case, it is the bank who is in charge of control and deny the `Withdrawal` protocol whether the customer does not have enough funds (overspending control in a debit-based scenario). Instead, in the later case, it is required to introduce some measures, such as to set a deterrent to dissuade customer to leave less money in her bank account (overspending management in a credit-based scenario).

As we consider a debit-based micropayment scheme, we can define an overspending game 0 in which an adversary $\mathscr{A}$ as a p.p.t. Turing Machine plays the role of a malicious customer. As usual, $\mathscr{A}$ can take part in the system since it has all the public parameters, so:

- $\mathscr{A}$ can execute the `Withdrawal` protocol with the honest bank.

So, $\mathscr{A}$ runs the `Withdrawal` protocol asking to issue a coin with a value $\mathrm{v}_{\mathscr{A}}^{\texttt{Withdrawal}}$ while in her personal bank account there are only $\mathrm{v}_{\mathscr{A}}^{\texttt{acc}}$, where $\mathrm{v}_{\mathscr{A}}^{\texttt{acc}} < \mathrm{v}_{\mathscr{A}}^{\texttt{Withdrawal}}$. Then, it is required that the probability for an adversary to withdraw a coin with a higher value than the value she is holding in her bank account is negligible.

***Definition* 3.2.9** *(Overspending protection). The micropayment scheme protects from overspending if there is no p.p.t. adversary $\mathscr{A}$ that can win the overspending game 0 with non-negligible probability (in* K*).*

3.2.10 Fairness

In electronic commerce scenarios, such as in those where one item is exchanged for another, there is present an important security requirement: the fairness of the exchange. An exchange is fair if at the end of the exchange, either each player receives the item it expects or neither player receives any additional information about the other's item [67, 68]. It represents the classical and general definition of fairness, usually attached to non-repudiation protocols, i.e. protocols where non-repudiation evidences are accumulated during the exchange to enable the settlement of any future disputes.

Solutions to the fair exchange fall into two categories [67,68]: *gradual exchange* protocols, where the probability of correctness is gradually increased over several rounds of communications, and *third party* protocols which make use of an on-line or off-line Trusted Third Party (TTP).

Concerning gradual exchange protocols, it is generally pursued the probabilistic fairness, as defined in [73] for the case of non-repudiation (NR) protocols:

***Definition* 3.2.10** *A non-repudiation protocol is ϵ-fair if and only if the probability that at the end of a protocol execution either Alice got the NR of receipt evidence for the message m, and Bob got the corresponding message m as well as the non-repudiation of origin evidence for this message, or none of them got any valuable information, is $\geq 1-\epsilon$ (being $\epsilon \in [0,1]$).*

However, solutions meeting the gradual exchange and the probabilistic fairness may have theoretical sense, but they seem to be too cumbersome as they involve a high communication overhead to be considered really practical [68].

Regarding to TTP-based protocols, it is considered that a fair exchange may present either *strong* or *weak* fairness [67]. Following, we give the definition of both types of available fairness for TTP-based protocols.

***Definition* 3.2.11** *(Strong fairness). Upon protocol finalization, all honest parties have the corresponding requested items or none of them have gained additional information.*

***Definition* 3.2.12** *(Weak fairness). Upon protocol finalization, all honest parties have the corresponding requested item or honest parties have enough evidences to prove to an arbiter that they have behaved honestly.*

Protocols using TTPs may cause a bottleneck problem as a consequence of the constant involvement of that third party. To minimize this fact, it is desirable to reduce the involvement of that party. As a result, the *optimistic* protocol approach

was proposed [74], in which the TTP is only invoked in case of exceptional circumstances where an unfair situation has come up, i.e. the exchange does not end successfully for some involved party.

Despite TTP-based protocols with an optimistic approach reduce the required involvement of the third party to such cases in which the fair exchanged may be compromised, they need a special party to resolve these conflicts and some protocols to ask for a fairness recovery. As a consequence, the security of the protocol grows up, but also the associated cost. Moreover, non-repudiation protocols have to use digital signatures and asymmetric cryptography, or similar means of authentication of data and identification of parties in order to accumulate non-repudiation evidences proving that parties have behaved correctly. These kind of protocols very often have been designed to exchange valuable items, so they require higher levels of security in order to avoid to the maximum the risk of fraud and the corresponding loss for some party. This fact usually is opposed to the efficiency required for some kind of protocols, such as micropayment schemes. Although during a micropayment transaction some items are exchanged for a payment, the current item's value is so small as to claim that parties assume some controlled level of risk of loss whether it benefits the achieved performance. Fairness definitions described above do not include this trade-off. As a result, none general fairness definition fits completely with the particular micropayment requirements focused on efficiency in the exchange of low-value items. So, we think that the fairness property, as to be properly applied to the micropayment scenario, must be redefined as to include other notions such as reputation, controlled risk of loss and low-value items.

***Definition* 3.2.13** *(Controlled fairness enforcement). Upon micropayment transaction involving low-value items, all honest parties have the corresponding item from the others or honest parties being cheated may only suffer a limited and controlled loss never higher than the item value, in front of the risk for cheaters to either suffer a partial loss of reputation or to be denied to achieve the current and later items.*

Summarizing both functional and security models, we can remark the following about an ideal micropayment scheme:

***Remark* 3.2.2** *An ideal micropayment system to purchase low-value items must be secure, off-line, it must protect the anonymity and the untraceability of customers when they purchase that items and it must provide a fair exchange with limited and controlled risk of loss, together with low storage, computation and transactional costs.*

3.3 Related Work

As already said in the introduction, there are several micropayment proposals in the literature that follow the general micropayment scheme but they have differences related to the concrete scenario, as pointed out in §3.2.1. Therefore, Table 3.1 exposes the functional features and security properties of some representative micropayment schemes [11–15].

All the analyzed schemes provide the basic protocols, such as withdrawal and spend, as well as a deposit protocol, although it seems that some schemes do not document well the latter protocol. However, only one of the analyzed micropayments [14] defines a revocation protocol that acts as a simple refund protocol to return or exchange an expired coin.

There are some of the analyzed protocols that deal with a multi-merchant scenario [11, 12]. Paytree [11] proposes a micropayment with the ability to pay to multiple merchants. It is based on the use of binary trees to create and manage coins, in which each node of the tree is in fact a single coin. However, the scheme has an important drawback related to the efficiency, because the amount of data to be stored by customers is very large. In a practical application, in which the number of nodes could be large, the customer may need large amounts of storage. Moreover, regarding to the security, coin reuse cannot be efficiently avoided, although it can be detected afterwards. The other analyzed micropayment to pay to multiple merchants [12] claims that it covers more security requirements, as it provides anonymity to customers and protects them from being traced. However, as it uses asymmetric cryptography during the payment phase, the achieved performance is really weaken.

Regarding to the single-merchant micropayments [13–15], Payword [13] was the first practical micropayment scheme to consider the use of hash chains to build coins. Similarly, and based on Payword, the solution presented in [14] considers the use of multiple hash chains to achieve coins with multiple denominations (i.e different monetary values per chain). As a main difference, solution in [14] uses asymmetric cryptography during the payment phase, so the perceived performance is lower than the efficiency provided by Payword. However, neither of them protects customer's anonymity. The proposal described in [15] improves the security as to provide anonymity to customers but it uses asymmetric cryptography during the payment.

Analyzing other features, all the proposals consider that the bank is not involved during the payment phase, although solution in [12] allows merchants to perform on-line validations with the bank. Concerning the management of customer funds, there are proposals which consider the use of credit-based coin [11, 13, 14] and other ones work as debit-based schemes [12, 15]. It is important to note that, at least for the analyzed protocols in the Table 3.1, debit-based schemes provide both high privacy

Table 3.1: Micropayment solutions - A comparative analysis

	Jutla et al. [11]	Nguyen [12]	Rivest et al. [13]	Wang et al. [14]	Zhao et al. [15]	*μEasyPay*
Scenario	MM	MM	SM	SM	SM	SM
Withdrawal	✓	✓	✓	✓	✓	✓
Spend	single	multi	multi	multi	multi	multi
Deposit	✓	✓	✓	✓	✓	✓
Refund	-	-	-	✓	-	optional
Bank involvement	off-line	on-line/off-line	off-line	off-line	off-line	off-line
Funds	credit	debit	credit	credit	debit	debit
Customer anonymity	-	✓	-	-	✓	✓
Unforgeability	✓	✓	✓	✓	✓	✓
Coupon reuse avoidance	-	✓	✓	✓	✓	✓
Unlinkability	-	✓	-	-	✓	✓
Untraceability	-	✓	-	-	✓	✓
Fairness	-	-	-	-	-	✓
No asymmetric crypto in payment	partial	-	✓	-	-	✓
Payment efficiency	low	medium	high	low	low	high

✓YES , -NO

SM- single-merchant

MM- multi-merchant

levels for customers and more confidence for merchants to being paid in by the bank at the moment of deposit.

Besides protocols analyzed in Table 3.1, we also have reviewed other micropayment proposals [75–79] that achieve different levels of privacy for customers. However, none of the reviewed proposals [11–15, 75–79] achieves (or at least give a discussion) about the fairness property, so we claim that it is one of the main drawbacks of such proposals. Besides, none micropayment scheme presents a well defined trade-off between the provided security and the resulting efficiency. Therefore, new proposals are needed to obtain an efficient but secure micropayment following one of the available scenarios.

With the objective to remark that micropayments may be used by heterogeneous applications and services (often accessed by customers using mobile devices) where low-value items are purchased by customers, we have build a brief list with some examples. The scheme proposed in [80] is designed to pay for IPTV services using RFID modules. The solution described in [81] presents a micropayment scheme to pay for real-time SIP services through new generation networks. In addition, another micropayment scheme [82] defines a solution to be used by mobile devices. Another interesting research line is the application of micropayments to peer-to-peer networks, such as the solution described in [83], which transfers Payword chains between peers. To conclude, it is also interesting the use of the Near Field Communications (NFC) technology to design a mobile payment service as scheme in [84] does.

3.4 *μEasyPay*: Micropayment Scheme

In this section we describe the micropayment scheme, that we call *μEasyPay*. This scheme follows the efficiency requirements stated by the functional model (see §3.2.2) as well as the Definition 3.2.1. It also achieves all the security requirements defined by the security model we are dealing on (see §3.2.3) and improves previous micropayment proposals.

3.4.1 Our Proposal in a Nutshell

The micropayment scheme, for short *μEasyPay*, involves the same parties as the scenario we have described before in §3.2.1: customer ($\mathscr{C}$), merchant ($\mathscr{M}$) and the bank ($\mathscr{B}$). Following the formal Definition 3.2.2, Figure 3.1 shows how the interactions among parties are, as well as the general scheme flow. The first step is to set up the system in which all parties acquires the elements to be involved in the system following a `Setup` algorithm. Besides, both $\mathscr{M}$ and $\mathscr{C}$ must set up a personal bank account at $\mathscr{B}$ by using the `BRegistration` protocol. Note that at least $\mathscr{C}$ has

to pay some money in her bank account. At this point, $\mathscr{C}$ can look for merchants selling interesting services checking a public list. Once she decides with which $\mathscr{M}$ wants to begin a relationship, $\mathscr{C}$ starts the `InitialReq` protocol in order to make an agreement with $\mathscr{M}$ and to download some elements from him. Then, $\mathscr{C}$ can run the `Withdrawal` protocol to anonymously request a coin to $\mathscr{B}$ according to the information agreed between $\mathscr{C}$ and $\mathscr{M}$ during the previous step. As a result, $\mathscr{C}$ obtains an anonymous and untraceable coin that can be used by her to purchase services from $\mathscr{M}$ in exchange of a set of microcupons by using the `Spend` protocol. After receiving a set of microcoupons, $\mathscr{M}$ is allowed to ask for a deposit in his bank account according to the value of those spent microcoupons, following the `Deposit` protocol. Finally, in a similar way as $\mathscr{M}$ does, $\mathscr{C}$ has the chance to run the `Refund` protocol to ask for a reimbursement equivalent to the value of her remaining and not yet used microcoupons. The `Refund` protocol is optional and it will be present if all parties agree and if it makes sense depending on the service offered. For example, in a real-time service (such as a video streaming), the recovery of a piece of video afterwards does not make sense. In such examples, our micropayment scheme really enforces the efficiency but it results on a relaxation of the fairness property because the type of service sought it.

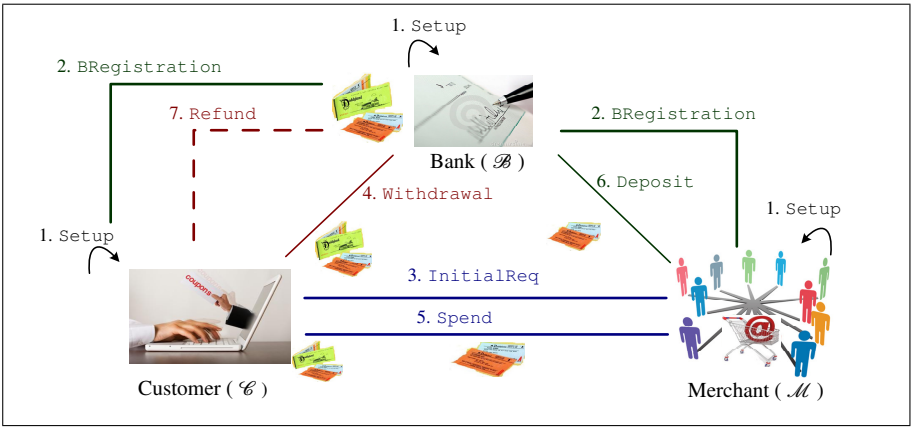


Figure 3.1: Micropayment scenario. Entities and protocol flow among them.

Figure 3.2 shows some time marks we have defined to control the coin's life cycle, i.e. when each protocol can be used. Moreover, these time marks are also useful from the point of view of efficiency, due to the fact that parties can remove coins and its related data when they are no longer valid, saving, this way, storage.

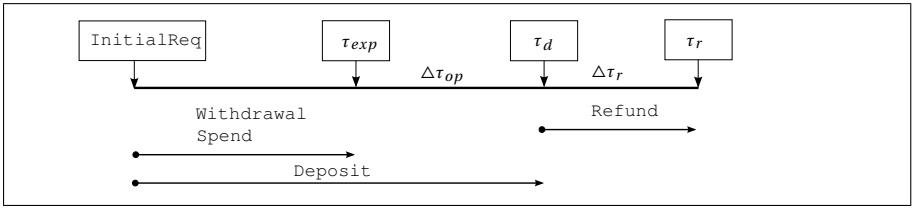


Figure 3.2: Life cycle of coins.

3.4.2 Coin Structure

Figure 3.3 shows the structure of the coin ($\mathbb{C}$) used by our micropayment scheme. It consists on two different elements: a microcoupon list ($\mathbb{C}_\omega$) made by applying the hash chain procedure and the partially blind signature ($\mathbb{C}_{\mathscr{PBS}}$) over the coin identifier .

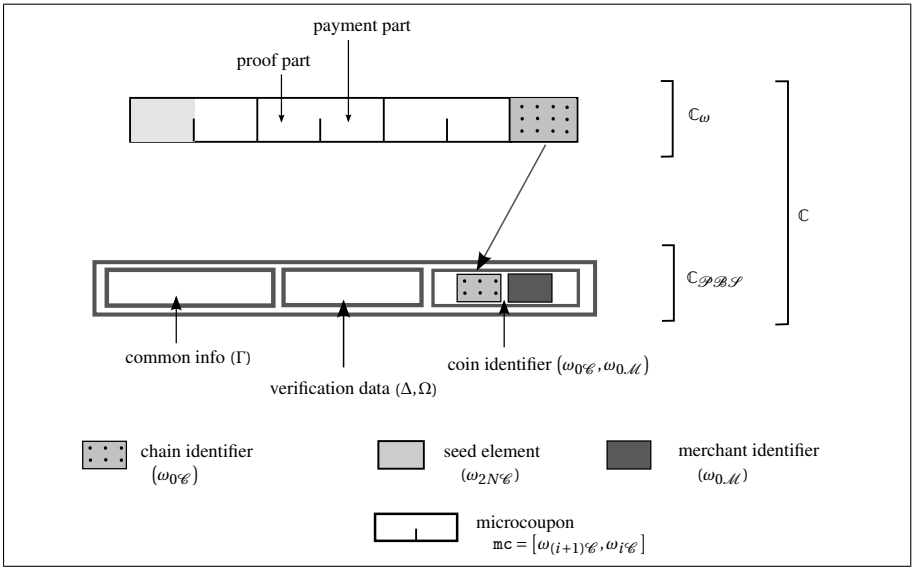


Figure 3.3: Coin structure ($\mathbb{C}$).

On one hand, the $\mathbb{C}_\omega$ element contains $2N+1$ chained elements. Each element is defined as $\omega_{i_{\mathscr{C}}}$, where i denotes its position (index) within the chain. The list is obtained by the use of the hash chain procedure, applying $2N$ hash operations on a random seed element $(\omega_{(2N)_{\mathscr{C}}})$ up to the final one, called *chain identifier* or *root identifier* $(\omega_{0_{\mathscr{C}}})$. These elements are organized in such a way they must be used in pairs, generating a microcoupon $\texttt{mc} = (\omega_{(i+1)_{\mathscr{C}}}, \omega_{i_{\mathscr{C}}})$. The first used is the

payment part, or *payment* mc ($\omega_{i\mathscr{C}}$) while the second one is the *proof part*, or *proof* mc $\left(\omega_{(i+1)\mathscr{C}}\right)$. This way, the *proof* mc proves that the previous *payment* mc are related to the same $\mathbb{C}$ by means of hash chain, i.e. $\mathscr{H}\left(\omega_{(i+1)\mathscr{C}}\right) = \omega_{i\mathscr{C}}$.

On the other hand, the $\mathbb{C}_{\mathscr{P}\mathscr{B}\mathscr{S}}$ element is the result of applying the partially blind signature over the coin identifier $(\omega_{0\mathscr{C}}, \omega_{0\mathscr{M}})$ together with the common information field (Γ). As a result of the signature process, $\mathbb{C}_{\mathscr{P}\mathscr{B}\mathscr{S}}$ contains three elements:

- The common information $\left(\Gamma = \left(2N, v, \tau_{exp}, \Delta\tau_{op}, \Delta\tau_r\right)\right)$, describing the coin's properties agreed between $\mathscr{C}$ and $\mathscr{M}$. It contains the number of microcoupons $\left(N \text{ microcoupons, thus } 2N \text{ hash identifier}\right)$, the unit-value of each of them (v) (thus, both elements defines the total value carried by $\mathbb{C}$) and the list of time marks to limit different time ranges where each protocol can be used. The first one $\left(\tau_{exp}\right)$ marks the time up to $\mathscr{C}$ can spend her microcupons; the second one $\left(\tau_d = \tau_{exp} + \Delta\tau_{op}\right)$ defines the time up to $\mathscr{M}$ is allowed to request a deposit; finally, the last one $\left(\tau_r = \tau_{exp} + \Delta\tau_{op} + \Delta\tau_r\right)$ is the time up to $\mathscr{C}$ can ask for a money refund. After the latter time mark, $\mathbb{C}$ is no longer valid and parties could remove all the data related to $\mathbb{C}$.
- The verification data (Δ, Ω) used to verify the partially blind signature.
- The coin identifier $(\omega_{0\mathscr{C}}, \omega_{0\mathscr{M}})$, where $\omega_{0\mathscr{C}}$ is the microcupon chain identifier and $\omega_{0\mathscr{M}}$ is the merchant identifier. The latter element will be used by $\mathscr{M}$ to prove that he is the actually the right payee for the coin it belongs.

3.5 The Full Specification of the Micropayment

After the given overview about how the micropayment proposal works, now we describe all the involved protocols in detail following the Definition 3.2.2 and the scenario proposed in §3.2.1. Table 3.2 shows the notation used during the protocol description.

3.5.1 System Setup

As a previous step, all parties must be enrolled in a system setup. It has two parts or algorithms: the global system `Setup` and the `BRegistration` protocol.

First, all the parties must obtain a RSA key pair together with a digital certificate from a trusted Certification Authority (CA) following the `Setup` algorithm. This key pair will be also used by $\mathscr{B}$ to use the selected partially blind signature scheme, as described in §2.2.2 and §2.2.3. Besides, merchants publish somewhere a list with the services they are willing to offer to customers together with the terms and conditions and the overall price of them.

Table 3.2: Notation used along the protocol description.

Element	Description
$\mathscr{H}(x)$	One-way collision resistant hash function applied on x
$\mathscr{H}^i(x)$	$\mathscr{H}$ function applied i times iteratively on x
$\mathbb{SK}_{\mathscr{P}}$, $\mathbb{PK}_{\mathscr{P}}$	$\mathscr{P}$'s key pair of a public key cryptosystem
$Cert_{\mathscr{P}}$	$\mathscr{P}$'s digital public key certificate
$\mathbb{S}_{\mathscr{P}}(x)$	$\mathscr{P}$'s signature on the element x
$x \xleftarrow{R} \mathbb{Z}_r$	Element x randomly chosen from $\mathbb{Z}_r$ set
$x \xleftarrow{R} \mathbb{Z}_n^*$	Element x randomly chosen from $\mathbb{Z}_n^*$ set
$\mathbb{K}_S$	Symmetric key $\mathbb{K}_S$
$\mathbb{E}_{\mathbb{K}_S}[x]$, $\mathbb{D}_{\mathbb{K}_S}[x]$	Encryption and decryption of x using $\mathbb{K}_S$
$\mathbb{C}$	Coin issued by $\mathscr{B}$ composed by $[\mathbb{C}_\omega, \mathbb{C}_{\mathscr{P}\mathscr{B}\mathscr{S}}]$
$\mathbb{C}_\omega$	Coupon chain list
$\mathbb{C}_{\mathscr{P}\mathscr{B}\mathscr{S}}$	Partially blind signature on the coin identifier
(e, n)	$\mathscr{B}$'s RSA public key
(d, p, q)	$\mathscr{B}$'s RSA private key
τ_d	Time mark up to $\mathscr{M}$ can request a deposit
τ_r (if it applies)	Time mark up to $\mathscr{C}$ can request a refund
REQ	Request made by $\mathscr{C}$ asking for a concrete good offered by $\mathscr{M}$
PRD	The good send by $\mathscr{M}$ to $\mathscr{C}$ who has requested it

Secondly, both $\mathscr{C}$ and $\mathscr{M}$ must execute the `BRegistration` protocol to establish a bank account in $\mathscr{B}$ and pay a sum into it. Then, $\mathscr{B}$ can link the provided digital certificate ($Cert_{\mathscr{P}}$) to the corresponding bank account ($\texttt{acc}_{\mathscr{P}}$) in order to authenticate parties when it is required (during the `withdrawal`, `Deposit` and `Refund` protocols (if the latter is allowed)).

3.5.2 Initial Request Protocol

Upon $\mathscr{C}$ have found a merchant among the published list (as stated before in §3.5.1) on which she want to establish a relationship, she must begin the `InitialReq` protocol, as described in Figure 3.4. During the process protocol, $\mathscr{M}$ computes a unique identifier ($\omega_{0\mathscr{M}}$) applying a hash function on a secret and random element ($\omega_{1\mathscr{M}}$). The latter identifier must be kept in secret by $\mathscr{M}$ because it is the element required to prove that $\mathscr{M}$ is in fact the proper payee of the coin. In addition, $\mathscr{M}$ completes the information about the terms and conditions and the price with some time marks, as already pointed out in §3.4.2. In this step, thus, $\mathscr{M}$ and $\mathscr{C}$ agree on the common information $\big(\Gamma = \big(2N, v, \tau_{exp}, \Delta\tau_{op}, \Delta\tau_r\big)\big)$. $\mathscr{M}$ signs the random identifier just generated ($\omega_{0\mathscr{M}}$) together with the common info as a commitment that he will eventually provide that service to someone (note that $\mathscr{C}$ has not provided nothing

to $\mathscr{M}$, neither some kind of identification). Finally, both parties store the obtained results in their respective databases.

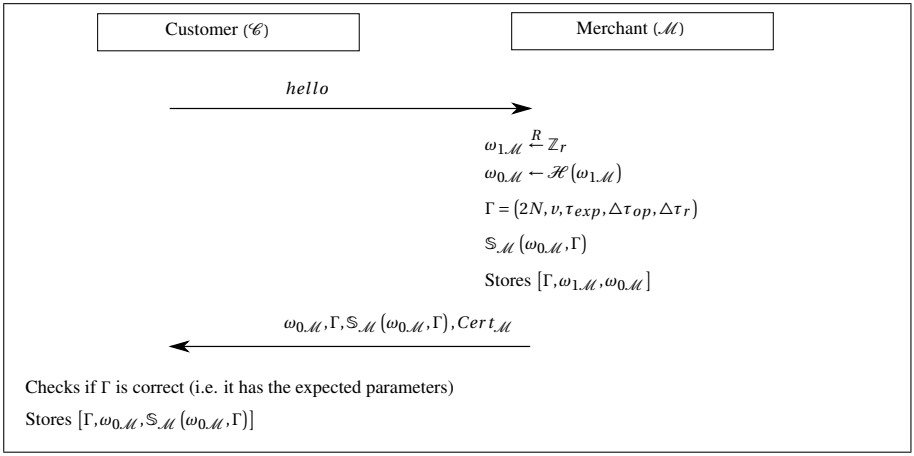


Figure 3.4: `InitialReq` protocol flow.

3.5.3 Withdrawal Protocol

Using as input the data previously obtained through the `InitialReq` protocol, $\mathscr{C}$ runs the `withdrawal` protocol with $\mathscr{B}$, as detailed by Figure 3.5. As a result of the protocol run, $\mathscr{C}$ obtains an anonymous coin ($\mathbb{C}$) even though the protocol runs within an authenticated channel with the real identity of $\mathscr{C}$, due to the fact that $\mathscr{B}$ needs to identify her in order to access her bank account ($\texttt{acc}_{\mathscr{C}}$). However, $\mathbb{C}$ does not contain any information about her identity and the coin identifier ($\omega_{0\mathscr{C}}, \omega_{0\mathscr{M}}$) is hidden to $\mathscr{B}$ as a result of using the partially blind signature scheme (see §2.2.2). Moreover, $\mathscr{B}$ does not have any information about the final look of $\mathbb{C}$ because the process is completed locally by $\mathscr{C}$. $\mathscr{B}$ only can assure that $\mathbb{C}$ is properly withdrawn and ready to use.

The first step that $\mathscr{C}$ must follow to enroll in the `Withdrawal` protocol is to pick a random seed identifier ($\omega_{2N\mathscr{C}}$). This identifier has to be used to create a microcoupon chain applying iteratively a secure cryptographic hash function resistant to collisions up to the last microcoupon ($\omega_{0\mathscr{C}}$) which will be the first part of the coin identifier. As a result, $\mathscr{C}$ obtains a hash chain with $2N$ identifiers $\left([\omega_{i\mathscr{C}}]_{i=1}^{2N} = \omega_{2N\mathscr{C}}, \ldots, \omega_{i\mathscr{C}}, \ldots, \omega_{1\mathscr{C}}\right)$, so a total number of N microcoupons, together with the coin identifier ($\omega_{0\mathscr{C}}$). Next, $\mathscr{C}$ blinds the coin identifier with the previously obtained merchant identifier ($\omega_{0\mathscr{M}}$) and sends the result signed together with the pre-agreed common information ($\mathbb{S}_{\mathscr{C}}(\Gamma, \alpha)$).

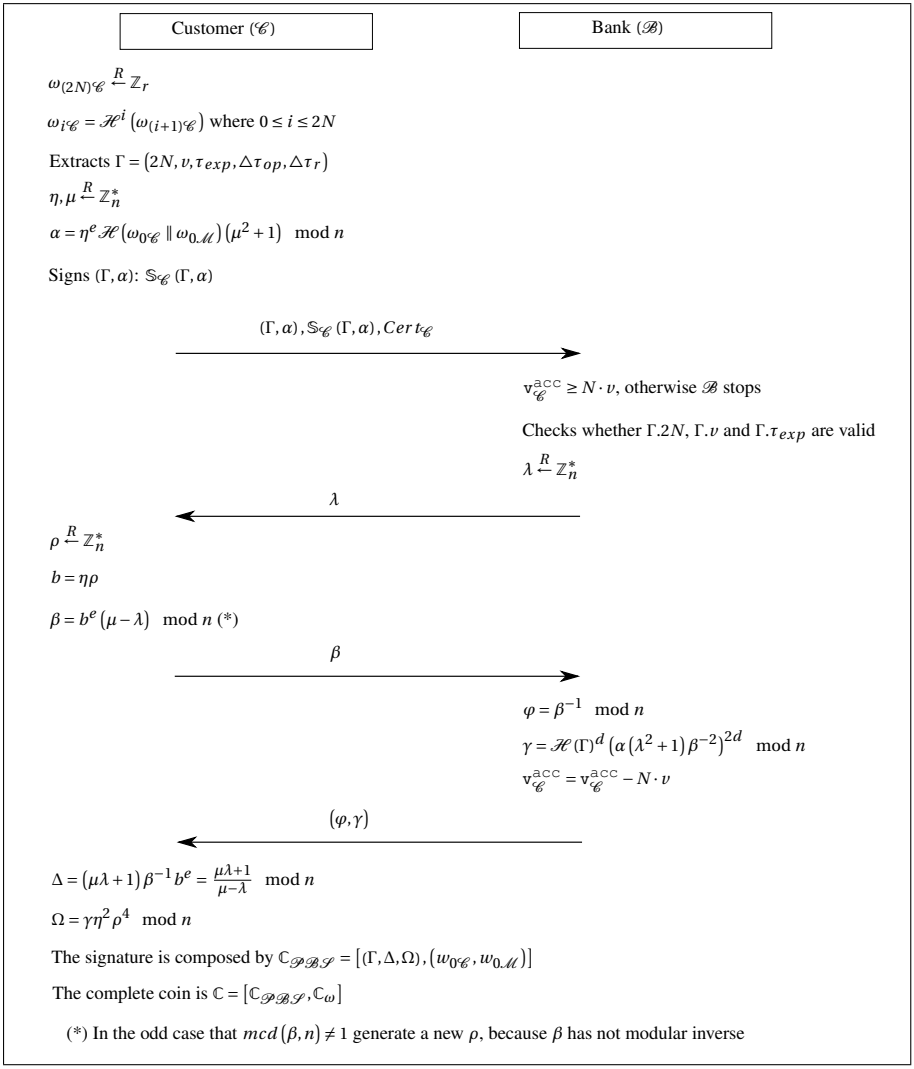


Figure 3.5: `Withdrawal` protocol flow.

Upon receiving the first message, $\mathscr{B}$ checks whether $\mathscr{C}$ has enough funds at her bank account $\left(v_{\mathscr{C}}^{acc} \ge N \cdot v\right)$. After exchanging some parameters, $\mathscr{B}$ during the fourth step of the protocol, in fact diminishes the $\mathscr{C}$'s bank account according to the value of the coin currently withdrawn. Finally, during the last step, $\mathscr{C}$ unblinds and finishes the partially blind signature, obtaining $\mathbb{C}_{\mathscr{PBS}} = [(\Gamma, \Delta, \Omega), (w_{0\mathscr{C}}, w_{0\mathscr{M}})]$ ele-

ment. Therefore, the complete coin specification is $\mathbb{C} = [\mathbb{C}_{\omega}, [(\Gamma, \Delta, \Omega), (w_{0\mathscr{C}}, w_{0\mathscr{M}})]]$.

The partially blind signature $\mathbb{C}_{\mathscr{PBS}} = [(\Gamma, \Delta, \Omega), (w_{0\mathscr{C}}, w_{0\mathscr{M}})]$ can be verified by anyone who knows $\mathscr{B}$'s public key (e, n) verifying whether both sides of the following equation are equal:

$$\Omega^e \stackrel{?}{\equiv} \mathscr{H}(\Gamma)\,\mathscr{H}(\omega_{0\mathscr{C}} \parallel \omega_{0\mathscr{M}})^2 \left(\Delta^2 + 1\right)^2 \mod n \tag{3.1}$$

If this equation holds, the verifier accepts $\mathbb{C}_{\mathscr{PBS}}$, thus the verifier also accepts $\mathbb{C}$. Otherwise, the verifier rejects to receive $\mathbb{C}$ and outputs a *fail* message.

3.5.4 Spend Protocol

Once $\mathscr{C}$ has withdrawn $\mathbb{C}$, she can run the `Spend` protocol described in Figure 3.6 on condition that $\mathbb{C}$ is not expired. This protocol allows $\mathscr{C}$ to spend some microcoupons from $\mathbb{C}$ in exchange of a service from $\mathscr{M}$. $\mathscr{C}$ can spend multiple microcoupons at once if the service worth more than a single unit. This fact has a positive impact on the efficiency, as opposed to other previous proposals where $\mathscr{C}$ had to run as many transactions as the number of coupons needed to accomplish the value of the service. Hence, it is an off-line protocol due to the fact that $\mathscr{B}$ is not involved in any spend transaction. All messages exchanged between parties have been encrypted by means of a secure symmetric cryptosystem using a previous secure key exchange protocol (out of scope). This way, messages are protected from adversaries listening the channels.

The `Spend` protocol begins when $\mathscr{C}$ sends a message with the coin ($\mathbb{C}$), the payment part of the microcoupon she is willing to spend ($\omega_{i\mathscr{C}}$), the index of this payment part (i) and the request (REQ). $\mathscr{M}$ performs some verifications on $\mathbb{C}$: checks whether $\mathbb{C}$ is not yet expired and checks whether the microcoupon was used before (he checks reuse). If all the verifications hold, $\mathscr{M}$ sends the service to $\mathscr{C}$. During the third step, $\mathscr{C}$ sends the proof part of the microcoupon with the corresponding index. As before, $\mathscr{M}$ checks whether all the elements are correct and if it is true, updates his database with the last received microcoupon $\left(\omega_{(i+1)\mathscr{C}}\right)$ and the corresponding index $\left(j' = i + 1\right)$ for later spend transactions.

3.5.5 Deposit Protocol

Figure 3.7 describes the `Deposit` protocol flow. This protocol is executed by $\mathscr{M}$ to ask $\mathscr{B}$ for a deposit in his bank account ($\mathtt{acc}_{\mathscr{M}}$) equivalent to the number and the value of received microcoupons from $\mathscr{C}$. $\mathscr{M}$ is allowed to request a deposit any time before τ_d. Moreover, the `Deposit` protocol can be called by $\mathscr{M}$ as many times he wants, because the protocol allows doing partial deposits. It means that

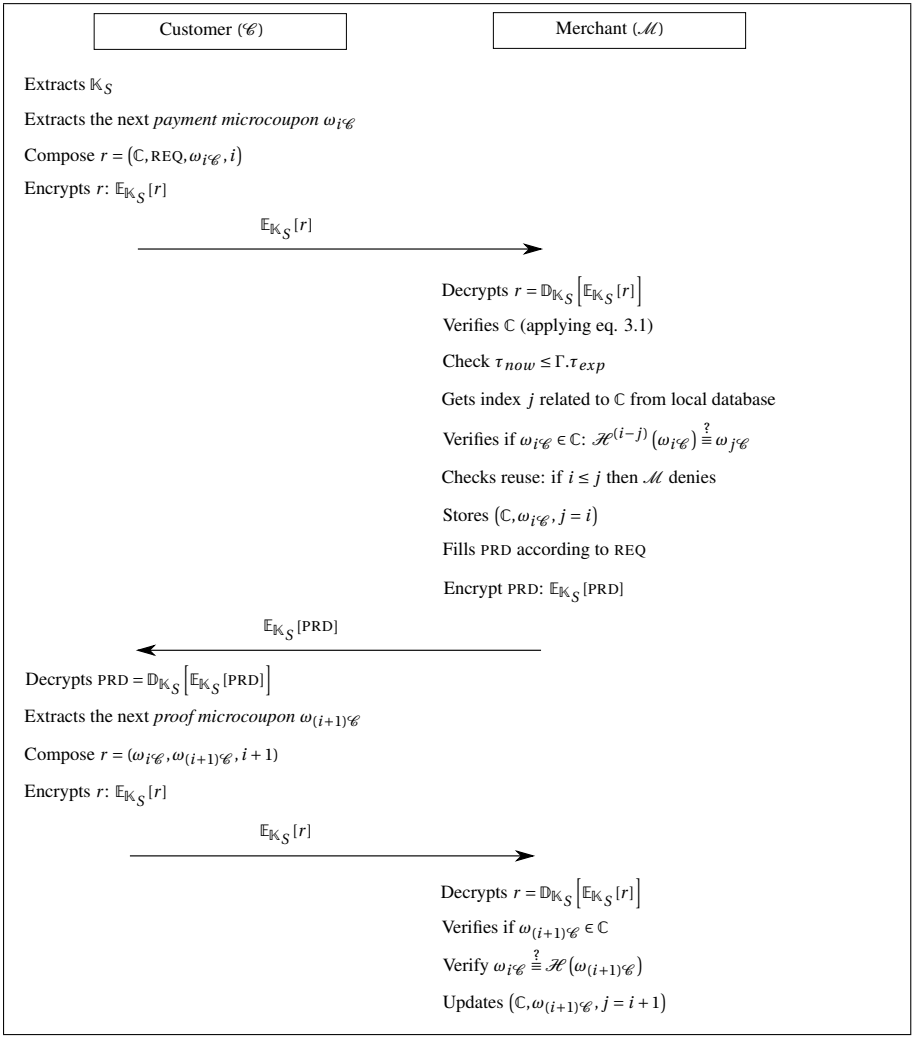


Figure 3.6: `Spend` protocol flow.

it is not necessary that $\mathscr{M}$ has to wait to have all the microcupons of $\mathbb{C}$ to request a deposit. As the `Spend` protocol does, the `Deposit` protocol is able to carry multiple microcoupons at the same transactions.

In order to receive a deposit, $\mathscr{M}$ must send the last received proof microcoupon, the corresponding coin and the secret merchant's identifier for the current coin ($\omega_{1\mathscr{M}}$) to prove that he is the intended receiver of these funds. After several verifications

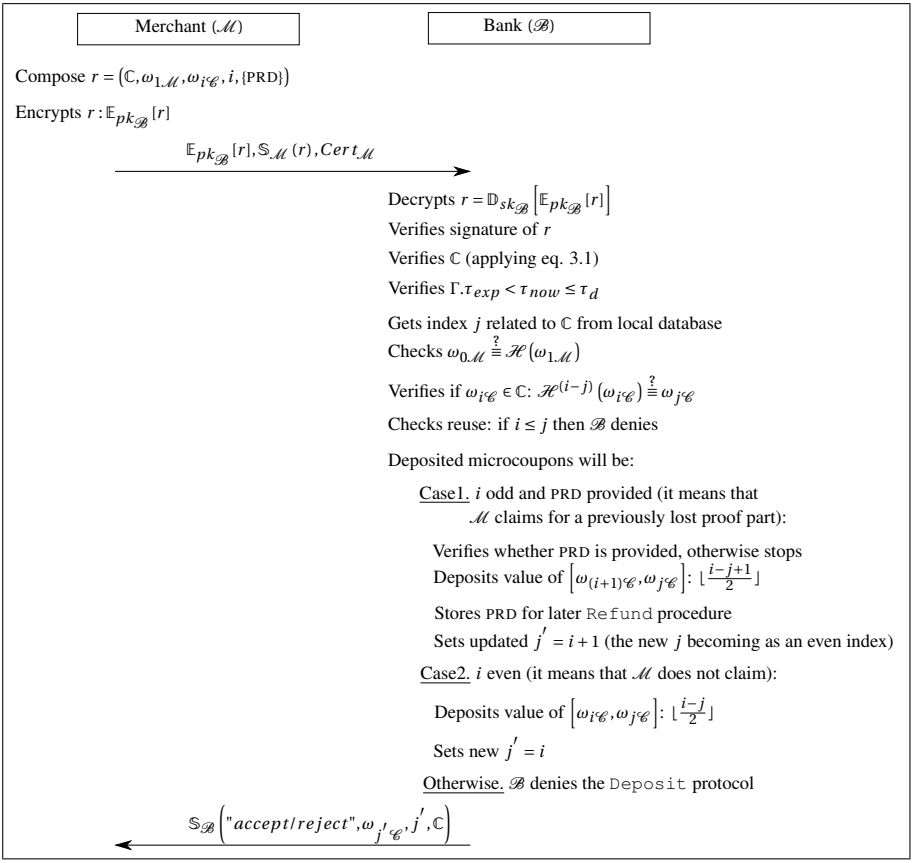
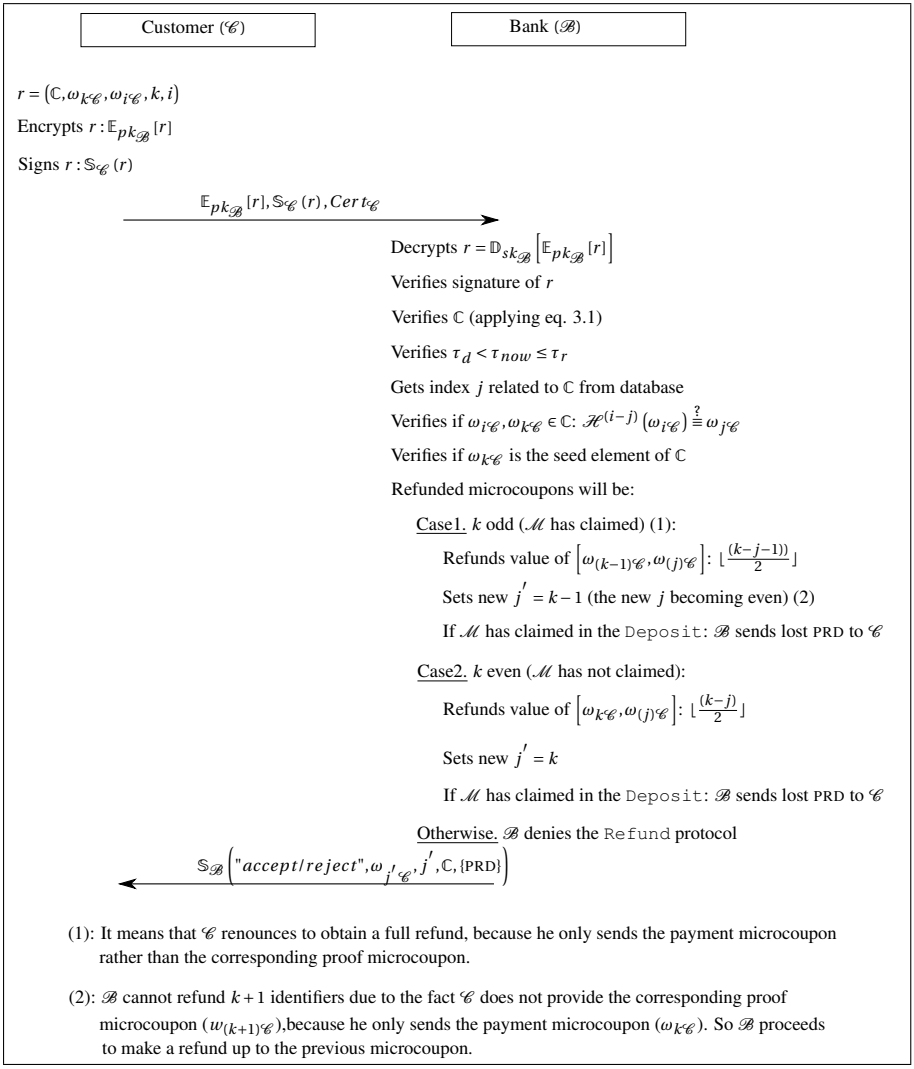


Figure 3.7: `Deposit` protocol flow.

on the validity of $\mathbb{C}$ and after checking for reuse attempts, $\mathscr{B}$ pay the corresponding amount of money into $\mathscr{M}$'s bank account ($\texttt{acc}_{\mathscr{M}}$), depending on whether the presented index is odd or even. Note that, depending on the type of service offered, the `Claim` protocol optionally allows $\mathscr{M}$ to claim for a not yet received proof microcoupon at the moment of sending the requested service (PRD) (the not yet received proof microcoupon could be either lost or stolen). $\mathscr{B}$ stores the lost service to send afterwards to $\mathscr{C}$ during the `Refund` protocol (see §3.5.6). This way, the protocol is able to recover from a possible unfair situation.



(1): It means that $\mathscr{C}$ renounces to obtain a full refund, because he only sends the payment microcoupon rather than the corresponding proof microcoupon.

(2): $\mathscr{B}$ cannot refund $k+1$ identifiers due to the fact $\mathscr{C}$ does not provide the corresponding proof microcoupon ($w_{(k+1)\mathscr{C}}$),because he only sends the payment microcoupon ($\omega_{k\mathscr{C}}$). So $\mathscr{B}$ proceeds to make a refund up to the previous microcoupon.

Figure 3.8: `Refund` protocol flow.

3.5.6 Refund Protocol

The `Refund` protocol is an optional procedure in which $\mathscr{C}$ can ask $\mathscr{B}$ for a reimbursement of her not yet used microcoupons from an expired coin on condition that it must be run after finishing the deposit time, i.e. $\tau_d < \tau_{now} \leq \tau_r$. As usual, this

protocol can carry more than a single microcoupon within the same refund transaction. Figure 3.8 shows that the `Refund` protocol flow looks similar to the `Deposit` protocol. $\mathscr{C}$ must send the seed identifier ($\omega_{2N_{\mathscr{C}}}$) to prove that she really owns the full microcoupon chain, together with the first unused microcoupon. Similarly to the `Deposit` protocol, $\mathscr{B}$ performs several verifications upon receiving the input elements. If all of them hold, the money corresponding to the value of her unused microcoupons is refunded to her bank account ($\texttt{acc}_{\mathscr{C}}$) and a message to confirm it is send back to $\mathscr{C}$. In case $\mathscr{B}$ had received a lost service from $\mathscr{M}$, $\mathscr{B}$ sends it to $\mathscr{C}$ as to recover a possible unfair situation. As already stated before (§3.4.1), the availability of returning a lost service depends on the type of service offered by $\mathscr{M}$ and the agreement between parties.

3.6 Micropayment Application Example: Pay-per-use Location-Based Services

One of the applications in which the micropayment scheme described in §3.4 could be applied is the access to LBSs subject to payment. LBS is a perfect kind of service used in mobility environments to purchase low-value pieces of information about customer's context and location. In this Section we briefly describe what LBSs are and why our micropayment proposal fits on its service requirements, such as security, privacy, functionalities and legal framework of application.

3.6.1 Overview about Location-Based Services and Their Security

LBS are a new kind of pervasive services that make use of location data in a mobile environment to provide their customers specific and valuable information about the context where they are located. For example, customers can ask to LBS providers context-aware or location-aware questions, such as where is the nearest hospital, where their friends are located, etc. However, the area most benefited by the use of LBS is the tourism due to its special features, requirements and needs. Tourists usually demand services because they are not aware of the information related with their surroundings, so they are an important target for the LBS market. Moreover, they usually are willing to pay if this fact provides them valuable information about where they are being located.

From the security viewpoint, it is clear that the use of private data opens many threats related to the customer' security because either providers or observers could have access to both customer identity and her current location (or also where they are not located now) at the same time. This means that customers are worried about being profiled by providers or traced by other parties with commercial or malicious purposes. So, customers do not trust on using LBS subject to payment thus providers

lose an incredible opportunity to make profit and increase their revenue. Indeed, these applications gather private information of the user in order to offer him the service. This information could be used to generate a location profile of the user in order to know his usual movements. Therefore, these type of services need to add techniques to increase customers trust in order to attract them.

Regarding to the anonymity of users (whether the service is subject to payment or not), LBSs can be classified as follows [85]:

- *Anonymous*. The user can be fully anonymous, because the service does not need any type of identification nor pseudonym. For example, a service of meteorological alerts for the city where the user is located.
- *Identified*. It can only work if the user provides his true identity. An example could be an application to alert about a broken protective order by an assailant.
- *Pseudonym based*. The user does not show his true identity but he only shows a pseudonym. As instance, a dating application, where it is not mandatory to show the real identity although other personal data could be shared, like age or sex.

Concerning the location data, in the literature there are different means to protect it from malicious behaviors. One of them is to use obfuscation algorithms to hide the exact location by mixing locations of k users located in the same vicinity. This technique is called *k-anonymization* [86, 87], but it only has sense when there are many other users in the nearby area. The other important trend to hide location is to use a technique called Private Information Retrieval (PIR) [88]. It is a method to completely hide both location and the location-aware request from the provider. Despite it seems a solution that protects the user privacy, it has the drawback that it requires intensive computation and it could require to query the whole location database. So, it is a solution that cannot provide scalability.

Therefore, there are means to try to protect the location data but there are no proposals taking into account the ability to merge the anonymity and the privacy of location data by allow providers to charge their users (thus customers) in a secure, reliable and scalable way. In this field, we state that our micropayment proposal could fulfill this requirements.

3.6.2 Payment Methods to Access LBS

The access to the service offered by LBS providers could be classified depending on the payment method as follows:

- *Free of charge*. The user can send requests to providers free of charge and so, providers have not any profit. However, providers can obtain benefits through advertising banners.
- *Subscription*. The provider charges the user by a subscription that is paid in advance by the user. It is valid for a limited number of accesses or a limited time. It assumes a long term relationship. That is, the whole package should be used by the user or otherwise it will be lost, because a refund is not possible.
- *Full payment*. A full payment is a type of electronic payment scheme designed to pay large amounts of money with plenty of security mechanisms. Since in LBS each request has low cost and we need the maximal efficiency with the minimal cost, it is not a good idea to use a full payment scheme to pay for LBS.
- *Micropayment*. A micropayment fits the requirements of providers and users. Its cost is low, it is suitable for the payment of small amounts of money and the user pays only for the consumed services.

Therefore, it seems that micropayment is the best method to charge users to access LBS.

3.6.3 Legal Framework on Privacy and LBS

Privacy is a special issue in the field of LBS. Mobile device owners must know that they are transmitting their location. Besides, several questions arise: who knows their location? Are they informed in a clear and comprehensible way about the use of their location data?

The fact is that location data are considered personal data. So, the legal framework applied to LBS is the Data Protection Directive (95/46/EC) [89], applied in all cases that personal data are being processed. Location data is defined in the Electronic Privacy Directive as:

> “Location data conveys any data processed in an electronic communications network or by an electronic communication service, indicating the geographic position of the terminal equipment of a user of a publicly available electronic communications service

Moreover, it is also regulated by the article 9 of the Directive 2002/58/EC of the European Parliament and by the Council of 12 July 2002 (together with revisions by 2009/136/EC) [90,91] concerning the processing of personal data and the protection

of privacy in the electronic communications sector. The Whereas 35 of this Directive establishes about this question that:

> "... in addition, digital mobile networks may have the capacity to process location data which are more precise than is necessary for the transmission of communications and which are used for the provision of value added services such as services providing individualized traffic information and guidance to drivers. The processing of such data for value added services should only be allowed where subscribers have given their consent. Even in cases where subscribers have given their consent, they should have a simple means to temporarily deny the processing of location data, free of charge.

The Article 29 Working Party [92] considers that this information should be provided by the party collecting the location data for processing, i.e. by the provider of the value-added service or, where the provider is not in direct contact with the data subject, by the electronic communications operator [93]. In the same way, the Spanish Telecommunications Act (Act 32/2003, November 3rd) in the article 38.3 establishes [94]:

> "Location data can only be processed when it is anonymous or if the provider has the consent of the user or subscriber.

Summarizing, the most important requirements to process personal data (such as the location information) are the customer consent, the anonymity of customers and how her location data will be processed. So, the requirements for service providers to use the location data and how security and privacy features of our micropayment scheme can cover these special requirements to access to LBS providers, are the following:

- Location data related to customers or subscribers of public communications networks or publicly available electronic communications services can be processed when they are either anonymous or with the consent of the users or subscribers.

  The micropayment scheme allows customers to be fully anonymous due to the fact that they never provide their real identity nor the provider can infer it even he tries to process the received data.

- Providers of value-added services must take appropriate measures when they obtain consent to ensure that the person to whom the location data belongs is the same as the person who has given the consent.

  Note that this means that from the point of view of the legal framework, the LBS provider must be able to link requests coming from the same customer in order to prove that this data is also linked to the corresponding consent. Our micropayment scheme allows partial linkability as using microcoupons from the same coin, so providers can check whether every microcoupon is linked to the coin in which customer had deposited her consent, even though the provider has no means to trace these requests to a concrete identity (see Definition 3.2.7 in §3.2.7).

- Location data can only be processed to the extent and for the duration necessary for the provision of the value added service.

  The coin defined by the micropayment proposal carries a public common information that contains some time marks to limit when it is valid and when it is no longer valid. So, when the coin is no longer valid (after refund time mark, if it applies), location data must be removed from the provider.

- Service providers must inform clearly and in a comprehensive way the customers, prior to obtaining their consent, about the type of location which will be processed, the identity of the controller, the purposes and the duration of processing and whether the data will be transmitted to a third party to provide the value added service.

  The common information field agreed by both involved parties and signed by the bank can be used to describe the features and terms and conditions of the provided service.

- Customers shall be given the possibility to withdraw their consent for the processing of location data at any time.

  Since the customer fully manages the service requests, simply stopping to make requests to the LBS provider is enough to not provide location data anymore to the provider. Moreover, the customer can wait to ask for a refund of her unused microcoupons in case she is no more interested on the services offered by the provider or she is no more agreed with the management of her location data.

- The provider should regularly remind the customer about the fact that her mobile device has been, will be or can be located.

  It must be a task for the service implementation applied to the customer side application.

3.7 Conclusions

In this Chapter we have presented *μEasyPay*, a micropayment solution suitable to be used to pay merchants for low value items. We have described the security model and the general scenario for micropayments, clarifying and formalizing some security requirements. Customers withdraw a coin with a pre-agreed number of microcoupons belonging it and by using these microcoupons, customers can pay for items to merchants. It is a lightweight scheme due to the fact that it does not use asymmetric cryptography during the payment phase and the costs related to the withdrawn can be amortized. Besides, we have presented two versions of the *μEasyPay* solution to apply on slightly different services depending on the their special requirements. The first one is more efficient but parties have to assume a controlled and limited risk that consists on the risk of losing a single microcoupon. Instead, the second version avoids this limited risk. All security properties will be analyzed in §7.1 and performance evaluation will be provided in Chapter 10.

Moreover, we have studied the applicability of *μEasyPay* to concrete service, such as LBSs. We have proved that our scheme fits not only the functional and privacy requirements but it also accomplishes the legal requirements stated by the European legal framework in this specific field.

CHAPTER 4

$\mathcal{MC}-2\mathcal{D}$: ELECTRONIC MULTICOUPONS WITH MULTI-MERCHANT SCENARIO

Typically, a paper-based coupon is a document that allows customers to obtain goods or services from a merchant, often achieving discounts or gifts. Paper-based coupons have a good commercial adoption, since they are used in several scenarios. However, the electronic version of them are not yet widespread although there are some interesting proposals in the literature. In this Chapter, we have reviewed the previous proposals in this field and we have detected some issues in them that probability reduces the trust on their use by both merchants and customers. Despite there are two types of electronic coupons scenarios (i.e. single-merchant and multi-merchant), we focus our efforts in the multi-merchant scenario. There is only a previous scheme that deals with the multi-merchant scenario although it has some security and functional issues. Therefore, based on some of the knowledge acquired in the previous Chapter, we design a new electronic multicoupon proposal for the multi-merchant scenario that enhances the security as well as the efficiency of that previous scheme. Moreover, our scheme has been designed keeping in mind that it should be used on current mobile devices, such as smartphones, tablets and so on.

4.1 Introduction

A paper-based coupon is a document that allows a customer to obtain goods or services from a merchant, typically achieving discounts or gifts. Coupons are working properly in conventional commerce, because customers obtain benefits of using them, and so do merchants, increasing sales. We can find some successful examples: booklet for restaurants [16, 17], hotels [18], etc. So, it seems interesting to provide an electronic version of paper-based coupons to add some advantages: save paper, process improvement, use of mobile devices such as smartphones, etc. The true reality is that electronic coupons topic is receiving a remarkable attention in recent years [19–26], but we have not found real experiences (unlike the case of electronic tickets). We can surmise two reasons for this situation. On one hand, we can mention lack of trust for merchants and customers, and lack of privacy for customers. Thus, we need schemes accomplishing some security requirements in order to provide trust and privacy. On the other hand, we need viable and efficient solutions that can be deployed on mobile devices (smart-phones, tablets, PDAs, etc.).

There are paper-based coupons with different features: single coupons or booklet of coupons, anonymous or identified, for one merchant or multiple merchants, transferable or not, etc. In this Chapter we want to provide a scheme with the following initial conditions: multicoupon, multi-merchant, with a high degree of anonymity for customers and not detachable (unsplittable). In this context, and due to our solution is multi-merchant, we must provide more flexibility in this kind of scenario, so it is interesting to allow that a merchant can affiliate or disaffiliate to/from a group of merchants dynamically, and without causing a security problem for other merchants and customers. Being this way, we have decided that a specific entity will be responsible of issuing electronic coupons for a set of merchants. Therefore, we have, at least, three kinds of actors: customers, merchants and the issuer.

We want to provide a multicoupon scheme because some merchants prefer that customers buy a set of coupons (better than single coupons) to establish long-term relationships with them. This way, customers are encouraged to spend all the coupons of the set. Even, sometimes, a deadline is established to obtain the benefits of the coupons. In order to increase flexibility we foresee two possible situations: electronic coupons that can be refunded before a deadline, and electronic coupons without refund (with or without deadline to spend them). The refund process is a new procedure that cannot be found in previous solutions. Some other processes are involved in the use of coupons (this kind of processes can be found in all previous solutions). The claim process is necessary when the issuer and the merchant are not the same entity (we only find this process in [25]). The redeem process can be found in all previous solutions but only one previous solution allows a multi-redeem process [22], even it is an useful process for efficiency purposes, as we will explain in our proposal.

Some merchants want that multicoupons meet the unsplitability requirement. It is to say, a customer cannot share electronic coupons from a booklet with another customer. In order to establish a long-term relationship with customers, merchants wish their multicoupons to be unsplittable [22]. This way a multicoupon cannot be shared among different customers. If a multicoupon is unsplittable, the customer will not be able to transfer coupons to another customer, or in the case she is allowed to do it then she must give all her coupons to the receiver. The unsplittability requirement can be classified as full or weak [23]. On one hand, in weak unsplittability (or all-or-nothing sharing) a customer must give the whole multicoupon to the receiver and she risks losing every remaining coupon on it. On the other hand, full unsplittability implies that the customer who shares a coupon with another customer cannot use any other coupon until the second customer provides some data to the first one.

So far we have highlighted merchant desires, but we also have to deal with customer desires, and privacy is especially important. Users want to maintain the same degree of privacy in the electronic version as in the paper version. So, confidentiality of transactions, unlinkability of uses, and anonymity have to be achieved. Moreover, we do not forget some common security requirements that our solution has to meet, and that all previous solutions satisfy, such as unforgeability and coupon reuse avoidance.

Thus, we propose in this Chapter a complete scheme for multicoupon (that we call $\mathscr{MC}$-2$\mathscr{D}$) to be used in a multi-merchant scenario. Customers can spend different coupons of the multicoupon at different merchants affiliated to an issuer. The proposal builds unlikable and anonymous multicoupons, since nobody can determine who spends a coupon from a specific multicoupon and where these coupons are being spent. Therefore, a high degree of privacy is achieved for honest customers. Basic security requirements for merchants and issuers are met. Merchants are protected in front of possible customer misbehavior in such a way that whether a customer tries to reuse or to forge a coupon (at the same or different merchants), her anonymity will be revoked. This measure is the key point to discourage customers from detaching an electronic coupon from a booklet. Similarly, the scheme provides measures against dishonest behavior from merchants, allowing the revocation of the merchant affiliation. All of these security requirements will be proved by formal analysis afterwards in Chapter 7.

In this Chapter we prove by analytic efficiency comparison that our proposal enhances efficiency and flexibility in relation to the previous solution that deals with the multi-merchant scenario. This efficiency improvement comes from the fact that the scheme is based on lightweight cryptographic operations and allows to redeem and claim one or more coupons during the same run of the protocol. Meanwhile, the flexibility refers to the fact that the customer and the issuer can agree on the value and the number of coupons of each multicoupon, and merchants can join and leave

the affiliation without reducing the security of the multicoupon scheme. In addition to the complexity analysis, the proposed scheme will be compared by implementation to the previous solution [25] in Chapter 11. Furthermore, in Chapter 11 we will provide a full and practical implementation together with a full performance evaluation of the multicoupon scheme using commercial mobile devices deployed on a real production scenario.

The Chapter is organized as follows. In Section 4.2 we give an overview of the scenario considered in this Chapter, describing the security model and the considered security requirements of our multicoupon scheme. In Section 4.3 we review the related work. In Section 4.4 we outline how our proposal works and we define the structure and the life cycle of multicoupons. The full specification of protocols and data flows of our proposal are explained in Section 4.5. Section 4.6 compares the efficiency of our multicoupon scheme in an analytic way, proving that our scheme outperforms the previous proposal. Finally, in Section 4.7 we conclude the Chapter. As stated before, we leave the security analysis for the Chapter 7 and the practical implementation and the corresponding performance analysis in Chapter 11.

4.2 Overview: Scenarios and Security Model

In this Section, we present the multicoupon scenario, the entities involved and their role. Moreover, we describe the security model along with the definition of the considered security requirements.

4.2.1 The Multicoupon Scenario

As defined in the introduction Section, we consider three entities: issuer, merchants and customers. So, based on the involved entities and the relationship among them, we define two types of scenarios that we have identified in the previous proposals [20–25]. This will help to clarify the differences between our proposal and the existing ones, and to understand the need for new security measures.

- *Single-merchant*. This scenario involves many customers and a single merchant. That is, one merchant is in charge of the issuance, distribution and validation of multicoupons for the services that he offers to customers. Therefore, a customer can only use coupons at the merchant who issued them [20–24]. In this case, the issuer and the merchant are the same entity, so security is controlled by the merchant since he is responsible for issuing and validating coupons. Thus, he has all the information about the list of issued and already used coupons.

- *Multi-merchant*. As in the single-merchant scenario many customers can use multicoupons at one merchant, but in this case, in contrast to the single-merchant scenario, a customer can also use the same multicoupon at different merchants [25]. In this context, a merchant can accept coupons issued by another entity (another merchant or an issuer). Therefore, the security of the scheme must take into account that the merchant validating the multicoupon must have enough and valid information to verify the coupons presented by customers. This scenario allows attracting more customers to merchants. In addition, customers and merchants obtain greater benefits.

The scenario considered in this Chapter can be classified as a multi-merchant scenario where the issuer is the only entity in charge to issue multicoupons to customers, and merchants are the entities that accept and verify multicoupons. Moreover, a merchant could accept multicoupons issued by different issuers, but merchants should have an agreement with those issuers before accepting their multicoupons. In our scenario, a new entity called group manager, has been introduced to provide anonymity to the customers. This entity will be able to revoke the anonymity if it is required, as we will explain in §4.4. This new entity follows from a group signature scheme (see §2.3).

4.2.2 Security Model

In the rest of this Section we present a security framework based on the adversary model in which actions can be performed by the entities involved in the multicoupon scenario. We follow a security model description similar to other authors [22, 24, 25]. We begin by defining the algorithms and protocols available to both honest and adversarial entities. An adversary is a probabilistic polynomial time algorithm (hereinafter p.p.t. algorithm) $\mathscr{A}$, which can play the role of either customers or merchants, so $\mathscr{A}$ can interact with the other entities by means of a set of algorithms and protocols. Finally, and based on the later description, we define the security requirements.

***Definition* 4.2.1** *(Multicoupon Scheme). A multicoupon scheme ($\mathscr{MC}$-2$\mathscr{D}$) for an affiliation of merchants ($\mathscr{M} \in \{\mathscr{M}_0 \cdots \mathscr{M}_m\}$), a set of customers ($\mathscr{C} \in \{\mathscr{C}_0 \cdots \mathscr{C}_n\}$), an issuer ($\mathscr{I}$) and a group manager ($\mathscr{G}$), consists of a set of algorithms and protocols:* {`GMSetup`, `ISetup`, `Affiliation`, `Disaffiliation`, `GMRegistration`, `Issue`, `Multiredeem`, `Claim` *and* `Refund`}.

These algorithms and protocols are specified as follows:

- `GMSetup`. It is a probabilistic algorithm executed by $\mathcal{G}$ taking as inputs the security parameter (1^K) and the size of the group of users ($\mathcal{N}$). It outputs a group public key (pk^G) and up to $\mathcal{N}$ group member private keys $\left(\{sk^G_{\mathcal{C}_i}\}_{i=0}^{\mathcal{N}-1}\right)$.

- `ISetup`. It is a probabilistic algorithm executed by $\mathcal{I}$ taking as input the security parameter (1^K). It outputs an RSA key pair $(pk_{\mathcal{I}}, sk_{\mathcal{I}})$.

- `Affiliation`. It is a deterministic and interactive protocol performed between $\mathcal{I}$ and merchants interested to redeem multicoupons issued by $\mathcal{I}$, taking as input the merchant affiliation list ($\mathcal{AL}$). It outputs an agreement between $\mathcal{I}$ and each $\mathcal{M}$ that allows the latter the ability to accept and claim multicoupons to $\mathcal{I}$. In addition, the merchant identifier ($id_{\mathcal{M}}$) is included into the affiliation list ($\mathcal{AL}$).

- `Disaffiliation`. It can be either a non-interactive and deterministic algorithm or an interactive and deterministic protocol. As non-interactive algorithm, it is performed by $\mathcal{I}$ as a result of a merchant misbehavior. It outputs updated both the merchant revocation list ($id_{\mathcal{M}}$ is included on the affiliation revocation list ($\mathcal{ARL}$)) and the merchant affiliation list ($id_{\mathcal{M}}$ is deleted from the affiliation list ($\mathcal{AL}$)). As interactive protocol, it is performed between the $\mathcal{I}$ and a merchant wanting to leave the affiliation. In this case, it outputs the merchant affiliation list updated.

- `GMRegistration`. It is a probabilistic and interactive protocol between $\mathcal{G}$ and customers taking as inputs the identity of $\mathcal{C}$ and returning to $\mathcal{C}$ a group key pair $(pk^G, sk^G_{\mathcal{C}})$. As a result, $\mathcal{G}$ obtains the link between $\mathcal{C}$'s identity and her group member private key. If something fails, $\mathcal{G}$ outputs an error message.

- `Issue`. It is a probabilistic and interactive protocol between $\mathcal{I}$ and $\mathcal{C}$. $\mathcal{I}$ takes as inputs its RSA private key $(sk_{\mathcal{I}})$ and $\mathcal{C}$ uses as input the $\mathcal{I}$'s RSA public key $(pk_{\mathcal{I}})$. $\mathcal{I}$ and $\mathcal{C}$ agree on some common information containing the value and the number of coupons within a multicoupon, together with some timestamps (τ) to limit when each algorithm can be executed. At the end of the protocol, $\mathcal{C}$ obtains a multicoupon, denoted as $\mathbb{MC}^{2D}$, whose identifier is hidden to $\mathcal{I}$ due to the use of a partially blind signature scheme. If something fails, $\mathcal{I}$ outputs an error message. The issuer outputs its view of the protocol $\left(V_{\mathcal{I}}^{\texttt{Issue}}\right)$.

- `Multiredeem`. It is a probabilistic and interactive protocol executed between $\mathcal{C}$ and $\mathcal{M}$. $\mathcal{C}$ takes as inputs her $\mathbb{MC}^{2D}$ and her group key pair $(pk^G, sk^G_{\mathcal{C}})$. $\mathcal{M}$ uses as inputs his private key $(sk_{\mathcal{M}})$ and the group public key (pk^G). Then, $\mathcal{C}$ releases a list of coupons $(\mathcal{L}_R)$ group signed using her group key pair in exchange of the requested good or service. Thus, at the end of the algorithm, $\mathcal{M}$

obtains a list ($\mathscr{L}_R$) of coupons from $\mathscr{C}$. If something fails, $\mathscr{M}$ outputs an error message. The merchant outputs its view of the protocol ($\mathcal{V}_{\mathscr{M}}^{\texttt{Multiredeem}}$).

- `Claim`. It is a probabilistic and interactive protocol between $\mathscr{M}$ and $\mathscr{I}$. $\mathscr{M}$ uses as inputs the list of received coupons from customers ($\mathscr{L}_R$) and its private key ($sk_{\mathscr{M}}$). $\mathscr{I}$ takes as inputs his public key ($pk_{\mathscr{I}}$) and the list of previously received coupons from affiliated merchants. At the end of the protocol, $\mathscr{I}$ allows $\mathscr{M}$ to deposit the value of the claimed coupons ($\mathscr{L}_R$) or outputs an error message if something fails. The issuer outputs its view of the protocol ($\mathcal{V}_{\mathscr{I}}^{\texttt{Claim}}$).

- `Refund`. It is a probabilistic and interactive protocol between $\mathscr{C}$ and $\mathscr{I}$. $\mathscr{C}$ uses the list of coupons not yet redeemed ($\mathscr{L}_{\overline{R}}$) and her private key ($sk_{\mathscr{C}}$). As in the `Claim` algorithm, $\mathscr{I}$ takes the same parameters as input. At the end of the protocol, $\mathscr{C}$ obtains a refund of her unused and not yet redeemed coupons or an error message if something fails. The issuer outputs its view of the protocol ($\mathcal{V}_{\mathscr{I}}^{\texttt{Refund}}$).

***Definition* 4.2.2** *(Correctness). If an honest customer $\mathscr{C}$ runs the* `Issue` *protocol with an honest $\mathscr{I}$, then $\mathscr{C}$ obtains a signed and properly issued multicoupon from $\mathscr{I}$. If an honest $\mathscr{C}$ executes the* `Multiredeem` *protocol with an honest $\mathscr{M}$ using some valid coupons from a valid multicoupon, then $\mathscr{M}$ always accepts them. If an honest $\mathscr{M}$ runs the* `Claim` *protocol with an honest $\mathscr{I}$ using a list of coupons from some valid multicoupons received from honest customers, then $\mathscr{I}$ always accept this list if all the contained coupons are valid.*

Note that a valid coupon is a coupon properly obtained from a valid `Issue` protocol and it has not been previously redeemed.

Below, all security properties required by the security model (and previously described in §1.3 as general definitions for privacy-protecting solutions) will be formally described in detail: unforgeability, unlinkability, anonymity of customers, coupon reuse avoidance, anonymity revocation of misbehaving customers, disaffiliation of merchants and unsplittability protection.

4.2.3 Unforgeability

There is an intrinsic monetary value associated to any coupon, either explicitly or implicitly. Therefore, the system needs to be unforgeable.

From the point of view of both issuer and merchants, no coalition of customers should be able: to redeem more coupons than they have been rightfully issued, to self-issue new multicoupons or to modify their content. In this case, let an adversary

$\mathcal{A}_0$ be a p.p.t. Turing Machine acting as a malicious customer or a coalition of them. $\mathcal{A}_0$ has all the public parameters needed to operate in the system, so:

- $\mathcal{A}_0$ can execute the `Issue` protocol with an honest issuer.
- $\mathcal{A}_0$ can execute the `Multiredeem` protocol with an honest merchant.

Let us consider a game 0, where $\mathcal{A}_0$ can legitimately obtain a multicoupon ($\mathbb{MC}^{2D}$) composed by a set of valid single coupons. Then, $\mathcal{A}_0$ can execute the `Multiredeem` algorithm with an honest merchant using a coupon $\mathbf{c} \notin \mathbb{MC}^{2D}$. Therefore, it is required that for an adversary $\mathcal{A}_0$ playing the game 0, the probability an honest merchant accepts the `Multiredeem` protocol is negligible.

On the another hand, the issuer is also interested on that no coalition of merchants should be able to claim more coupons than they had been received by customers. In this case, let an adversary $\mathcal{A}_1$ be a p.p.t. Turing Machine acting as a malicious merchant or a coalition of them. $\mathcal{A}_1$ has all the public parameters needed to participate in the system, so:

- $\mathcal{A}_1$ can execute the `Multiredeem` protocol with honest customers.
- $\mathcal{A}_1$ can execute the `Claim` protocol with an honest issuer.

Let us consider a game 1, where $\mathcal{A}_1$ receives a list of coupons $\mathcal{L}_R$ from honest customers. Let us define some counters: $\mathcal{I}_R$ as the number of coupons redeemed by honest customers; $\mathcal{I}_C$ as the number of coupons claimed by the adversary. Then, $\mathcal{A}_1$ can execute the `Claim` protocol transferring $\mathcal{I}_C$ coupons that is greater than $\mathcal{I}_R$, i.e. $\mathcal{I}_C > \mathcal{I}_R$. Then, it is required that the probability for an adversary $\mathcal{A}_1$, playing game 1, to claim more coupons than the number of coupons redeemed by honest customers during the `Multiredeem` algorithm, is negligible.

Note that, $\mathcal{I}$ may issue as many coupons as he wants, and hence, unforgeability with a corrupted issuer would make no sense.

Formaly, the unforgeability property is defined as follows:

***Definition* 4.2.3** *(Unforgeability). A multicoupon scheme is unforgeable if there is no p.p.t. adversary $\mathcal{A}(\mathcal{A}_0, \mathcal{A}_1)$ that can win any unforgeability game with non-negligible probability (in* `K`*).*

4.2.4 Unlinkability

Customers want to protect their operations in the system in a sense that it should be unfeasible for a merchant or a coalition of them to link a redeem procedure to the corresponding issue procedure where the multicoupon had been issued, or to

link two different redeem procedures from the same customer using different $\mathbb{MC}^{2D}$ multicoupons. That is, it tries to know whether two procedures have been made by a same customer no matters which is her identity [24].

On one hand, let an adversary $\mathcal{A}_0$ be a p.p.t. Turing Machine acting as a malicious merchant or a coalition of them. Assume that $\mathcal{C}_0$ (who holds $\mathbb{MC}_0^{2D}$) and $\mathcal{C}_1$ (who holds $\mathbb{MC}_1^{2D}$) have already executed at least one `Multiredeem` with the multicoupons they hold. Consequently, $\mathcal{A}_0$ has one view for each `Multiredeem` protocol run, i.e. $\mathcal{V}_{\mathcal{A}_0}^{\texttt{Multiredeem}_0}$ and $\mathcal{V}_{\mathcal{A}_0}^{\texttt{Multiredeem}_1}$. $\mathcal{A}_0$ has all the public parameters to be involved in the system, so:

- $\mathcal{A}_0$ can execute the `Multiredeem` protocol with honest customers.

Let us consider a game 0, where $\mathcal{A}_0$ outputs secretly and randomly a bit $b = \{0,1\}$. At some point, $\mathcal{A}_0$ executes a `Multiredeem` protocol with the honest customer b ($\mathcal{C}_b$) who owns the multicoupon used during the $\texttt{Multiredeem}_b$ and $\mathcal{A}_0$ outputs a bit $b' = \{0,1\}$. Then, we require that for every adversary playing the game, the advantage in the success probability that $\mathcal{C}_b$ and $\mathcal{C}_{b'}$ are the same customer, is at most negligible. In fact, we require that the probability for an adversary to win the game is not better than a random toss of the coin.

On the other hand, let an adversary $\mathcal{A}_1$ be a p.p.t. Turing Machine acting as a malicious merchant or a coalition of them. Moreover, suppose a corrupted issuer who provides $\mathcal{A}_1$ the complete view of a previously executed `Issue` protocol. $\mathcal{A}_1$ has all the public parameters to be involved in the system, so:

- $\mathcal{A}_1$ can execute the `Multiredeem` protocol with honest customers.

Let us consider a game 1, where $\mathcal{A}_1$ obtains two views from two `Issue` protocols, i.e. $\mathcal{V}_{\mathcal{J}}^{\texttt{Issue}_0}$ and $\mathcal{V}_{\mathcal{J}}^{\texttt{Issue}_1}$, from a corrupted issuer. Suddenly, $\mathcal{A}_1$ outputs a bit $b = \{0,1\}$. Then, $\mathcal{A}_1$ executes a `Multiredeem` protocol with the honest customer b who owns the multicoupon contained in $\mathcal{V}_{\mathcal{J}}^{\texttt{Issue}_b}$. Finally, $\mathcal{A}_1$ outputs another bit $b' = \{0,1\}$. As in the game 0, we require that for every adversary playing the game, the advantage in the success probability that $\mathcal{C}_b = \mathcal{C}_{b'}$ is at most negligible.

***Definition* 4.2.4** *(Unlinkability). The multicoupon scheme is $2\mathcal{D}$-unlinkable if there is no p.p.t. adversary $\mathcal{A}(\mathcal{A}_0, \mathcal{A}_1)$ that can win any unlinkability game with non-negligible linkability advantage over random guess (in* `K`*).*

4.2.5 Anonymity of Customers

Customers interested in using coupons do not want to reveal their identity to perform actions in the system. So, customers want to be anonymous, i.e. customers should

be able to obtain and redeem coupons without revealing any information about their real identity. Neither the issuer, who issues coupons to customers, nor merchants, who accept the coupons from customers, must not be able to obtain information concerning the identity of the customers.

Consider an adversary $\mathcal{A}_0$ as a p.p.t. Turing Machine playing a game 0 and acting as a malicious issuer. $\mathcal{A}$ has all the parameters needed to be involved in the system, so:

- $\mathcal{A}_0$ can execute the `Issue` protocol with honest customers.

Similarly, consider an adversary $\mathcal{A}_1$ playing a game 1 and acting as a malicious merchant and having all the parameters to be involved in the system, so:

- $\mathcal{A}_1$ can execute the `Multiredeem` protocol with honest customers.

At some point of game 0 (game 1, resp.), $\mathcal{A}_0$ ($\mathcal{A}_1$, resp.) tries to infer the real identity of the customer using all data contained in each protocol view (either $\mathcal{V}_{\mathcal{A}_0}^{\texttt{Issue}}$ or $\mathcal{V}_{\mathcal{A}_1}^{\texttt{Multiredeem}}$, resp.). Then, it is required that the probability for an adversary (playing any of the games) to obtain the real identity of a customer is negligible.

***Definition* 4.2.5** *(Anonymity). The multicoupon scheme is anonymous if there is no p.p.t. adversary $\mathcal{A}(\mathcal{A}_0,\mathcal{A}_1)$ that can win any anonymity game obtaining the identity of an honest customer when she is performing actions in the system (either issue a multicoupon from the issuer or redeem a coupon from a merchant) with non-negligible probability (in* `K`*).*

4.2.6 Coupon Reuse Avoidance

The system must prevent, or at least detect, that neither a customer has redeemed the same coupon more than once nor a merchant has claimed the same coupon more than once. It is clear that we cannot prevent entities making copies of coupons but we must assure that only one of them will be given as valid.

Formally, we can define an adversary $\mathcal{A}_0$ as a p.p.t. Turing Machine playing a game 0 with the role of a malicious customer or a coalition of them. $\mathcal{A}_0$ has all the parameters needed to be involved in the system, so:

- $\mathcal{A}_0$ can execute the `Issue` protocol with an honest issuer.
- $\mathcal{A}_0$ can execute the `Multiredeem` protocol with honest merchants.

At some point of the game 0, $\mathscr{A}_0$ tries to redeem the same coupon twice or more times with either the same merchant or different merchants, using the `Multiredeem` protocol.

Similarly, let define an adversary $\mathscr{A}_1$ who plays a game 1 with the role of a malicious merchant or a coalition of them. $\mathscr{A}_1$ has all the parameters needed to be involved in the system, so:

- $\mathscr{A}_1$ can execute the `Multiredeem` protocol with an honest customer.
- $\mathscr{A}_1$ can execute the `Claim` protocol with an honest issuer.

At some point of the game 1, $\mathscr{A}_1$ tries to claim the same coupon twice or more times to the issuer, using the `Claim` protocol.

Then, for each game we require that the probability for each adversary to redeem (or claim) a coupon already redeemed (or claimed) without the honest merchant (or the issuer) detecting and avoiding the reuse is negligible. Note that in every case, the system allows the anonymity revocation of the misbehaving party and the revocation of her identity (see Definition 4.2.7).

***Definition* 4.2.6** *(Coupon reuse avoidance). The multicoupon scheme allows detecting and avoiding coupon reuse if no p.p.t. adversary $\mathscr{A}(\mathscr{A}_0, \mathscr{A}_1)$ can win any coupon reuse game using two coupons, i.e. $\mathbf{c}$ and $\mathbf{c}'$ from the same multicoupon $\mathbb{MC}^{2D}$, where $\mathbf{c} = \mathbf{c}'$, without the verifier (either merchant or issuer) detecting it, with non-negligible probability (in $\mathtt{K}$).*

4.2.7 Anonymity Revocation of Misbehaving Customers

Although the anonymity of customers is desired, the scheme must provide a mechanism to reveal the identity of customers when they misbehave, i.e. when a customer tries either to use a fake coupon or to reuse a coupon already redeemed, her anonymity must be revoked. It should be also possible to revoke the anonymity of a malicious customer trying to unfairly blame an honest customer.

Let us consider an adversary $\mathscr{A}$ as a p.p.t. Turing Machine acting as a malicious customer, a coalition of them or probably also colluding with a malicious merchant. $\mathscr{A}$ has all the parameters required to operate in the system, so:

- $\mathscr{A}$ can execute the `Multiredeem` protocol with honest merchants.

At some point of the game, $\mathscr{A}$ tries to execute the `Multiredeem` protocol with an honest merchant using either a forged coupon (see Definition 4.2.3) or a coupon already used in a previous `Multiredeem` instance (see Definition 4.2.6). Then, through the use of an anonymity revocation mechanism, the anonymity of the

cheater should be revoked. Moreover her identification should be also correct even $\mathcal{A}$ tries to blame an honest customer (non-frameability). So, it is required that the probability for the adversary to be not identified by the revocation mechanism, after either a forging or a coupon reuse attempt, should be negligible.

***Definition* 4.2.7** *(Anonymity revocation of misbehaving customers). The multicoupon scheme allows the anonymity revocation and the correct identification of malicious customers if there is no p.p.t. adversary $\mathcal{A}$ that can win the anonymity revocation game with non-negligible probability (in* `K`*).*

4.2.8 Disaffiliation of Merchants

Every merchant who wants to accept and claim multicoupons issued by the issuer, must be affiliated to that issuer. Then, the issuer assigns a unique merchant identifier ($id_{\mathcal{M}}$) to each affiliated merchant, and adds this identifier to the affiliation list ($\mathcal{AL}$). Similarly to the affiliation, the system must provide a secure disaffiliation when merchants leave it. We define two types of disaffiliation: voluntarily or forced.

- *Voluntarily disaffiliation*. An honest merchant decides on his own, by some reasons, that he does not want to belong the affiliation anymore. Issuer removes merchant from the affiliation list.

- *Forced disaffiliation*. The scheme must provide measures against dishonest behavior of merchants. For example, assuming an honest customer, a merchant could misbehave similarly to a customer, that is, using a fake coupon or reusing a coupon received from a customer. When a merchant misbehaves, the issuer should be able to revoke the affiliation of this dishonest merchant adding him to the affiliation revocation list ($\mathcal{ARL}$) and also removing him from the affiliation list ($\mathcal{AL}$).

In both cases, when a merchant leaves the system, i.e. he leaves the affiliation with the issuer, the security of the multicoupon scheme must not be compromised. So, the issuer and merchants cannot share sensitive information about customers. Moreover, once a merchant has left the affiliation, he can no longer obtain benefits from the issuer, for example, changing coupons for money.

We can define an adversary $\mathcal{A}$ who plays a game with the role of a malicious merchant (merchant not affiliated to the issuer) or a coalition of them. $\mathcal{A}$ has all the parameters needed to be involved in the system, so:

- $\mathcal{A}$ can execute the `Multiredeem` protocol with customers.

- $\mathcal{A}$ can execute the `Claim` protocol with an honest issuer.

Let us consider a game, where $\mathscr{A}$ can execute the `Multiredeem` algorithm with customers obtaining a set of valid coupons. Later, $\mathscr{A}$ tries to claim these valid coupons using the `Claim` protocol. Then, we require that the probability for a not affiliated adversary ($\mathscr{A} \notin \mathscr{AL}$) to claim a coupon without the honest issuer detecting and avoiding it is negligible.

***Definition* 4.2.8** *(Disaffiliation of merchants). The multicoupon scheme is secure against merchants disaffiliation either forced by a proved misbehavior or voluntarily by the merchant decision, if there is no p.p.t. adversary $\mathscr{A}$ that can win the disaffiliation game, obtaining coupons from honest customers and then achieving the associated benefits of these coupons from the issuer, with non-negligible probability (in* `K`*).*

4.2.9 Unsplittability Protection

We can find two definitions about unsplittability. On one hand, *weak unsplittability*, as defined in [23], requires that whether a customer wants to share some coupons of a multicoupon with another entity, she must provide secret information related to that multicoupon. Thus, it is designed to discourage sharing. On the other hand, *strong unsplittability* requires that whether a customer gives a single coupon to another customer, the first customer cannot use any other coupon until the second customer provides some information to the first one [25]. In both cases (*weak* and *strong unsplittability*) customers sharing coupons must trust each other.

In our scheme, we take into account *weak unsplittability*. In this case, let an adversary $\mathscr{A}_0$ be a p.p.t. Turing Machine acting as a customer or a coalition of customers. $\mathscr{A}_0$ can share a set of coupons from her legitimately issued multicoupon $\mathbb{MC}^{2D}$, so:

- $\mathscr{A}_0$ can execute the `Issue` protocol with an honest issuer.
- $\mathscr{A}_0$ can execute the `Multiredeem` protocol with honest merchants.

Let us consider a game 0, where $\mathscr{A}_0$ obtains a valid multicoupon ($\mathbb{MC}^{2D}$) from the issuer. Then, $\mathscr{A}_0$ extracts a list $\mathscr{L}_s$ of valid coupons from a legitimately issued multicoupon $\mathbb{MC}^{2D}$, and gives $\mathscr{L}_s$ to an honest customer $\mathscr{C}_0$. Then, $\mathscr{A}_0$ can execute the `Multiredeem` algorithm with an honest merchant using a coupon $\mathbf{c} \in \mathscr{L}_s$.

Hence, for an adversary $\mathscr{A}_0$ playing the game 0, the probability that $\mathscr{C}_0$ can be falsely accused of coupon reuse is non-negligible.

Similarly, consider an adversary $\mathscr{A}_1$ be a p.p.t. Turing Machine acting as a customer or a coalition of customers. $\mathscr{A}_1$ can receive a set of coupons from another customer $\mathscr{C}_1$ who has a legitimately issued multicoupon $\mathbb{MC}^{2D}$, so:

- $\mathcal{A}_1$ can execute a `Multiredeem` protocol with honest merchants.

Let us consider a game 1, where $\mathcal{A}_1$ receives a list $\mathcal{L}_s$ of valid coupons $\mathbf{c}$ from $\mathcal{C}_1$. Then, $\mathcal{A}_1$ can execute the `Multiredeem` algorithm with an honest merchant using a coupon $\mathbf{c} \notin \mathcal{L}_s$, but $\mathbf{c} \in \mathbb{MC}^{2D}$.

Therefore, for an adversary $\mathcal{A}_1$ playing the game 1, the probability that $\mathcal{A}_1$ could cause $\mathcal{C}_1$ to be falsely accused of coupon reuse is non-negligible.

Protection against splitting was defined in [20] and we have adapted it to our scheme as follows:

***Definition* 4.2.9** *(Unsplittability Protection). Customers are discouraged to either split and share an N_j-redeemable multicoupon into (disjount) s_i-redeemable shares with $\sum_i s_i \leq N_j$, or receive a s_i-redeemable shares, without customers trust each other, since an honest customer is at risk of being unfairly accused of coupon reuse (with non-negligible probability in* `K`*).*

4.3 Related Work

As we already pointed out in the introduction, there are several proposals to provide security to multicoupons. We analyze these proposals considering two main aspects of multicoupon schemes: functionalities and security properties. Table 4.1 summarizes the set of features and properties of all the reviewed proposals.

Regarding to functionalities, all the analyzed solutions [20–25] provide the basic protocols required for operating with multicoupons (issue and redeem) in single-merchant scenarios. However, further procedures are needed for more general scenarios, such as claim and refund. On one hand, the claim protocol is required when the entity who issues coupons and the entity who exchanges them with the customer for goods or services are not the same entity. This is the case for a multi-merchant scenario like the proposed in [25]. On the other hand, a refund protocol allows customers to recover from the issuer the value of a list of already issued coupons but not yet used.

Even though the redeem process is supported by all the proposals, solutions for multicoupons should provide mechanisms to allow redeeming more than one single coupon within the same redeem process. This process is called multi-redemption and it is an interesting process for flexibility and efficiency purposes. However, the vast majority of analyzed solutions require executing the redeem process as many times as individual coupons are provided. To the best of our knowledge, this process is only supported by [22, 95].

As mentioned in [20, 95] a privacy protecting coupon system should at least provide the property of *weak unsplittability*. The solutions presented in [20, 21] allow

Table 4.1: Multicoupons solutions - A comparative analysis

	Chen et al. [20]	Nguyen [21]	Canard et al. [22]	Escalante et al. [23]	Chen et al. [24]	Armknecht et al. [25]	$\mathcal{MC}$-2$\mathcal{D}$
Scenario	SM	SM	SM	SM	SM	MM	MM
Issue	✓	✓	✓	✓	✓	✓	✓
Redeem	single	single	multi	single	single	single	multi
Claim	-	-	-	-	-	✓	✓
Refund	-	-	-	-	-	-	optional
Unsplitting	weak	weak	-	strong	strong	strong	weak
Unforgeability	✓	✓	✓	✓	✓	✓	✓
Coupon reuse avoidance	✓	✓	✓	✓	✓	✓	✓
Unlinkability	✓	✓	✓	✓	✓	✓	✓
Customer anonymity	✓	✓	✓	✓	✓	✓	✓
Confidentiality	-	-	-	-	-	-	✓
Merchant Disaffiliation	-	-	-	-	-	-	✓
Revocation of Customer's Anonymity	-	-	-	-	-	-	✓
Revocation of Merchant's Affiliation	-	-	-	-	-	-	✓

✓YES , -NO
SM- single-merchant
MM- multi-merchant

weak unsplittability while the schemes in [23–25] obtain *strong unsplittability*. Instead, the proposal in [22] provides customer with the possibility to detach a coupon from a multicoupon and transfer it to another customer, but in this case, the issuer entity must be involved in the transfer process.

Concerning security, multicoupon solutions should consider the customer privacy, and the detection and prevention of fraudulent use of multicoupons. Almost all the analyzed schemes accomplish with these requirements [20–25], but although these schemes deal with customer anonymity, they do not take into account the possibility to revoke the anonymity of the customer when she makes a fraudulent use of coupons, or to provide confidentiality of the data exchange between the two edges of the communication. However, the main drawback presented by most of the multicoupon schemes is that they are designed for *single-merchant* scenarios [20–24], so security is controlled by the merchant, since he is responsible for issuing and validating coupons. But, these kind of schemes reduce the use scope of multicoupons because customers can only interact with the merchant who issued the multicoupons.

The above limitation was partially solved by [25] allowing a merchant federation and hence a *multi-merchant* scenario. In that scheme, customers obtain multicoupons from any merchant within the same federation and customers can spend the coupons at any federated merchant. The federation is an association of merchants where all merchants share a key pair (public and private), and each merchant has a different private key to sign coupons. However, the merchant who receives a coupon, previously issued by another merchant, must find and contact with the original issuer in order to recover the applied discount. Moreover, the merchants must share the same federation private key and a common database where data about issued and already used coupons must be updated by merchants. So, when a merchant leaves the federation, the shared private key and the shared data can be compromised, opening a serious security problem. In addition, the multicoupon scheme should provide measures to expel dishonest merchants from the federation. This is a critical security issue unresolved by [25].

Therefore, new mechanisms are needed to obtain a multicoupon scheme more secure and efficient in a *multi-merchant* scenario.

4.4 $\mathscr{MC}$-2$\mathscr{D}$: A Multicoupon Scheme

In this Section we present a multicoupon scheme, that we call $\mathscr{MC}$-2$\mathscr{D}$. This scheme accomplishes all the security requirements for a *multi-merchant* scenario (see §4.2.2) in which merchants are affiliated to an issuer entity. Therefore, a merchant neither needs a prior relationship with other affiliated merchants nor share any information, unlike the merchants federation proposed in [25] in which merchants must trust each other.

4.4.1 Our Proposal in a Nutshell

As already explained in §4.2.2, the involved entities in our scenario are: issuer ($\mathscr{I}$), merchants ($\mathscr{M}$), customers ($\mathscr{C}$) and group manager ($\mathscr{G}$). The latter is a trusted third party that comes from the group signature primitive, explained in Chapter 2 (see §2.3).

Figure 4.1 sketches the protocol flow and the interactions among all the involved parties. As an initial assumption, we consider the use of a secure channel between the two edges of the communication in order to provide data confidentiality to the involved parties. To initialize, $\mathscr{G}$ and $\mathscr{I}$ must execute a setup procedure (`GMSetup` and `ISetup` algorithms, resp.) in order to receive requests from the other parties.

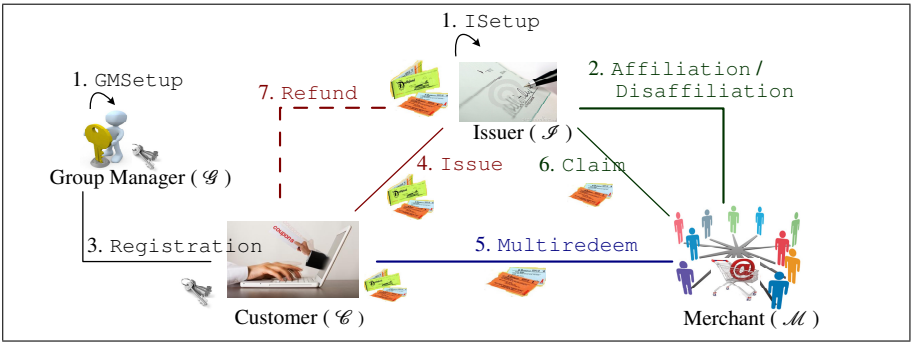


Figure 4.1: Our proposed scenario. Entities relationship and protocol flow.

On one hand, each $\mathscr{M}$ interested in accepting multicoupons issued by $\mathscr{I}$ must affiliate to $\mathscr{I}$ by means of the `Affiliation` protocol. This is a simple step because the only thing they do is a contract agreement. No sensitive data is shared among $\mathscr{I}$ and merchants, so they can join and leave (using the `Disaffiliation` protocol) the affiliation without any security trouble neither for customers nor for $\mathscr{I}$.

On the other hand, $\mathscr{C}$ should register to $\mathscr{G}$ running the `GMRegistration` protocol. This way, $\mathscr{C}$ receives a group key pair which will be used by her to hide her real identity when she acquires goods or services (henceforth services) at merchants. At this point, $\mathscr{G}$ links a group key pair with $\mathscr{C}$'s identity, allowing further anonymity revocation if she misbehaves.

Since this moment, $\mathscr{C}$ can request to $\mathscr{I}$ a multicoupon (`Issue` protocol). After the issuing process, $\mathscr{C}$ is ready to (multi-)redeem a set of coupons from the received multicoupon at any $\mathscr{M}$ in exchange of a service (`Multiredeem` protocol). When $\mathscr{M}$ has a list of received coupons from customers, he can exchange it for a money transfer to his account balance (`Claim` protocol). It is important to note that the last three protocols allow sending several coupons using a single transaction, so it increases the protocol efficiency, as we will prove in §4.6 by means of a complexity

analysis. Then, in Chapter 11 we will emphasize this fact by practical implementation and full performance analysis using mobile devices.

Finally, a `Refund` protocol has been designed to allow $\mathscr{C}$ to request $\mathscr{I}$ a reimbursement of her unused coupons whether she is no more interested in the use of her multicoupon. This procedure is designed to be optional and its application will depend on the agreement of the involved parties and the offered service.

In Figure 4.2 we define some time values used to manage the coupons life cycle in order to limit when each party can execute each protocol. This definition is also useful from the point of view of the efficiency because when a multicoupon is not usable anymore, parties can erase the related data. These time values are included inside the multicoupon together with other parameters that will be explained below.

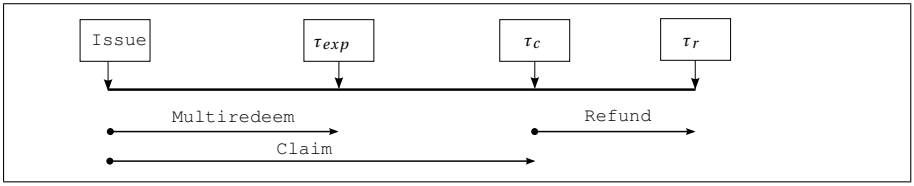


Figure 4.2: Life cycle of multicoupon.

4.4.2 The Structure of a Multicoupon

Our scheme is designed around a general multicoupon structure, called $\mathbb{MC}^{2D}$ (see Figure 4.3), which is made up of two elements (similarly to the structure proposed in Chapter 3):

1. $\mathbb{MC}_{\omega}$: the set of multicoupons generated applying hash chain operations.
2. $\mathbb{MC}_{\mathscr{PBS}}$: the partially blind signature over the list of identifiers of each multicoupon.

The $\mathbb{MC}_{\omega}$ element holds a number J of multicoupons, each one of them containing a chained list of single coupons, denoted as $\mathbf{c}_j(m)$, where m $(1 \leq m \leq N_j)$ is the coupon position within a multicoupon j, and N_j is the number of coupons of the multicoupon j. These multicoupons are obtained applying the hash chain procedure (see §2.1) over a random seed identifier $\omega_{N_j,j}$ (the first element of the hash chain) up to the final element of the chain which is called the multicoupon identifier $(\omega_{0,j})$. Figure 4.3 shows each coupon contains two fields $(\mathbf{c}_j(m) = [\omega_{i+1,j}, \omega_{i,j}])$: proof information $(\omega_{i+1,j})$ and payment information $(\omega_{i,j})$. These two fields are related by a single hash operation $\left(\mathscr{H}\left(\omega_{i+1,j}\right) = \omega_{i,j}\right)$, so that the proof information piece proves that the previous payment information belongs to the same multicoupon from an issued $\mathbb{MC}^{2D}$, thanks to the use of hash chain properties.

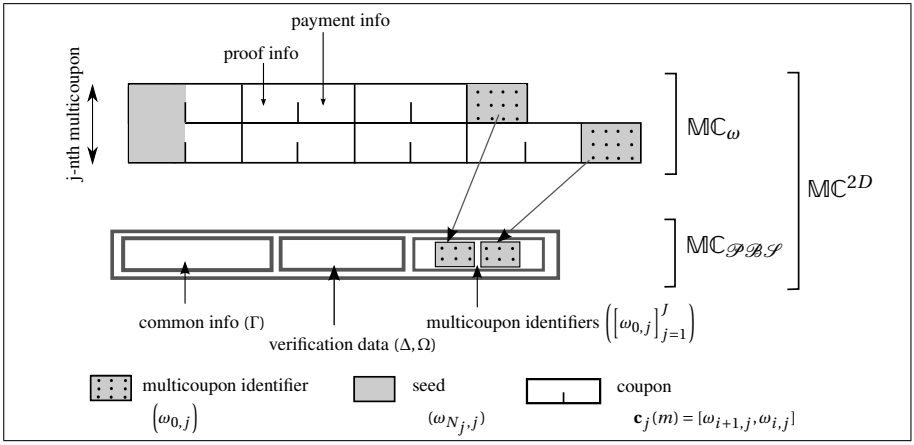


Figure 4.3: $\mathbb{MC}^{2D}$ structure composed by $\mathbb{MC}_{\omega}$ and $\mathbb{MC}_{\mathcal{PBS}}$.

The $\mathbb{MC}_{\mathcal{PBS}}$ element is generated by the cooperation of both $\mathscr{C}$ and $\mathscr{I}$ when the $\mathbb{MC}^{2D}$ is issued. This element will be used by $\mathscr{C}$ to prove that she has a $\mathbb{MC}^{2D}$ properly issued by $\mathscr{I}$. The partially blind signature contains three fields: the common information (Γ), the verification data (Δ, Ω) and the multicoupon identifiers list from each multicoupon $\left(\left[\omega_{0,j}\right]_{j=1}^{J}\right)$.

First, the common information field $\Gamma = \left(\mathscr{I}_{id}, s_{id}, \left[\left(N_j, v_j\right)\right]_{j=1}^{J}, \tau_{exp}, \tau_c, \tau_r\right)$ is an array of public data previously published by $\mathscr{I}$, which describes each of the offered services. There are some mandatory parameters and some of them are optional depending on the provided service and its features. The list of parameters is as follows:

- $\mathscr{I}_{id}$. An optional identification of the $\mathscr{I}$ who issues the $\mathbb{MC}^{2D}$.
- s_{id}. An optional identifier about the target service.
- N_j. It is the number of coupons within a single multicoupon j.
- $\left[\left(N_j, v_j\right)\right]_{j=1}^{J}$. This matrix contains J rows where each of them specifies the number of coupons and their value for each multicoupon j, where $1 \leq j \leq J$. Note that all coupons inside each multicoupon j have the same value v_j, but coupons from another multicoupon could have different value.
- τ_{exp}. It is the time up to $\mathscr{C}$ can redeem her coupons to merchants.
- τ_c. It is the time up to $\mathscr{M}$ can claim to $\mathscr{I}$ the received coupons from customers.

- τ_r. If merchants and $\mathcal{I}$ agree on providing the refund feature, this parameter indicates the time up to $\mathcal{C}$ can ask $\mathcal{I}$ for a refund of her unused coupons. After this time mark, the $\mathbb{MC}^{2D}$ will no longer valid.

Secondly, the verification field (Δ, Ω) contains data generated during the partially blind signing process which will be used to verify the signature. Finally, the last field contains the list of multicoupons identifiers $\left([\omega_{0,j}]_{j=1}^{J}\right)$. This data is hidden to $\mathcal{I}$ when the $\mathbb{MC}_{\mathcal{PBS}}$ is issued by means of the use of a partially blind signature scheme.

4.5 The Full Specification of the Multicoupon Proposal

Now we are going to define all the algorithms and protocols in detail, according to the Definition §4.2.1. Table 4.2 shows the notation used along the protocol description.

Table 4.2: Notation used in the protocol description.

Element	Description
$\mathcal{H}(x)$	One-way collision resistant hash function applied on x
$\mathcal{H}^i(x)$	$\mathcal{H}$ function applied i times iteratively on x
$\mathbb{S}_{\mathcal{P}}(x)$	$\mathcal{P}$'s signature on the element x
$x \overset{R}{\leftarrow} \mathbb{Z}_r$ and $x \overset{R}{\leftarrow} \mathbb{Z}_n^*$	Element x randomly chosen from either $\mathbb{Z}_r$ or $\mathbb{Z}_n^*$ sets
$\mathbb{MC}^{2D}$	The structure of a multicoupon containing two elements $[\mathbb{MC}_\omega, \mathbb{MC}_{\mathcal{PBS}}]$
$\mathbb{MC}_\omega$	Set of multicoupons containing a chained list of single coupons $\mathbf{c}_j(m)$
$\mathbb{MC}_{\mathcal{PBS}}$	Partially blind signature on multicoupon identifiers of $\mathbb{MC}_\omega$
$\mathbf{c}_j(m) = [\omega_{i+1,j}, \omega_{i,j}]$	A single coupon: proof information ($\omega_{i+1,j}$) and payment data ($\omega_{i,j}$)
$\mathbb{SG}_{\mathcal{C}}(x)$	Group signature generated by $\mathcal{C}$ on x using $Sign_{\mathcal{C}}^{G}$ (see §2.3)
$\omega_{0,j} \Vert_{j=1}^{J}$	Concatenation of J multicoupon identifiers in $\mathbb{MC}_\omega$
(e, n)	RSA public key of $\mathcal{I}$
(d, p, q)	RSA private key of $\mathcal{I}$
$id_{\mathcal{M}}$	Merchant's unique identifier in the affiliation
id_r	Unique identifier for a `Multiredeem` transaction

4.5.1 System Setup

The system setup is divided into two parts: $\mathcal{G}$ key deployment (`GMSetup` algorithm) and $\mathcal{I}$ setup (`ISetup` algorithm).

- *$\mathcal{G}$ key deployment*. During the algorithm `GMSetup`, $\mathcal{G}$ executes the procedure $KeyGen^G$ from the group signature scheme, as explained in §2.3, and outputs a

group public key (pk^G), a group manager private key ($sk^G_{\mathscr{G}}$) and a predefined number (n) of user private keys $\left(sk^G_1, sk^G_2, ..., sk^G_n\right)$ according to the defined group size.

- $\mathscr{I}$ *setup*. $\mathscr{I}$ performs the `ISetup` algorithm obtaining a RSA key pair to be used by the partially blind signature scheme running the procedure $Init^{\mathscr{PBS}}$ as described in §2.2.2.

4.5.2 Affiliation/Disaffiliation

The scheme requires that each merchant who is interested in receiving coupons issued by $\mathscr{I}$ should affiliate to $\mathscr{I}$, signing an agreement. This agreement could consist on a list of terms and conditions. Without this affiliation, $\mathscr{M}$ cannot execute the `Claim` protocol to exchange a list of received coupons through a payment. At the end of the `Affiliation` protocol, $\mathscr{I}$ assigns a unique identifier ($id_{\mathscr{M}}$) to the merchant. With this information, $\mathscr{I}$ can maintain a list of affiliated merchants.

Similarly to the affiliation, the scheme defines a `Disaffiliation` protocol. The scheme distinguishes two kinds of disaffiliation: by merchant own decision or forced. In the first case, $\mathscr{I}$ only need to remove the merchant from the affiliation list. In the latter case, in addition to remove the merchant from the affiliation list, $\mathscr{I}$ keeps up an affiliation revocation list that can be useful if a merchant expelled from the affiliation tries to join the affiliation again in the future.

In any case, $\mathscr{I}$ and merchants do not share sensitive information, since merchants only need to know $\mathscr{I}$'s public key (e, n) and group's public key (pk^G). Therefore, merchants can leave the affiliation from $\mathscr{I}$ without causing security problems.

4.5.3 Group Manager Registration

Before $\mathscr{C}$ can use a $\mathbb{MC}^{2D}$ from $\mathscr{I}$, she must register herself with $\mathscr{G}$ by using the `GMRegistration` protocol in order to download and securely store a group key pair. $\mathscr{G}$ authenticates $\mathscr{C}$ using her digital certificate $Cert_{\mathscr{C}}$. So $\mathscr{G}$ links a user private key $\left(sk^G_{\mathscr{C}}\right)$ to the $\mathscr{C}$'s identity and sends her a group key pair $\left(pk^G, sk^G_{\mathscr{C}}\right)$ and a digital certificate $Cert_{\mathscr{G}}$ (issued by a trusted CA) that authenticates the group public key (pk^G). This link is needed to revoke the customer's anonymity when she misbehaves. At this step, customers agree that their identity can be disclosed whether they are not honest or whether an authority (e.g. a judge) requires the revocation of the anonymity.

4.5.4 Issue Protocol

The `Issue` protocol (Figure 4.4) involves $\mathscr{C}$ and $\mathscr{I}$ and allows the former to obtain an anonymous $\mathbb{MC}^{2D}$ from the latter. It is an anonymous process because $\mathscr{I}$

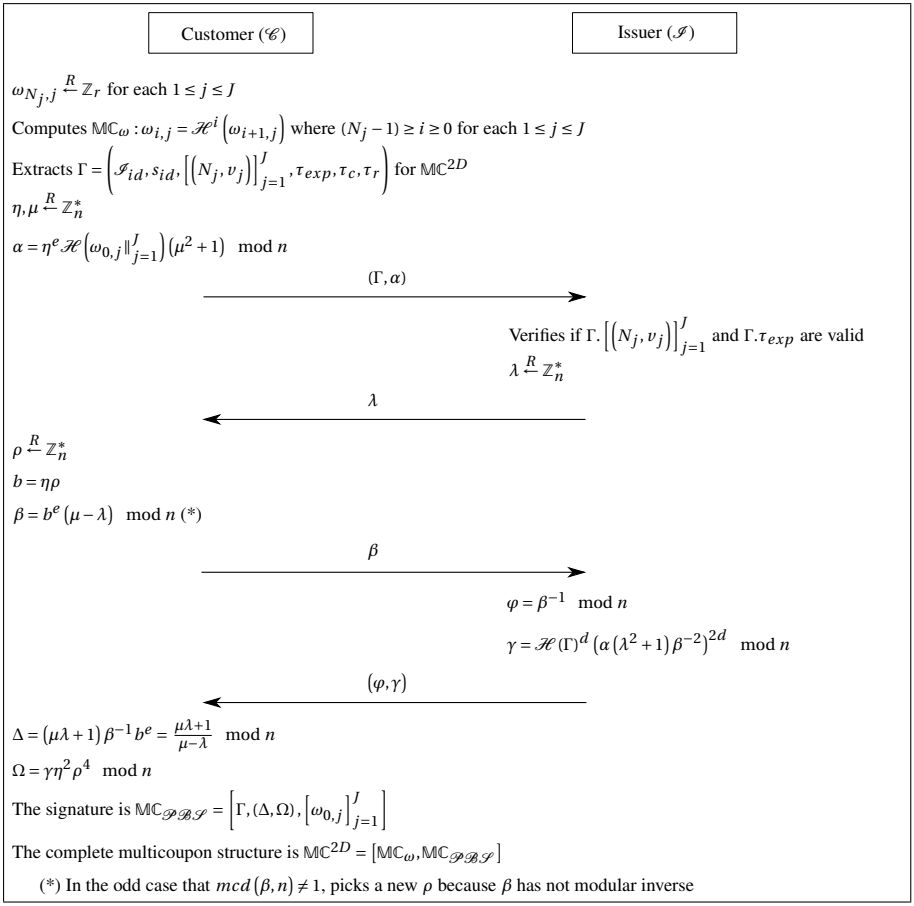


Figure 4.4: `Issue` protocol flow.

does not require any kind of user identification nor pseudonym. In addition, neither $\mathbb{MC}_{\mathscr{PBS}}$ nor $\mathbb{MC}_\omega$ contain data about $\mathscr{C}$'s identity. Moreover, thanks to the way we use a partially blind signature scheme, the list of multicoupons identifiers, denoted as $\left[\omega_{0,j}\right]_{j=1}^{J}$, is hidden from $\mathscr{I}$, so $\mathscr{I}$ has no information about it, except that it is usable and properly issued. Thus, nobody (but $\mathscr{G}$) can trace $\mathscr{C}$'s activities.

The protocol starts when $\mathscr{C}$ picks J random seed identifiers and using hash chains properties, she obtains $\mathbb{MC}_\omega$, that is, J multicoupons containing each of them up to N_j coupons with v_j value. Then, she must choose and agree with $\mathscr{I}$ on the public common information Γ (see §4.4.2). After that, $\mathscr{C}$ blinds the concatenated

list of multicoupon identifiers using a cryptographic hash function $\left(\mathscr{H}\left(\omega_{0,j}\|_{j=1}^{J}\right)\right)$ and sends it, together with the common information, to $\mathscr{I}$.

After the execution of the `Issue` protocol, on one hand $\mathscr{C}$ obtains a multicoupon defined as $\mathbb{MC}^{2D} = [\mathbb{MC}_{\omega}, \mathbb{MC}_{\mathscr{PBS}}]$. The first part is the set of coupons while the second part is formalized as $\left[\Gamma, (\Delta,\Omega), \left[\omega_{0,j}\right]_{j=1}^{J}\right]$, where Γ is the common information, (Δ,Ω) contains verification data and $\left[\omega_{0,j}\right]_{j=1}^{J}$ is the list of the multicoupons identifiers (see Figure 4.3). On the other hand, $\mathscr{I}$ knows nothing about the $\mathbb{MC}^{2D}$ but it is properly issued and ready to use on the multi-merchant environment.

The $\mathbb{MC}_{\mathscr{PBS}} = \left[\Gamma, (\Delta,\Omega), \left[\omega_{0,j}\right]_{j=1}^{J}\right]$ verification can be done applying a public equation that can be used by anyone who knows the $\mathscr{I}$'s RSA public key (e,n) and the $\mathbb{MC}_{\mathscr{PBS}}$ element. In order to do it, a verifier follows the $Verify^{\mathscr{PBS}}$ (see §2.2.2) procedure checking if the following equation holds:

$$\Omega^{e} \stackrel{?}{\equiv} \mathscr{H}(\Gamma)\,\mathscr{H}\left(\omega_{0,j}\|_{j=1}^{J}\right)^{2}\left(\Delta^{2}+1\right)^{2} \mod n$$

If the result is true, the verifier is convinced that the $\mathbb{MC}^{2D}$ is properly issued and the $\mathbb{MC}^{2D}$ is accepted. Otherwise, the verifier rejects the $\mathbb{MC}^{2D}$.

4.5.5 Multiredeem Protocol

The `Multiredeem` protocol (Figure 4.5) runs between $\mathscr{C}$ and $\mathscr{M}$. It is initiated by $\mathscr{C}$ and it allows her to spend coupons from her $\mathbb{MC}^{2D}$ in exchange of a service from the desired $\mathscr{M}$. The `Multiredeem` protocol can be executed by $\mathscr{C}$ while the $\mathbb{MC}^{2D}$ is not yet expired, i.e., while $\tau_{now} \leq \tau_{exp}$.

This protocol defines a four steps exchange, where $\mathscr{C}$ can multiredeem various coupons from different multicoupons with $\mathscr{M}$ using a single transaction. This fact enhances the computing and networking efficiency, as we will explain in §4.6.

In order to execute the exchange, $\mathscr{C}$ needs to send $\mathscr{M}$ a set of payment informations from the first unused coupon within each selected multicoupon up to the value of the requested service. This set of payment information is defined formally as $A_{l_j,j,n_j} = \left[\omega_{i,j}, (l_j, j), n_j\right]_{j=1}^{J}$, i.e., the list of payment information associated to all the first unused coupons together with the corresponding coordinates (l_j, j) and the number of the needed coupons n_j from each selected multicoupon j within $\mathbb{MC}_{\omega}$.

This set of payment information (A_{l_j,j,n_j}) along the merchant identifier $(id_{\mathscr{M}})$ and a redeem identifier (id_r) are group signed by $\mathscr{C}$ (using her group private key $sk_{\mathscr{C}}^{G}$) and sent to $\mathscr{M}$. Then, $\mathscr{M}$ verifies that the presented coupons are bound to the $\mathbb{MC}^{2D}$, and $\mathscr{C}$ has a sufficient number of coupons to pay for the requested service: $\lfloor l_j/2 \rfloor + n_j \leq N_j$. Moreover, $\mathscr{M}$ must check that the $\mathbb{MC}_{\mathscr{PBS}}$ and the presented set of coupons are properly signed: the partially blind signature in the former case, and

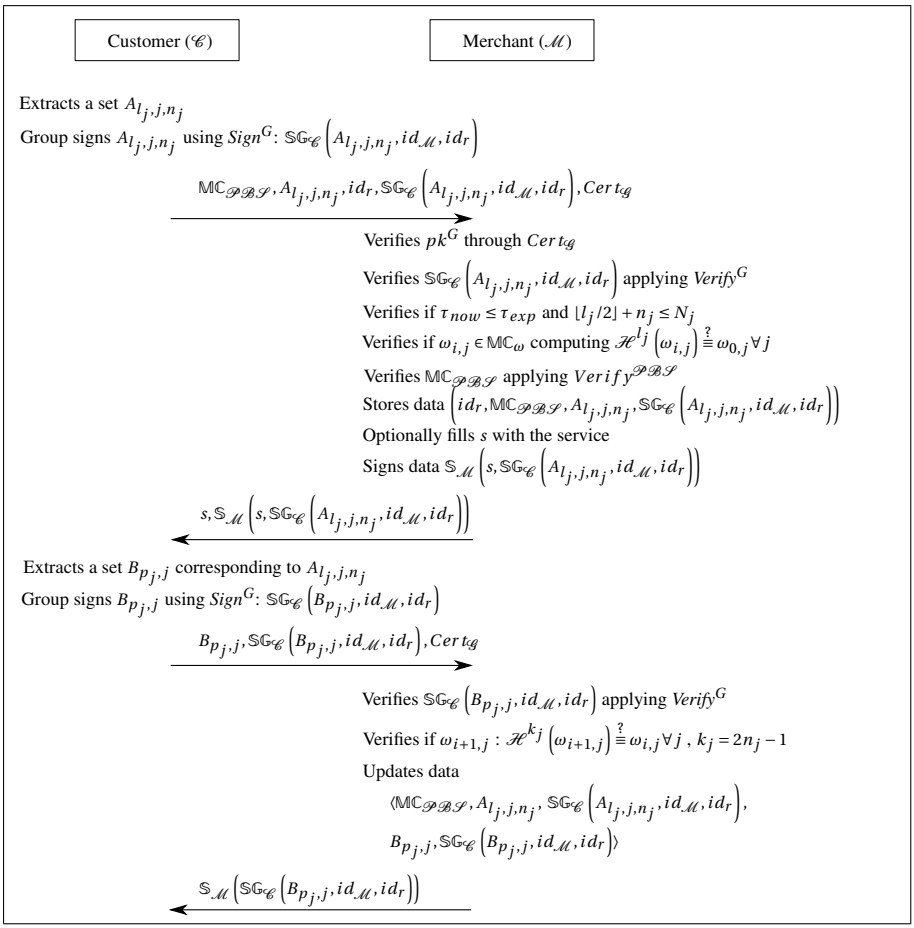


Figure 4.5: `Multiredeem` protocol flow.

the group signature in the latter case. If verifications do not hold, $\mathscr{M}$ rejects the service request and he does not provide the service to $\mathscr{C}$.

If it passes the validations, $\mathscr{C}$ should send signed (using $sk^G_{\mathscr{C}}$) the set of proof information bound to A_{l_j,j,n_j}. This proof information is used by $\mathscr{M}$ to validate the previously sent payment information. The set of proof information is defined formally as $B_{p_j,j} = \left[\omega_{i+1,j}, (p_j, j)\right]_{j=1}^{J}$ where $p_j = l_j + (2n_j - 1)$ depicts the position of the proof information to be used in the multicoupon j. If all verifications hold and all coupons are not previously used, $\mathscr{M}$ updates his database with the received set of coupons. Otherwise, $\mathscr{M}$ will reject $\mathscr{C}$'s request. The reuse detection and avoidance

can be performed checking whether any single coupon is in the $\mathscr{M}$'s local database or in the $\mathscr{I}$'s global database executing the `Claim` protocol (see §4.5.6). Note that the `Claim` protocol could be used either online or offline, depending on $\mathscr{M}$ and whether $\mathscr{I}$ allows it.

Finally, $\mathscr{M}$ sends $\mathscr{C}$ a signed message as a proof of the accepted coupons, together with the pair of identifiers $(id_{\mathscr{M}}, id_r)$.

4.5.6 Claim Protocol

The `Claim` protocol (Figure 4.6) is executed by $\mathscr{M}$ to request $\mathscr{I}$ a money deposit in exchange of a set of received coupons from customers (as `Issue` protocol does, `Claim` protocol uses a single transaction to send multiple coupons). This protocol can be executed any time before τ_c ($\tau_{now} \leq \tau_c$), thus $\mathscr{M}$ can execute partial claims, for example, at the end of the day.

When the `Claim` protocol is executed, $\mathscr{I}$ performs a list of verifications over the data sent by $\mathscr{M}$. For instance, it checks whether $\mathscr{M}$ is actually affiliated (included in the merchants affiliation list), and some other verifications similar to those made at the `Multiredeem` procedure. If all verifications are successful, $\mathscr{I}$ authorizes a deposit that can be done through a payment method, such as a deposit made by e-cash or into a bank account. If some verification fails, $\mathscr{I}$ stops the transaction and notifies it to $\mathscr{M}$. If $\mathscr{I}$ detects an already spent coupon (reuse attempt), $\mathscr{I}$ sends a request to $\mathscr{G}$ together with the data sent by $\mathscr{M}$ in order to prove the reuse. $\mathscr{G}$ checks it using the received data and if he also concludes that a reuse has been tried, then $\mathscr{G}$ can reveal the identity of the misbehaving customer through the use of the $Open^G$ procedure (as explained in §2.3). Then, $\mathscr{C}$'s identity may be reported to the proper authority in order to proceed as that authority determines.

4.5.7 Refund Protocol

The `Refund` protocol is an optional protocol that can be applied if the involved parties agree. It is used by $\mathscr{C}$ to request $\mathscr{I}$ a reimbursement of unspent coupons when the $\mathbb{MC}^{2D}$ is no longer valid and it is not fully used. This protocol can be executed while $\tau_c < \tau_{now} \leq \tau_r$. Note that this time range is later than the time window where $\mathscr{M}$ is allowed to request a deposit in exchange of a list of coupons received from customers. So, it is clear that whether $\mathscr{M}$ has claimed all of his received coupons, no problems arise with double-claims, because if $\mathscr{C}$ tries to refund coupons already claimed by $\mathscr{M}$, $\mathscr{I}$ could ask $\mathscr{G}$ to revoke $\mathscr{C}$'s anonymity. The protocol flow is omitted because it is quite similar to the `Claim` protocol. The only remarkable difference is the fact that $\mathscr{C}$ remains anonymous, i.e., she does not sent data digitally signed using her digital credentials. If the protocol ends properly, $\mathscr{I}$ authorizes a refund analogous to the `Claim` protocol specification. Otherwise, if $\mathscr{C}$ has tried to

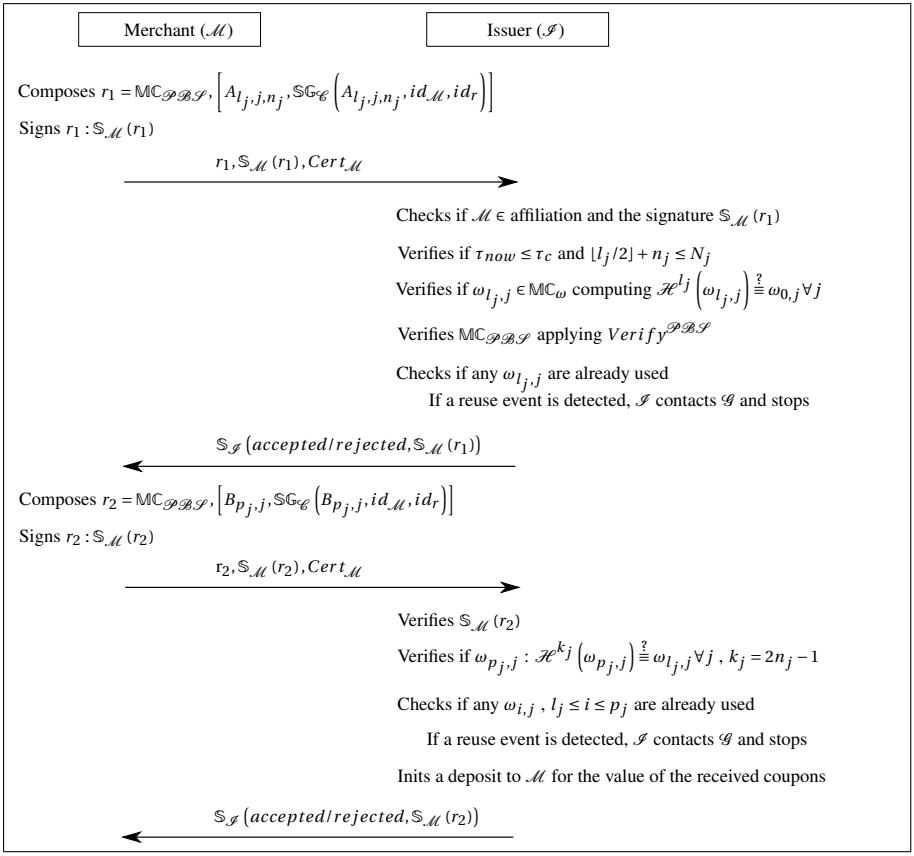


Figure 4.6: `Claim` protocol flow.

cheat, then $\mathscr{I}$ can request $\mathscr{G}$ to revoke $\mathscr{C}$'s anonymity in the same way as `Claim` protocol defines.

4.6 Analytical Efficiency Comparison with Armknecht Scheme

We analytically compare the performance in terms of computational cost measured as the complexity of the operations performed in each protocol run. For simplicity and without loss of generality, we have obtained the computational cost as the number of long and short exponentiations [46, 55, 96]. The rest of the operations have been considered negligible compared to exponentiation operations. Indeed, the cost to apply a hash function (e.g. SHA2) is similar to two modular multiplications.

Modular multiplications are negligible respect to exponentiations because an exponentiation with an l-bit exponent is the order of l modular multiplications. Thus, considering that a short exponentiation is about 160 bits [96], the number of modular multiplications required in both Armknecht's scheme [25] and in our proposal, is very low with regard to a modular exponentiation.

Table 4.3: Multi-merchant solutions - An analytical evaluation comparison.

	Operations		
	`Issue`	`Redeem`	`Claim`
Armknecht et al. [25]	$(2k+9)E_s+2E_l$	$(16+6k)E_s+(4+2k)E_l$	$k(2E_s+2E_l)$
Our proposal	$4E_s+2E_l$	$31E_s+E_l$	$2E_s+E_l$

k: number of single coupons in a multicoupon
E_s: short exponentiation
E_l: long exponentiation

Table 4.3 presents an analytical evaluation of the above parameters considering an scenario where each issued multicoupon contains k coupons. As shown in Table 4.3 and stated by authors in [25], the complexity of their solution is linear in k. This is because each coupon must be signed individually during the issue protocol. In addition, if a customer wants to redeem more than one coupon, the redeem protocol must be run as many times as coupons are needed because the redeem protocol was designed to work with individual coupons. This fact increases the computational cost of that multicoupon scheme, as Table 4.3 shows, because the computational cost of the Armknecht's scheme is linear w.r.t. k, meanwhile in our scheme is constant w.r.t. k. Even if the number of coupons is low (e.g. $k = 1$) and if it is assumed that a long exponentiation can be approximated to nine short exponentiations [96], our solution actually improves the efficiency of [25]. For example, let us consider the worst case in which a redeem protocol run with $k = 1$, then the number of short exponentiation operations required by Armknecht's scheme is 76 in front of the 40 required in our proposal. In fact, our proposal requires about less than half of the operations of the Armknecht's scheme. As k increases, the performance of the Armknecht's scheme declines while the performance of our scheme remains constant.

4.7 Conclusions

In this Chapter we have presented $\mathscr{MC}-2\mathscr{D}$, a scheme for multicoupons suitable for multi-merchant scenarios. This way, customers can redeem coupons from the same multicoupon to different merchants. $\mathscr{MC}-2\mathscr{D}$ shares from *μEasyPay* the use of a partially blind signature to provide customer with anonymity and untraceability.

However, the scheme presented in this Chapter provides more security according to the requirements stated by the scenario and the security model required by the described secure model for this kind of proposals. Thus, the scheme makes use of a group signature scheme to provide revocable anonymity for customers. So, in case customers act in a malicious behavior, such as forging multicoupons or trying to reuse coupons already redeemed, their anonymity can be revoked and their real identity sent to proper authorities. Moreover, $\mathscr{MC}-2\mathscr{D}$ also takes into account the merchant affiliation revocation in case he acts fraudulently. This disaffiliation, whether it is forced due to a merchant misbehavior or by his own decision, does not cause security problems since the system does not share any kind of sensitive data with merchants. In fact, our proposal is the first one to achieve this property. All these security requirements and the other ones described in the security model will be proved by a formal security analysis in Chapter 7, Section 7.2.

Regarding to the complexity analysis, we show how our scheme outperforms the previous proposal. It is due to various reasons. First, our scheme is based on lightweight cryptographic operations. Indeed, each coupon is a pair of hash identifiers related by the use of hash chain procedures. Additionally, group signature may cause a partial reduction of performance, but as we will demonstrate later in Chapter 11 it does not affect the performance at all. Secondly, our multicoupon scheme allows redeeming and claiming multiple coupons within a single protocol run. This way, as a key difference from previous proposal, and proved by a complexity analysis, our scheme needs a number of complex operations independent to the number of coupons issued, redeemed or claimed instead of needing as many protocol calls as coupons required, as previous proposal does. Moreover, a customer can use different coupons of the same multicoupon at different merchants without requiring any agreement and any shared data among the different merchants.

As already aforementioned, we will provide a full and practical implementation with Android mobile devices and a complete performance analysis on a production scenario in Chapter 11. This way, we will complete the complexity analysis presented in this Chapter with a practical performance analysis.

On one hand, customers achieve a high degree of privacy (anonymity and unlinkability), and they are protected against malicious merchants. That is, the reuse of coupons by merchants cannot involve the customer who used those coupons honestly. In fact, this kind of behavior by merchants could result in the revocation of their affiliation to the multicoupon system. This disaffiliation, whether it is forced due to a merchant misbehavior or by his own decision, does not cause security problems since the system does not share any kind of sensitive data with merchants. In fact, our proposal is the first one to achieve this property. On the other hand, merchants are protected from malicious customers trying either to reuse or to forge coupons, because the solution provides mechanisms to detect and avoid those misbehaviors. In case of misbehavior, $\mathscr{MC}-2\mathscr{D}$ can revoke the anonymity of malicious

customers to provide their identities to proper authorities. This way, as we provide measures to protect both customers and merchants from malicious behaviors, this proposal could improve the trust in this kind of systems, attracting more attention to them.

CHAPTER 5

AUTOMATIC FARE COLLECTION SCHEME

The electronic ticketing is another field within the electronic commerce world. There are lots of applications and services taking advantage of the use of ticket, so the electronic ticket version is always desirable to allow users to utilize these services by means of their mobile devices, as already stated in previous Chapters. Some services in which electronic ticketing can provide clear benefits to users and service providers are those where users are charged by a fare according to the use they make of the provided service, usually called Automatic Fare Collection (AFC) systems or services. In this Chapter we will describe and enhance a previous AFC scheme to deal with this concrete scenario. The proposal gives honest users anonymity and their actions are untraceable by service providers. In addition, we detect, describe and solve a security flaw allowing colluding users to cheat providers in order to pay a smaller fare than the rightful one. This work comes from a fruitful joint collaboration with researchers from *Universitat Rovira i Virgili (URV)* within a CONSOLIDER-INGENIO project framework.

5.1 Introduction

Electronic ticketing is another important field of electronic commerce as it could be applied to several everyday scenarios in which paper-based tickets are used. In the same way as we have described a proposal to transform a paper-based coupon to

the electronic world in Chapter 4, it is also interesting the same process applied to ticketing. However, the electronic ticketing field is so huge as to deal with it due to the large number of application and services with concrete functional and security requirements. Because of this, we focus this Chapter on a concrete scenario (but very important and interesting) such as this one composed by those services in which users are charged with a fare depending on the given utilization. We can list some examples such as public transportation (e.g. train, subway, bus, etc.), closed parking lots or access to public and sport facilities. These services demands special functional requirements. Indeed, an AFC system should provide easy and quick ticket checking (even in peak hours) and it should allow users checking entrance and exit by using their mobile devices. Besides functional requirements, focused on straightforwardness and performance, an AFC scheme should also protect users' privacy. Similarly to the multicoupon proposal, an AFC scheme should protect users from being identified by service providers (users want to be anonymous), as well as users also want that different sessions should be unlinkable and untraceable by those providers in order to avoid the creation of profiles related to their identities. However, service providers also want to be protected from misbehaving users trying to cheat in order to utilize their service with unfairly advantageous conditions. Indeed, providers want to be sure that they will be paid according to the fares establish. Therefore, an AFC scheme provide this protection to providers together with the ability to revoke the anonymity of malicious customers. This way, honest customer can utilize service without being identified, but in case they misbehave in some way, their anonymity can be revoked and their identity could be submitted to authorities to punish such behaviors.

In a previous work [34], authors describe a novel AFC scheme to charge users based on the distance they cover. So, that proposals can be applied to services such as road tolls. In that scheme, users receive an entrance ticket before to enter the service and they will be charged according to the distance between entrance and leaving points. Honest users act anonymously but misbehaving users can suffer an anonymity revocation in which their anonymity is revealed. Moreover, providers cannot trace actions made by users with their real identity. So, the scheme supplies revocable anonymity for malicious users by means of a group signature scheme (as $\mathscr{MC}-2\mathscr{D}$ multicoupon scheme does). However, the original AFC scheme is vulnerable to a colluding attack in which users moving in opposite directions exchange their respective entrance tickets as to be charged with an unfairly smaller fare than the rightfully one. Then, the committed solution is to add the direction as a parameter in the entrance ticket. If this solution is applied and users cannot change their direction without leaving the system and checking stations are also separated by direction, the scheme can avoid the colluding attack. However, if we take into account scenarios where entrance and exit stations are not really separated by direction, in such a way that users can choose the direction to where they want to travel after

entering the service, the AFC scheme continues to be vulnerable as the direction parameter set in the entrance ticket is not useful anymore.

In order to enhance the scheme, together with researchers from the *Universitat Rovira i Virgili (URV)*, we propose in this Chapter a modified AFC scheme based on the original one presented in [34] to completely avoid the colluding attack even in scenarios without separated stations. In addition to the main contribution, we have defined a detailed security model with all required security properties an AFC scheme should achieve. Besides, these security requirements will be proved by an extensive security analysis presented in Chapter 7 (Section §7.3). Finally, a complete and functional implementation will be described in Chapter 12 along with a complete performance evaluation with mobile devices as to prove that AFC schemes described in this Chapter (i.e. the original AFC proposal and the enhanced version avoiding the colluding attack) can be deployed in a real scenario.

The Chapter is organized as follows. In Section 5.2 we describe how an AFC system is related to the electronic ticketing field. Moreover, we present the description of the involved parties and the security model with the considered security requirement an AFC scheme should achieve. In Section 5.3 we briefly review some of the previous AFC proposals. We describe the original distance-based AFC presented in [34] in Sections 5.4 and 5.5, but generalizing the description as to fit it regardless of whether the fare is computed based on time or distance. In Section 5.6 we list the changes to be applied to the general AFC scheme as to allow it to charge users based on either time or distance. It is also in this Section where we describe a colluding attack made by malicious users in scenarios without separated directions and stations. In Section 5.7 it is described an enhanced AFC scheme to avoid this colluding attack. Finally, in Section 5.8 we give the conclusion to this Chapter. As in the other Chapters, we leave the security analysis to Chapter 7 (Section §7.3) and implementation and performance evaluation to Chapter 12.

5.2 Overview: Automatic Fare Collection Systems

In this Section we describe the AFC scenario and the attached security requirements. First, we describe the entities involved in a general AFC scheme as well as the main classification among these systems: time-based AFC and distance-based AFC. Finally, we describe the security model with the security requirements that an AFC solution should achieve.

5.2.1 Scenario

In general, a digital ticket (ticket hereinafter) is a certificate that guarantees certain rights of the ticket owner [27, 97], such as the right to use a concrete service under

certain terms and conditions, usually described in the ticket. There are many applications and services that make use of tickets, and depending on the requirements considered for each application, the ticket properties will vary.

One of the fields of application of electronic ticketing are the Automatic Fare Collection systems, in which electronic ticketing is applied to scenarios where users pay for the enjoyed service before leaving the system depending on the usage they have had. In fact, users can be charged depending on two main parameters: the time they have used the service and the distance they have walked. Then, we can differentiate between time-based AFC (such as closed parking lots) and distance-based AFC (such as highway tolls).

Often, AFC solutions consider, at least, the following entities [28–33, 97]: the user (who uses the service), the provider (who offers the service to users) and the issuer (the entity in charge of ticket issuance). Regarding to general protocols, proposals often consider the following: *ticket issuance* (process in which either the issuer or the provider emits a ticket to the user) and *ticket redemption*, *consumption* or *validation* (procedure in which tickets are validated to allow user to utilize the service). However, other proposals consider additional protocols such as *ticket payment* and protocols to cover registration phases [30]. As stated before, as a consequence of the heterogeneity of applications and services, a concrete proposal will depend on the requirements of the target service.

5.2.2 Security Model

We follow the same way to construct a security model as in Chapters 3 and 4, together with general security requirements stated in §1.3 and definitions provided in [27] related to the electronic ticketing field, and specially applied to AFC solutions.

Compared to the general ticketing scenario, we have added two more entities (the payment manager and the group manager), while the issuer is the same entity as the provider. Thus, the entities considered in the AFC solution are the following:

- $\mathscr{U}$ (user). She is who accesses the system and pays for the received service when she leaves the system.

- $\mathscr{P}_{\mathscr{S}}$ and $\mathscr{P}_{\mathscr{D}}$ (source and destination providers). To generalize, we define separated providers, although both $\mathscr{P}_{\mathscr{S}}$ and $\mathscr{P}_{\mathscr{D}}$ can be the same entity whether the whole system is controlled by a single provider. $\mathscr{P}_{\mathscr{S}}$ is the entity who issues entrance tickets while $\mathscr{P}_{\mathscr{D}}$ computes the fare to be paid by $\mathscr{U}$ and issues exit tickets. The way to compute the fare can be based on either the time $\mathscr{U}$ has spent in the system (*time-based* fare) or the distance $\mathscr{U}$ has walked through the system (*distance-based* fare).

- $\mathcal{T}_P$ (payment manager). It is the entity in charge of manage accounts of users and providers. Thus, it controls and commits payments made by $\mathcal{U}$ when she leaves the system according to the protocol specifications.

- $\mathcal{T}_G$ (group manager). It is the party who manages the group keys and the revocation list. Moreover, $\mathcal{T}_G$ is the only entity who can disclose the anonymity of $\mathcal{U}$ in case it is required. It follows from the group manager entity defined by group signature schemes (see §2.3). It works as the group manager taken into account in the $\mathcal{MC}-2\mathcal{D}$ multicoupon scheme described in Chapter 4.

***Definition* 5.2.1** *(Automatic Fare Collection System). An Automatic Fare Collection System (AFC) involving a user $\mathcal{U} \in \{\mathcal{U}_0 \ldots \mathcal{U}_n\}$, a source station $\mathcal{P}_{\mathcal{S}}$, a destination station $\mathcal{P}_{\mathcal{D}}$, a payment manager $\mathcal{T}_P$ and a group manager $\mathcal{T}_G$, consists on a set of algorithms and interactive protocols:* {`Setup`, `MGSetup`, `URegistration`, `SysEntrance` *and* `SysExit`}*; and a set of resolutions or claims as interactive protocols:* {`UClaim1`, `UClaim2`, `PClaim1` *and* `PClaim2`}.

- `Setup`. It is a probabilistic algorithm in which $\mathcal{P}_{\mathcal{S}}$, $\mathcal{P}_{\mathcal{D}}$ and $\mathcal{T}_P$ deploy a public key pair certified by a trusted and recognized CA taking as input the security parameter (1^K).

- `GMSetup`. It follows from the `GMSetup` probabilistic algorithm already described in §4.2.2, in which $\mathcal{T}_G$, based on the security parameter (1^K), issues a group public key (pk^G) and $\mathcal{N}$ group member private keys $\left(\{sk^G_{\mathcal{U}_i}\}_{i=0}^{\mathcal{N}-1}\right)$.

- `URegistrationMG`. It is a probabilistic and interactive protocol between $\mathcal{U}$ and $\mathcal{T}_G$ that works similarly than the `GMRegistration` protocol described in §4.2.2, in which $\mathcal{U}$ acquires a group key pair $(pk^G, sk^G_{\mathcal{U}})$ and $\mathcal{T}_G$ links her identity to that group key pair for further anonymity revocations.

- `URegistrationMP`. It is a probabilistic and interactive protocol between $\mathcal{U}$ and $\mathcal{T}_P$ in which $\mathcal{U}$ registers anonymously with $\mathcal{T}_P$ by means of using a pseudonym. At the end of the protocol, $\mathcal{T}_P$ opens a prepaid account for $\mathcal{U}$ (under her pseudonym) to charge her according to the computed fare.

- `SysEntrance`. It is a probabilistic and interactive protocol between $\mathcal{U}$ and $\mathcal{P}_{\mathcal{S}}$ when $\mathcal{U}$ wants to enter the AFC service. $\mathcal{P}_{\mathcal{S}}$ takes as inputs its private key $(sk_{\mathcal{P}_{\mathcal{S}}})$ and the group public key (pk^G). $\mathcal{U}$ uses as inputs her group key pair $(pk^G, sk^G_{\mathcal{U}})$ and her pseudonym actually encrypted by the public key of $\mathcal{T}_P$. As a result of the protocol, $\mathcal{U}$ obtains an entrance ticket signed by $\mathcal{P}_{\mathcal{S}}$ $(\mathsf{t_{in}}^*)$ that allows $\mathcal{U}$ to enter the system. The entrance ticket contains data useful to compute the fare to be paid by $\mathcal{U}$ when she leaves the system.

Data computed and exchanged during the protocol is stored in a protocol view $V_{\mathscr{P}_{\mathscr{S}}}^{\texttt{SysEntrance}}$.

- `SysExit`. It is a probabilistic and interactive protocol between $\mathscr{U}$ and $\mathscr{P}_{\mathscr{D}}$ when the former wants to leave the AFC service. $\mathscr{T}_P$ is also engaged in the protocol to charge $\mathscr{U}$. $\mathscr{P}_{\mathscr{D}}$ takes as inputs its private key $(sk_{\mathscr{P}_{\mathscr{D}}})$ and the entrance ticket ($\mathsf{t_{in}}^*$) provided by $\mathscr{U}$. At the end of the protocol, $\mathscr{P}_{\mathscr{D}}$ sends $\mathscr{U}$ an exit ticket ($\mathsf{t_{out}}^*$) signed by $\mathscr{P}_{\mathscr{D}}$ to allow her to leave the system. Moreover, $\mathscr{T}_P$ charges $\mathscr{U}$ according to the computed fare based on current data at the moment of leaving the system and data carried by the entrance ticket. Data computed and exchanged during the protocol is stored in a protocol view $V_{\mathscr{P}_{\mathscr{D}}}^{\texttt{SysExit}}$.

- `UClaim1` and `UClaim2`. They are two probabilistic and interactive protocols allowing $\mathscr{U}$ and $\mathscr{T}_P$ be involved when $\mathscr{U}$ has either problems due to $\mathscr{P}_{\mathscr{D}}$ misbehavior or because communication troubles during the `SysExit` protocol. Thus, these claim protocols allow $\mathscr{U}$ to recover from a unfair situation. $\mathscr{U}$ uses as input some data to prove her honest behavior together with her pseudonym in order to anonymously authenticate with $\mathscr{T}_P$. As a result of the protocols, $\mathscr{U}$ is charged by $\mathscr{T}_P$ and the `SysExit` protocol continues normally.

- `PClaim1` and `PClaim2`. They are two probabilistic and interactive protocol in which $\mathscr{U}$, $\mathscr{P}_{\mathscr{D}}$, $\mathscr{T}_G$ and $\mathscr{T}_P$ are engaged. These protocols resolve unfair situations in which $\mathscr{P}_{\mathscr{D}}$ cannot charge $\mathscr{U}$ due she has misbehaved sending wrong or incorrect data during the `SysExit` protocol in order to difficult the process of charging. As a result of these protocols, $\mathscr{T}_G$ revokes the anonymity of $\mathscr{U}$ and $\mathscr{T}_P$ charges $\mathscr{U}$.

***Definition* 5.2.2** *(Correctness). If an honest $\mathscr{U}$ runs the `SysEntrance` protocol with an honest source provider $\mathscr{P}_{\mathscr{S}}$, then $\mathscr{U}$ obtains an entrance ticket properly issued and signed by $\mathscr{P}_{\mathscr{S}}$ and $\mathscr{P}_{\mathscr{S}}$ allows her to utilize the service. Using this valid entrance ticket, an honest $\mathscr{U}$ executes the `SysExit` protocol with an honest destination provider $\mathscr{P}_{\mathscr{D}}$. Then, the honest $\mathscr{U}$ receives an exit ticket properly issued and signed by $\mathscr{P}_{\mathscr{D}}$ while $\mathscr{T}_P$ charges $\mathscr{U}$ according to the fare computed based on the use she has had of the service, allowing $\mathscr{U}$ to leave it.*

In the next Sections, we provide the formal definition of the security requirements that an AFC solution has to achieve, formalizing the common requirements already stated in §1.3: unforgeability, non repudiation of origin, anonymity, unlinkability, untraceability, overspending avoidance and revocable anonymity of misbehaving customers.

5.2.3 Unforgeability

The core element of an AFC solution is the use of tickets to prove entrance and exit and charge users according to the service they have just enjoyed. So, AFC providers want to be protected from ticket forgery (such as improper modification) and they want that nobody other than them should be able to issue valid tickets.

First, consider an unforgeability game 0 played by an adversary $\mathscr{A}_0$ as a p.p.t. Turing Machine taking the role of a malicious user. $\mathscr{A}_0$ has all the parameters to enter the system (i.e. group credentials and public parameters), so:

- $\mathscr{A}_0$ runs the `SysEntrance` protocol with an honest source provider.

Once $\mathscr{A}_0$ wants to leave the system, $\mathscr{A}_0$ has to execute the `SysExit` protocol giving to $\mathscr{P}_{\mathscr{D}}$ the entrance ticket. $\mathscr{A}_0$ could try to forge the entrance ticket in order to modify some of the parameters to cheat $\mathscr{P}_{\mathscr{D}}$ in order to be charged with a fare less than the corresponding one. In that case, it is required that $\mathscr{A}_0$ has negligible probability to be allowed to leave the system paying less than the rightfully fare due to the use of a forged entrance ticket without honest $\mathscr{P}_{\mathscr{D}}$ detecting it by means of the use of the `SysExit` protocol.

Secondly, consider an unforgeabilty game 1 run by an adversary $\mathscr{A}_1$ as a p.p.t. Turing Machine acting as a malicious user, so:

- $\mathscr{A}_1$ runs the `SysEntrace` protocol with an honest source provider.

Even $\mathscr{A}_1$ has a valid entrance ticket issued by $\mathscr{P}_{\mathscr{S}}$, $\mathscr{A}_1$ tries to self-issue another ticket with her own credentials and setting ticket parameters as she wants. Then, it is required that $\mathscr{A}_1$ has negligible probability on be accepted by honest $\mathscr{P}_{\mathscr{D}}$ if she uses the self-issued ticket during the `SysExit` protocol. So, it is required that tickets should be authentic and rightfully generated only by allowed parties correctly identified by their own credentials.

Similarly, in an unforgeability game 2 run by an adversary $\mathscr{A}_2$ as a malicious provider, he could try to forge a message issued by $\mathscr{U}$ during both `SysEntrance` and `SysExit` protocols. Then, it is also required that messages issued by users cannot be modified.

***Definition* 5.2.3** *(Unforgeability). The AFC solution is unforgeable if no p.p.t. adversary $\mathscr{A}$ $(\mathscr{A}_0, \mathscr{A}_1, \mathscr{A}_2)$ can win any unforgeability game with non-negligible probability (in* `K`*).*

5.2.4 Non Repudiation of Origin

In general, non-repudiation of origin (NRO) means that who has sent or generated a message cannot deny it. So, applied to the AFC solution, providers cannot deny the issue of a ticket and users cannot deny they have sent a message.

***Definition* 5.2.4** *(Non-repudiation of origin (NRO)). The AFC solution provides non-repudiation of origin (NRO) if no p.p.t. adversary can deny the issue of a concrete message with non-negligible probability (in* K*).*

5.2.5 Anonymity

Users running the AFC scheme should be anonymous in a sense that neither providers nor payment manager are able to identify their real identities. Users have to be able to use tickets without requiring any type of identification during their use nor tickets have to include any information related to their real identity.

So, let us consider an anonymity game 0 in which $\mathscr{A}_0$ acts as a malicious provider (either source or destination provider), so:

- $\mathscr{A}_0$ can execute the `SysEntrance` with an honest user (similarly the `SysExit` protocol).

In this game, $\mathscr{A}_0$ tries to identify the user who has played the `SysEntrance` protocol (the `SysExit` protocol, resp.). Then, it is required that $\mathscr{A}_0$ has negligible probability to obtain the real identity of the user in any of the previous games.

Similarly, an adversary $\mathscr{A}_1$ playing an anonymity game 1 as a corrupted $\mathscr{T}_P$ cannot identify users with their real identities while they are engaged in the `SysExit` protocol.

***Definition* 5.2.5** *(Anonymity). The AFC solution is anonymous if there is no p.p.t. adversary $\mathscr{A}$ $(\mathscr{A}_0, \mathscr{A}_1)$ that can win any anonymity game while the user is using the system with non-negligible probability (in* K*).*

5.2.6 Unlinkability

The unlinkability property applied to the AFC scenario is defined as the impossibility for a provider to link two different journeys. It means that tickets issued for different journeys cannot be linked.

Formally, consider an adversary $\mathscr{A}$ as a p.p.t. Turing Machine playing the unlinkability game taking the role of a malicious provider or a coalition of them. Besides, consider two users ($\mathscr{U}_0$ and $\mathscr{U}_1$), who already have used the service, so they have a pair of entrance and exit tickets corresponding to that journey ($\mathsf{t}_{\mathsf{in}_0^*}$, $\mathsf{t}_{\mathsf{out}_0^*}$,

$\mathtt{t_{in}}_1^*$ and $\mathtt{t_{out}}_1^*$). Note that the security model allows providers to link entrance and exit tickets within the same journey. Consider first only the entrance tickets, so $\mathscr{A}$ has two views containing the transcript of each `SysEntrance` protocol execution $\left(\mathcal{V}_{\mathscr{A}}^{\mathtt{SysEntrance}_0}\right.$ and $\left.\mathcal{V}_{\mathscr{A}}^{\mathtt{SysEntrance}_1}\right)$. $\mathscr{A}$ can play with the scheme, so:

- $\mathscr{A}$ can execute the `SysEntrance` with an honest user.

At some point of the unlinkability game, $\mathscr{A}$ outputs a bit $b = \{0,1\}$ and executes a `SysEntrance` protocol with the honest user $\mathscr{U}_b$ who has received the entrance ticket $\mathtt{t_{in}}_b^*$, and stores data exchanged in a view $\mathcal{V}_{\mathscr{A}}^{\mathtt{SysEntrance}_b}$. Then, $\mathscr{A}$ outputs another bit $b' = \{0,1\}$. Therefore, it is required that the advantage for $\mathscr{A}$ to successfully deduce whether $\mathscr{U}_b \equiv \mathscr{U}_{b'}$ is negligible. The same reasoning applies to the linkability of two exit tickets.

***Definition* 5.2.6** *(Unlinkability). The AFC scheme avoids linkability of different journeys if there is no p.p.t. adversary $\mathscr{A}$ that can win the unlinkability game with non-negligible linkability advantage over random toss of a coin (in* `K`*).*

5.2.7 Untraceability

Even providers can link an entrance ticket with the corresponding exit ticket for the same journey, providers cannot trace the identity of the user $\mathscr{U}$ who has used the service with her actions. Besides, providers cannot trace $\mathscr{U}$ in different journeys. Indeed, providers should not be able to trace users in order to avoid building user profiles, such as when and where users usually enter and leave the AFC service.

The untraceability game follows from the above unlinkability game, in which $\mathscr{U}$ cannot know, in addition, whether $\mathscr{U}_b$ is in fact $\mathscr{U}_0$ or $\mathscr{U}_1$. It is the same reasoning already provided in §3.2.7 about the micropayment scheme presented in §3.

***Definition* 5.2.7** *(Untraceability). The AFC scheme avoids traceability of the identity of users by linking their actions in either the same or different journeys, if there is no p.p.t. adversary $\mathscr{A}_0$ able to win the untraceability game with non-negligible probability (in* `K`*).*

5.2.8 Overspending Avoidance

An electronic ticketing solution can allow two types of tickets depending on the number of times users can use them. Indeed, there are tickets that can only be used once (*non-reusable* tickets) and other ones that can be used a pre-established number of times within some validity time range (*reusable* tickets) [27]. Regardless of whether tickets are reusable or non-reusable, the system should control that tickets are only

used exactly the number of times agreed at the moment of their issuance. In the case of the AFC solution, both entrance and exit tickets should be used only once, and they have to accomplish that they are used within the allowed validity time.

We can define an overspending game in which an adversary $\mathscr{A}$ plays the role of a malicious user. $\mathscr{A}$ has all the parameters to use the scheme, so:

- $\mathscr{A}$ can run the `SysEntrance` protocol with an honest source provider.
- $\mathscr{A}$ can run the `SysExit` protocol with an honest destination provider.

Then, at some point of the overspending game, $\mathscr{A}$ can try to reuse an already validated entrance ticket at the moment of leaving the system. So, it is required that the probability for $\mathscr{A}$ to reuse the same ticket without it being detected by the destination provider is negligible.

***Definition* 5.2.8** *(Overspending Avoidance). The AFC solution avoids overspending if there no p.p.t. adversary $\mathscr{A}$ that can win the overspending game with non-negligible probability (in* `K`*).*

5.2.9 Revocable Anonymity of Misbehaving Users

Similarly to the $\mathscr{MC}-2\mathscr{D}$ multicoupon scheme described in Chapter 4, the AFC solution must provide revocable anonymity in order to reveal the real identity of users if they misbehave. The anonymity revocation could be applied whether a user forges a ticket (as described in §5.2.3) or reuses a ticket in different journeys (see overspending avoidance in §5.2.8). So, the revocable anonymity game is similar to the corresponding game defined in §4.2.7, but this time the adversary $\mathscr{A}$ takes the role of a malicious user executing the `SysExit` protocol with an honest destination provider.

***Definition* 5.2.9** *(Revocable anonymity of misbehaving users). The AFC solution allows the anonymity revocation and the correct identification of misbehaving users if there is no p.p.t. adversary $\mathscr{A}$ that can win the anonymity revocation game with non-negligible probability (in* `K`*).*

5.3 Related Work

In the literature there are AFC proposals that considers the application of electronic ticketing to scenarios with massive transport of people. We have reviewed some of that proposals that consider anonymity for users, offering revocable anonymity [27–33]. In Table 5.1 we classify the analyzed proposals depending on the level

Table 5.1: AFC Proposals - A comparative analysis.

Proposal	Anonymity	Untraceability	Device
Wang et al. [28]	Revocable	-	Mobile
Buttyán et al. [29]	Revocable	-	Smart-card
Heydt-Benjamin et al. [30]	Revocable	✓	Mobile and Smart-card
Hong and Kang [31]	Revocable	-	Smart-card
Jorns et al. [32]	Revocable	-	Mobile
Madlmayr et al. [33]	Revocable	-	Mobile

✓YES , -NO

of anonymity guaranteed for users, the availability to trace different journeys of the same user and the devices used in that systems.

In the majority of those schemes, the provider can link different journeys from the same user [28, 29, 31–33]. In that schemes, the disclosure of the user's identity leads to the disclosure of all the journeys of the same user. In fact, providers know here and when their users enter and leave the AFC service, opening a critical privacy flaw for users: profile creation. These profiles are useful for providers because they can use these information to know how users utilize their service and to improve the transportation system. However, it could be also a key tool for providers to share or sell information to other parties, e.g. for commercial interests. Therefore, proposals should protect users from being profiled by providers.

In [30], the provider cannot trace these journeys, but users need to issue a new credential for every journey. So, it causes an important degradation in the scheme's performance as it slows down the time of issuing entrance and exit tickets, which is a key point in this type of services.

The AFC proposed in [32] defines a ticket that includes route information obtained with some localization technology, such as GPS. It is used with mobile phones and PDAs. The anonymity is managed by pseudonyms and achieves revocable anonymity.

Related to the devices used in the proposals, the latest trends go in the direction of using mobile devices (e.g. mobile phones, PDAs, smart phones, etc.) [28, 30, 32, 33] instead of smart cards [29–31]. As these services often are used in mobility scenarios, it is important to consider that users will use the ticketing scheme with their mobile devices. Therefore, proposals should focus io it to define efficient and practical schemes to be used in those special mobile devices.

As already pointed out in §5.2.1, AFC proposals can be applied to either time-based or distance-based scenarios. Nowadays there are some examples of services that can be managed by AFC schemes. Regarding to time-based AFC services, we can cite services such as public or sport facilities, parking lots and public transport

services (tickets with limited time of validity). However, it is more usual for transport services apply fares based on distance. Indeed, examples of distance-based AFC services are highway tolls, subway, train and bus.

These services share a common and key functional requirement aside the privacy for their users: tickets must be validated as quick as possible, even in peak hours (such as peak hours in public transport services). Therefore, we need to define secure and efficient AFC systems to fulfill both security and functional requirements.

5.4 General Automatic Fare Collection Scheme

In this Section we review the AFC system originally presented in [34], which was designed as to charge users based on the distance they utilize the service. Due to the fact there are few differences between applying the general scheme to either time-based or distance-based scenarios, we provide a general description of the original protocol as a general scheme regardless of it is applied to time-based or distance-based scenarios. Then, in Section §5.6 we describe the minor changes to apply to the general system to become it in a specific scheme for time-based or distance-based fares. However, these schemes can only be applied to scenarios where entrances and exits are separated physically. Otherwise, if directions and checking stations are not separated, the distance-based AFC scheme is vulnerable to a colluding attack in which users traveling in opposite direction exchange their entrance tickets in order to be charged with a smaller fare than the use they really have made of the service. Therefore, in Section §5.7 we will provide an improved AFC system to avoid this attack.

5.4.1 Proposal Overview

The general AFC scheme involves parties and protocols formally described by the security model in §5.2.2. Figure 5.1 shows the scenario and the relationships among all parties.

First, $\mathscr{T}_G$ executes the `GMSetup` to deploy a list of group credentials to be linked to every user who wants to make use of the AFC service. So, $\mathscr{U}$ registers to $\mathscr{T}_G$ by means of the `URegistrationMG` protocol and she receives a group key pair from a group signature scheme. This way, $\mathscr{U}$ can sign data but remaining anonymous. Using these group credentials, $\mathscr{U}$ registers to $\mathscr{T}_P$ using the `URegistrationMP`. As a result, $\mathscr{U}$ opens a prepaid account in $\mathscr{T}_P$.

At this point, $\mathscr{U}$ has all parameters required to utilize the AFC service. So, when $\mathscr{U}$ is close to the entrance, she executes the `SysEntrance` protocol with $\mathscr{P}_{\mathscr{S}}$ to acquire a signed entrance ticket from the entrance provider $\mathscr{P}_{\mathscr{S}}$. If it all is correct, $\mathscr{U}$ is allowed to use the service. When $\mathscr{U}$ wants to leave it, she executes the `SysExit`

protocol with $\mathscr{P}_{\mathscr{D}}$ showing the signed entrance ticket. $\mathscr{P}_{\mathscr{D}}$ calculates the fare to be paid by $\mathscr{U}$ and if she is agree, she sends this information securely to $\mathscr{T}_P$ with his pseudonym (it is hidden from $\mathscr{P}_{\mathscr{D}}$). Then, $\mathscr{T}_P$ charges the fare to the corresponding $\mathscr{U}$'s account. If verifications hold, $\mathscr{U}$ receives an exit ticket as the evidence proving the protocol has been followed correctly. With the exit ticket, $\mathscr{U}$ is allowed to leave the service.

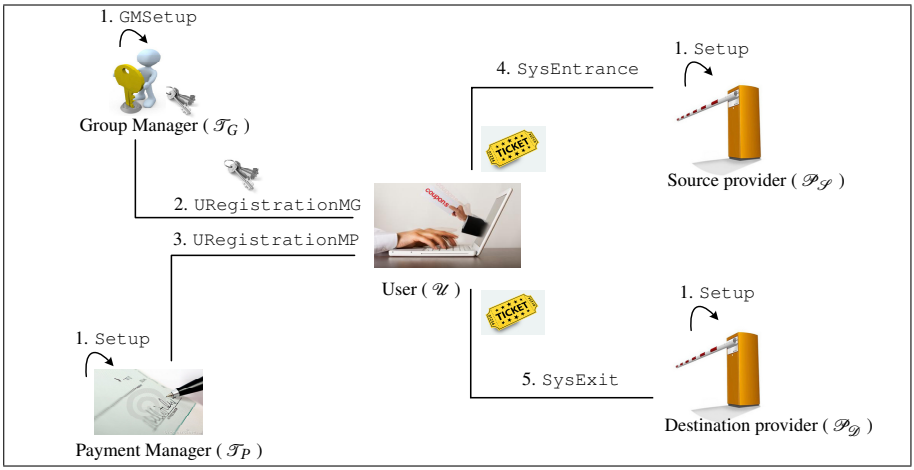


Figure 5.1: General AFC scenario and relations among parties.

If some troubles happen during the `SysExit` protocol, $\mathscr{U}$ and providers can recover from a failed execution by means of using some of the defined protocol claims.

5.4.2 Tickets Structure

In this Section, we show the information included in the entrance ticket (Table 5.2) and in the exit ticket (Table 5.3).

Table 5.2: Information included in the entrance ticket ($\mathsf{t_{in}}^*$)

Notation	Description
Sn	Serial number computed by $\mathscr{P}_{\mathscr{S}}$
Ps	Identifier of the entrance station (source provider)
τ_1	Time at the moment of entrance
σ^*	Commitment signed by $\mathscr{U}$
$\mathbb{S}_{\mathscr{P}_{\mathscr{S}}}(\mathsf{t_{in}})$	Digital signature by $\mathscr{P}_{\mathscr{S}}$ on the above parameters

Table 5.3: Information included in the exit ticket ($\mathsf{t_{out}}^*$)

Notation	Description
$\mathsf{t_{in}.Sn}$	Serial number set at the entrance
Pd	Identifier of the exit station (destination provider)
a	Fare just paid
τ_2	Time at the exit (payment timestamp)
$\mathbb{S}_{\mathscr{P}_\mathscr{D}}(\mathsf{t_{out}})$	Digital signature by $\mathscr{P}_\mathscr{D}$ on the above parameters

5.5 General Automatic Fare Collection Proposal Specification

In this Section, we describe the general AFC scheme that provides anonymity to users by the use of group signatures according to the *Definition 5.2.1*. Table 5.4 summarizes the notation used along the complete scheme description.

Table 5.4: General notation used along the protocol description.

Notation	Name
$\left(pk^G, sk^G_\mathscr{U}\right)$	Group pair key (from group signature scheme, see §2.3.3)
$\mathscr{L}_R$	List of group revocations
α	Exponentiation base
p, q	Random safe primes from $\mathbb{Z}_p$
r_j	j-th random number
τ_j	j-th timestamp
$\mathbb{S}_\mathscr{E}(c)$	Digital signature of the content c generated by the entity $\mathscr{E}$
$\mathbb{E}_\mathscr{E}[c]$	Probabilistic asymmetric encryption by $\mathscr{E}$ on c
$\mathbb{D}_\mathscr{E}[c]$	Decryption by $\mathscr{E}$ of ciphertext c
$y_\mathscr{U}$	$\mathscr{U}$'s pseudonym (for payment)
$x_\mathscr{U}$	Inverse exponentiation of $y_\mathscr{U}$ (secret)
$\mathscr{F}$	Fare calculation function
Ps	Source service provider identifier
Pd	Destination service provider identifier
$\mathsf{t_{in}}^*$	Entrance ticket, signed by $\mathscr{P}_\mathscr{S}$
$\mathsf{t_{out}}^*$	Exit ticket, signed by $\mathscr{P}_\mathscr{D}$
ok^*	Payment acceptance signed by $\mathscr{T}_P$
ko^*	Payment rejection signed by $\mathscr{T}_P$

5.5.1 Initial System Setup

The system should be globally deployed performing to set up procedures: providers initialization (`Setup` algorithm) and $\mathscr{T}_G$ key deployment (`GMSetup` algorithm).

- *Providers initialization (`Setup`).* Providers must acquire and certify a key pair from a public cryptosystem (such as RSA) in order to generate and verify digital signatures, and to perform asymmetric encryptions.

- $\mathcal{T}_G$ *key deployment (`MGSetup`).* As in the $\mathcal{MC}-2\mathcal{D}$ scheme, $\mathcal{T}_G$ must execute the $KeyGen^G$ algorithm from the group signature scheme (see §2.3.3). This way, $\mathcal{T}_G$ generates a group public key $\left(pk^G\right)$, n user private keys $\left(sk^G[n]\right)$ and a private key $\left(sk^G_{\mathcal{G}}\right)$. Additionally, $\mathcal{T}_G$ creates a revocation list ($\mathcal{L}_R$) and computes some public parameters such as α (public exponentiation base), p (prime number) and q (a safe prime such as $p = 2q + 1$).

5.5.2 User Registration

AFC users must register to both $\mathcal{T}_G$ and $\mathcal{T}_P$ in order to be accepted by the service.

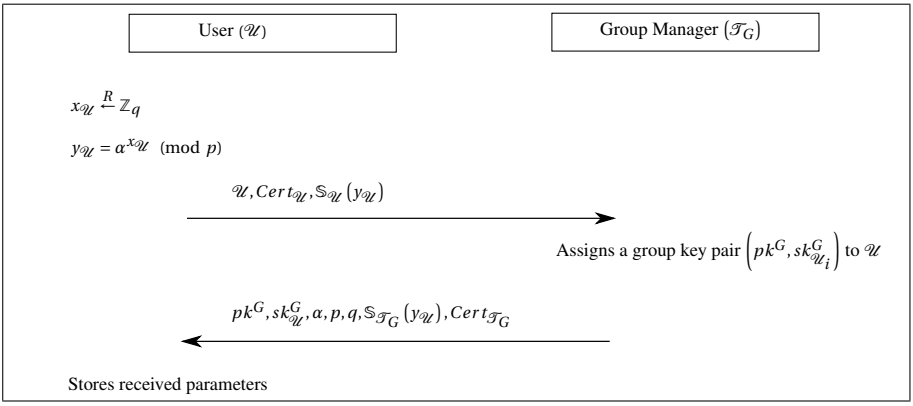


Figure 5.2: `URegistrationMG` protocol flow.

On one hand, similarly to the $\mathcal{MC}-2\mathcal{D}$ multicoupon scheme (Chapter 4, §4.5.3) in which it is also considered the use of a group signature scheme, AFC users should register with $\mathcal{T}_G$ by using the `URegistrationMG` protocol (Figure 5.2) to acquire and store in a secure way a group key pair $\left(pk^G, sk^G_{\mathcal{U}}\right)$ and a digital certificate $\left(Cert_{\mathcal{T}_G}\right)$ from a trusted CA certifying the group public key. At this point, $\mathcal{T}_G$ links the identity of $\mathcal{U}$ to a single secret key, so users should agree that their identity could be disclosed whether they are not honest or an authority requires the anonymity revocation. In addition to the group key pair, $\mathcal{U}$ receives common parameters $\left(\alpha, p, q\right)$ and her pseudonym signed by $\mathcal{T}_G$.

On the other hand, $\mathcal{U}$ registers anonymously to $\mathcal{T}_P$ using the `URegistrationMP` protocol (Figure 5.3). She uses her pseudonym just signed by $\mathcal{T}_G$ as input to a Zero-Knowledge Proof (ZKP) algorithm (such as Schnorr [98]) in order to convince $\mathcal{T}_P$

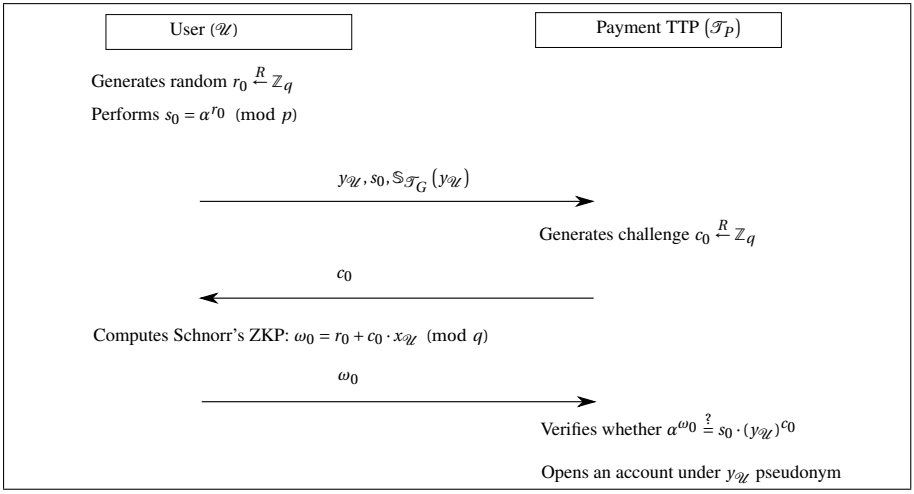


Figure 5.3: `URegistrationMP` protocol flow.

that she is actually the owner of her pseudonym $(y_{\mathscr{U}})$. At the end of the protocol, $\mathscr{T}_P$ opens an account for the user anonymously identified by $y_{\mathscr{U}}$ in which $\mathscr{U}$ must pay some money in with some payment method (bank account transfer, e-cash, etc.).

5.5.3 System entrance

Upon $\mathscr{U}$ is close to the entrance station controlled by the source provider ($\mathscr{P}_{\mathscr{S}}$), she starts the `SysEntrance` protocol (see Figure 5.4) by means of she will eventually receive an entrance ticket $\mathsf{t_{in}}^*$ signed by $\mathscr{P}_{\mathscr{S}}$. $\mathscr{U}$ computes an exponentiation which will be used later when she exits the system as to prove her identity without revealing it by means of a ZKP algorithm. Moreover, she computes $\mathscr{H}(k)$ from a random value k as to also prove later that she is the right owner of ticket. Finally, $\mathscr{U}$ sends these elements to $\mathscr{P}_{\mathscr{S}}$ group signed by using her private group key $(sk^G_{\mathscr{U}})$ and the group signature algorithm $(Sign^G)$. This way, $\mathscr{U}$ remains anonymous. Upon $\mathscr{P}_{\mathscr{S}}$ receives the message and all data is correct, $\mathscr{P}_{\mathscr{S}}$ builds the entrance ticket $(\mathsf{t_{in}})$, signs it with his private key and sends $\mathsf{t_{in}}^*$ to $\mathscr{U}$. Then, the source provider allows $\mathscr{U}$ to start using the service.

5.5.4 System exit

Upon $\mathscr{U}$ is beside exit station, which is controlled by the destination provider ($\mathscr{P}_{\mathscr{D}}$), she and $\mathscr{P}_{\mathscr{D}}$ engage the `SysExit` protocol (see Figure 5.5). The objective of the

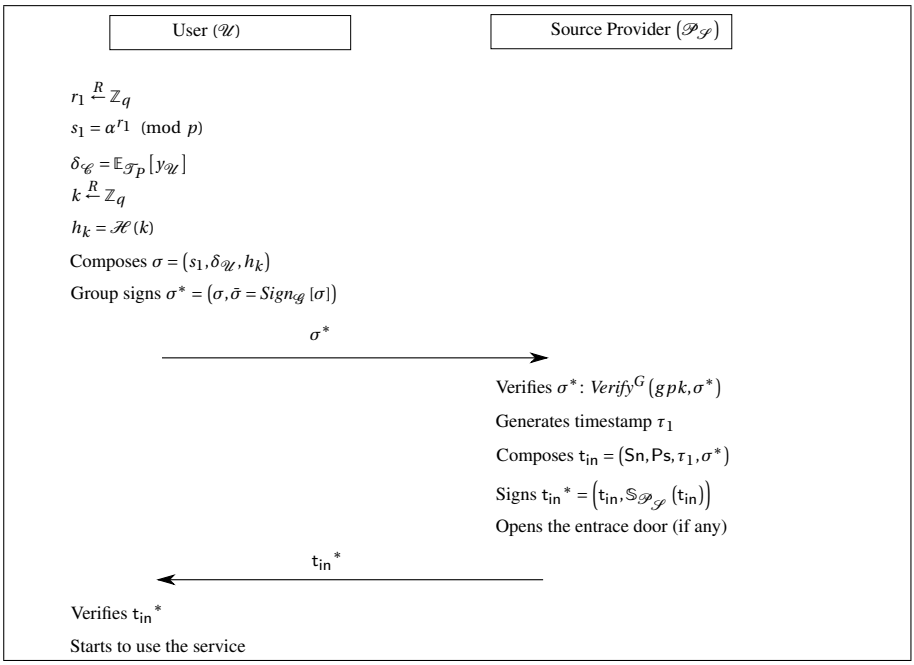


Figure 5.4: `SysEntrance` protocol flow.

protocol is to allow $\mathscr{U}$ to exit the system and to receive an exit ticket ($\mathsf{t_{out}}^*$) after paying the corresponding fare.

$\mathscr{U}$ shows the entrance ticket ($\mathsf{t_{in}}^*$) together with the proof that she is the right owner of this ticket. After several verifications, $\mathscr{P}_{\mathscr{D}}$ calculates the fare has to paid $\mathscr{U}$ according to the terms of service and the data carried by the entrance ticket. If $\mathscr{U}$ agree with the fare, she composes some information to prove anonymously her identity to $\mathscr{T}_P$ in order to be charged according to the aforementioned fare. Then, $\mathscr{T}_P$ charges the user identified by the pseudonym $y_{\mathscr{U}}$ and notifies $\mathscr{P}_{\mathscr{D}}$ accordingly. $\mathscr{P}_{\mathscr{D}}$ knows that $\mathscr{U}$ has been charged, so he composes and signs the exit ticket ($\mathsf{t_{out}}^*$), which allows $\mathscr{U}$ to exit the system.

However, in case some involved parties misbehave or a communication problem arises during the `SysExit` protocol, the AFC scheme provides hurt parties the opportunity to recover from an unfair situation through the use of different claim protocols. `UClaim1` and `UClaim2` are addressed to help users while `PClaim1` and `PClaim2` can be called by providers in case they have been harmed. These protocols can be called from BREAKPOINTS during the `SysExit` protocol execution (see Figure 5.5).

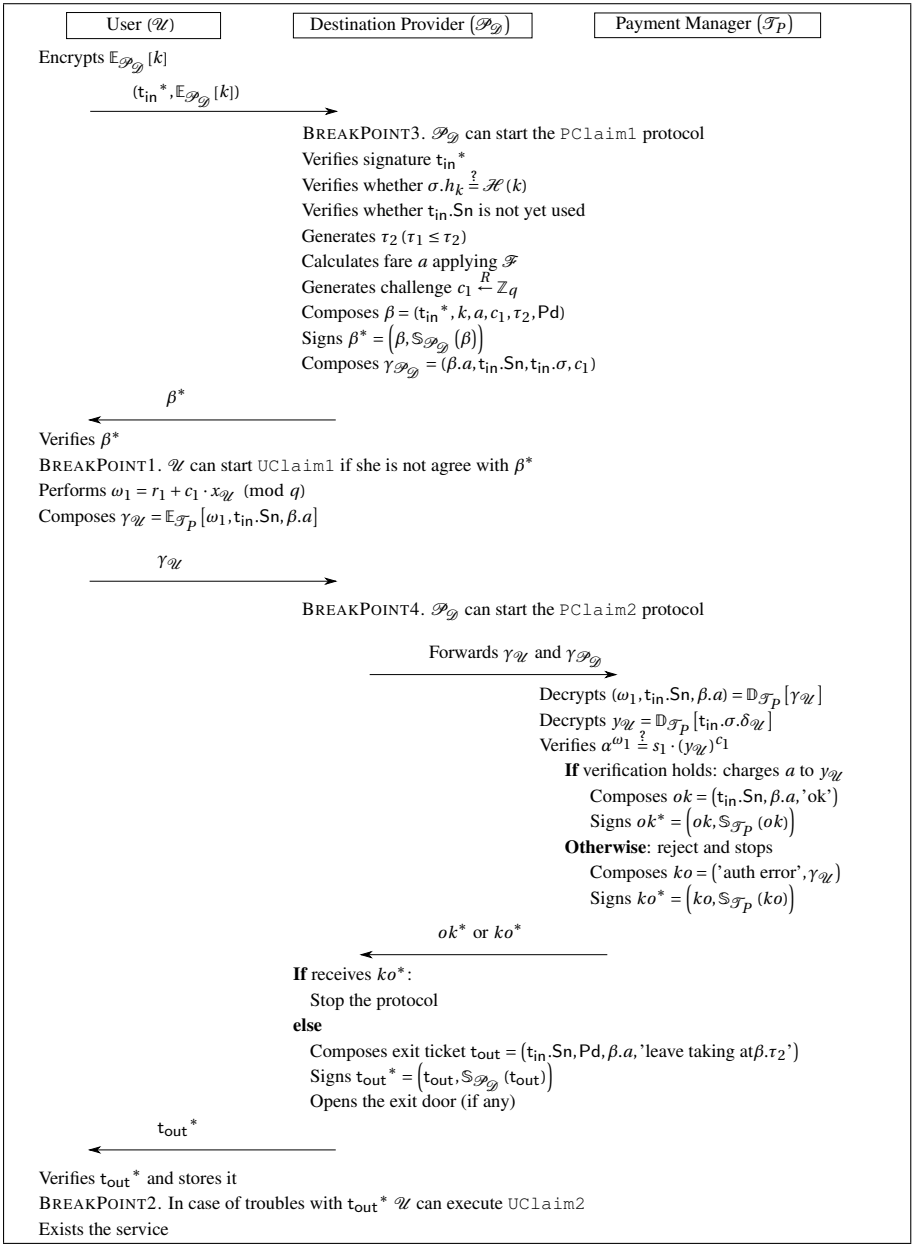


Figure 5.5: SysExit protocol flow.

5.5.5 First User Claim

In case $\mathscr{U}$ thinks that the β^* parameter, which contains the fare to be paid, is not right ($\mathscr{P}_{\mathscr{D}}$ could want to charge $\mathscr{U}$ with an unfairly expensive fare) or simply she has not received it. Then, $\mathscr{U}$ can run the `UClaim1` protocol (see Figure 5.6) to claim $\mathscr{T}_P$ about the wrong β^*. So, $\mathscr{T}_P$ calculates the new fare, sends $\mathscr{U}$ the new β^* and $\mathscr{U}$ continues with the execution of the `SysExit` protocol from the corresponding BREAKPOINT.

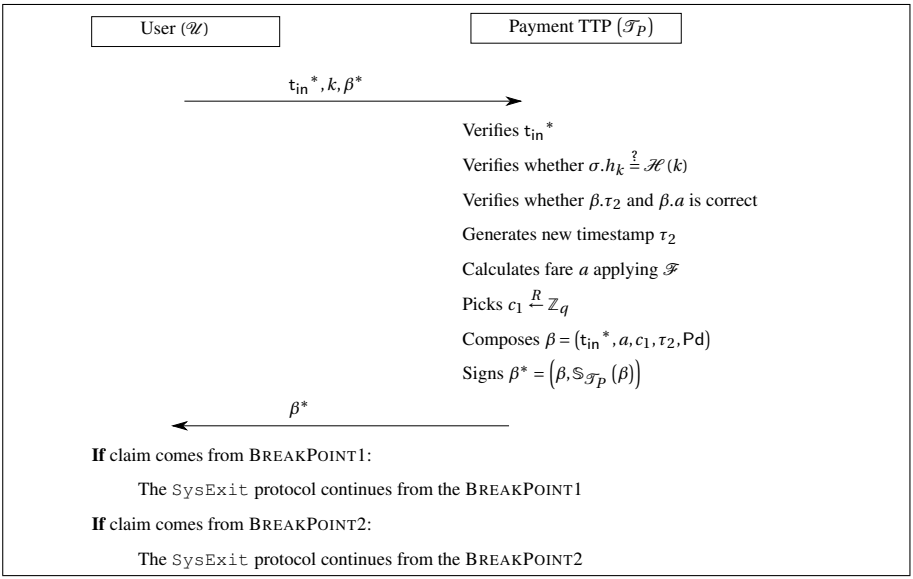


Figure 5.6: `UClaim1` protocol flow.

5.5.6 Second User Claim

In case $\mathscr{U}$ receives from $\mathscr{P}_{\mathscr{D}}$ a wrong $\mathsf{t_{out}}^*$ in the last message of the `SysExit` protocol or $\mathscr{P}_{\mathscr{D}}$ denies to send it, she can run the `UClaim2` protocol (see Figure 5.7) to receive another $\mathsf{t_{out}}^*$, but this time generated by $\mathscr{T}_P$. In fact, it is $\mathscr{T}_P$ who composes and signs the new exit ticket after charging $\mathscr{U}$ according to the presented fare.

Note that $\mathscr{T}_P$ must inform $\mathscr{P}_{\mathscr{D}}$ of its misbehavior or communication problems with users. In case either the behavior remains malicious or the network continues failing, $\mathscr{T}_P$ could take actions.

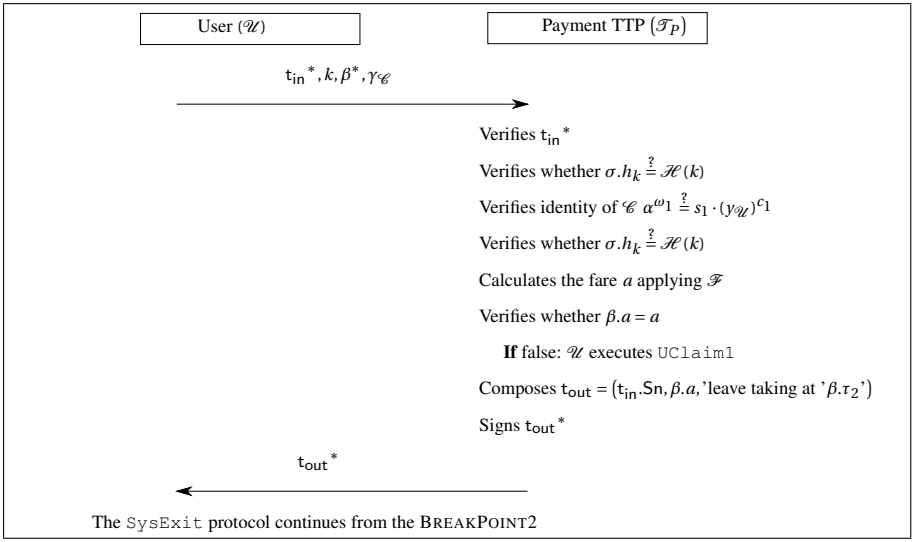


Figure 5.7: UClaim2 protocol flow.

5.5.7 First Provider Claim

In case $\mathscr{P}_{\mathscr{S}}$ receives a wrong message in the first step of the SysExit protocol from $\mathscr{U}$ or it has not been received, $\mathscr{P}_{\mathscr{D}}$ can run the PClaim1 protocol (see Figure 5.8) to inform both $\mathscr{T}_G$ and $\mathscr{T}_P$ about this fact. If $\mathscr{U}$ cannot prove that she has acted honestly, $\mathscr{T}_G$ revokes her anonymity and $\mathscr{T}_P$ charges her with the corresponding fare (and probably with a fine) thanks to the information provided by $\mathscr{T}_G$. As $\mathscr{U}$'s identity has been disclosed, it could be forwarded to proper authorities.

5.5.8 Second Provider Claim

Similarly to the last claim, $\mathscr{U}$ could provide a wrong $\gamma_{\mathscr{U}}$ to $\mathscr{P}_{\mathscr{D}}$ or $\mathscr{P}_{\mathscr{D}}$ may not receive it due to communication problems. In this case, $\mathscr{P}_{\mathscr{D}}$ can execute the PClaim2 protocol (see Figure 5.9) to request the anonymity revocation of the misbehaving user, as before.

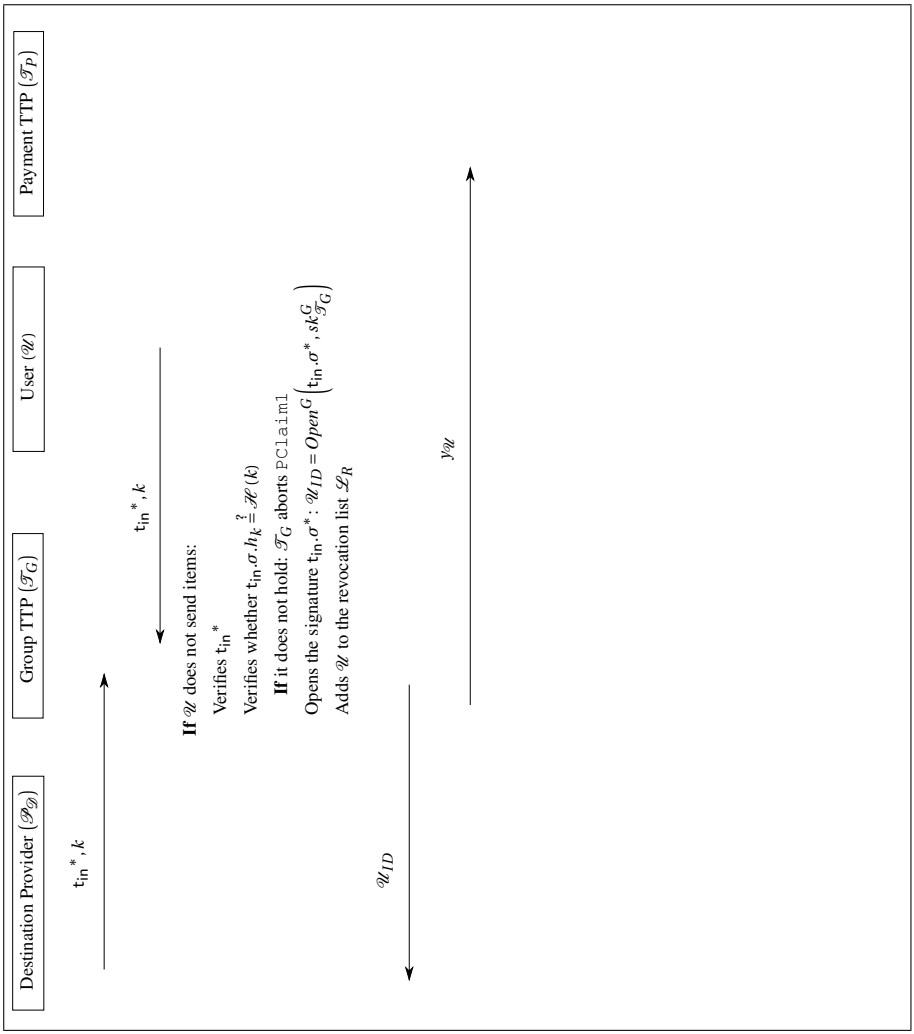


Figure 5.8: PClaim1 protocol flow.

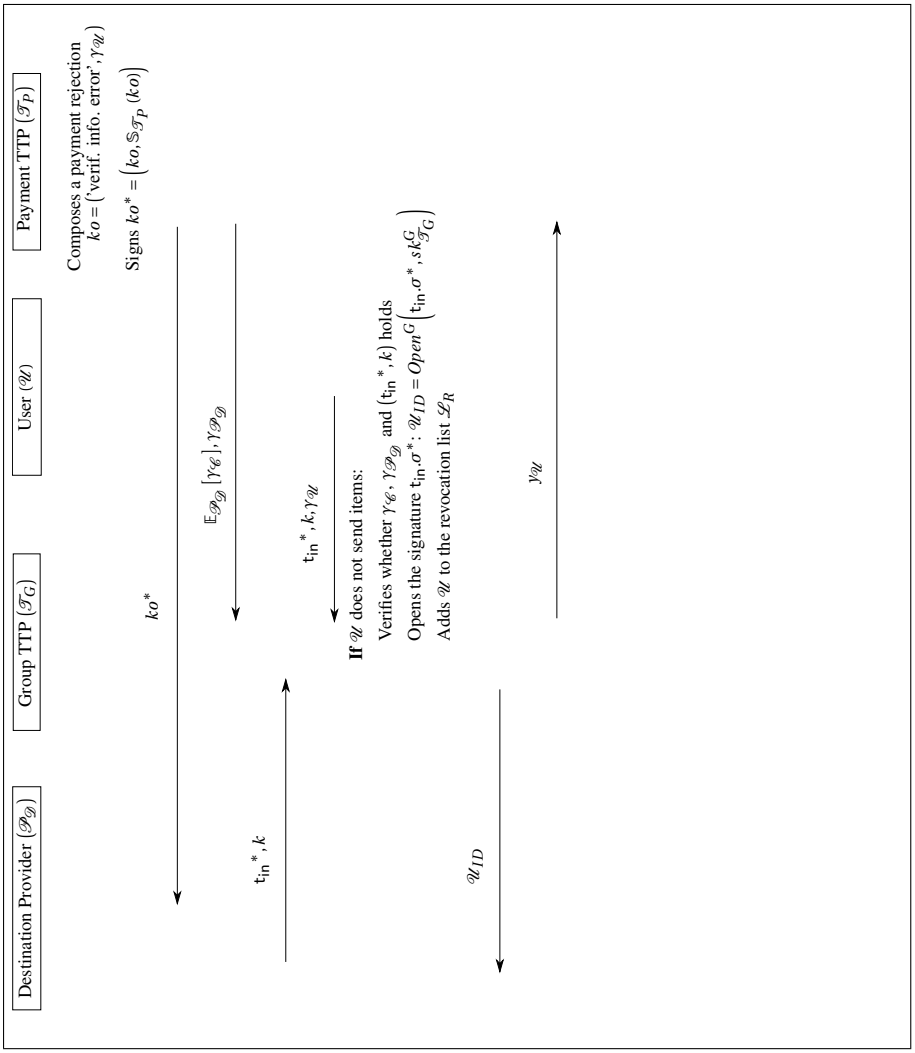


Figure 5.9: `PClaim2` protocol flow.

5.6 Apply General AFC Scheme to Time and Distance-Based Scenarios

The AFC scheme described in §5.4 is a general scheme that ca be applied to either time-based or distance-based scenarios with only minor changes. In fact, the function $\mathscr{F}$ to calculate the fare to be paid by $\mathscr{U}$ should be adapted depending on the scenario.

In case the scenario is based on time, the fare should depend on the time $\mathscr{U}$ has used the service. So, the fare is computed as the difference between the time of exit (τ_2) and the time of entrance (τ_1), as follows:

$$a = \mathscr{F}(\tau_2, \tau_1) = f_t(\tau_2 - \tau_1),$$

where f_t is some multiplicative constant (e.g. the prize of each minute using the service).

Instead, if the considered service is based on distance, the fare has to be proportional to the covered distance by $\mathscr{U}$ between entrance (Ps) and destination (Pd) stations, as follows:

$$a = \mathscr{F}(\mathsf{Pd}, \mathsf{Ps}) = f_d(\mathsf{Pd} - \mathsf{Ps}),$$

where f_d is another multiplicative constant (e.g. the prize per kilometer of the itinerary).

Table 5.5: Information included in the entrance ticket ($\mathsf{t_{in}}^*$) in case it has to be applied to the distance-based scenario with common entrance and exit stations.

Notation	Description
Sn	Serial number computed by $\mathscr{P}_{\mathscr{S}}$
Ps	Identifier of the entrance station (source provider)
τ_1	Time at the entrance
σ^*	Commitment signed by $\mathscr{U}$
$\mathbb{S}_{\mathscr{P}_{\mathscr{S}}}(\mathsf{t_{in}})$	Digital signature by $\mathscr{P}_{\mathscr{S}}$ on the above parameters
Added parameters to become a distance-based entrance ticket	
τ_v	The ticket will be valid up to this timestamp
ξ	Direction of the journey

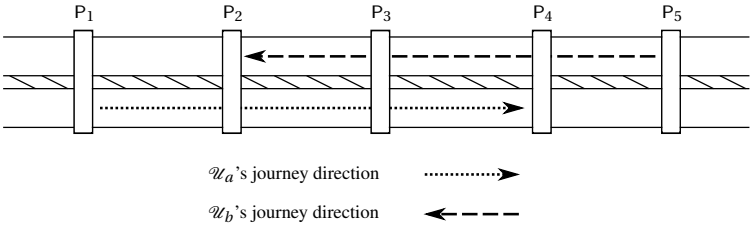
As a difference from time-based AFC system in which it is only required to adapt the fare calculation function, if it is considered a distance-based scenario, the entrance ticket must include additional information (see Table 5.5). This additional information is a simple timestamp (τ_v) and the direction of the journey (ξ). On one

hand, the τ_v parameter defines the time up to the user can remain in the system, i.e. the ticket will be valid up to this time mark. On the other hand, ξ is useful to know where $\mathscr{U}$ is going. Both parameters should be verified together with τ_1 when $\mathscr{U}$ runs the `SysExit` protocol.

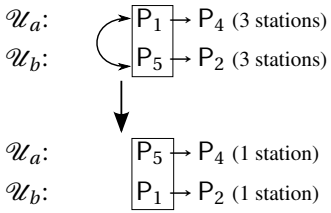
The reason to add the journey's direction ξ is to avoid colluding attacks performed by malicious users, as we explain in the next Section.

5.6.1 Colluding Attack: Exchange of Tickets between Malicious Users

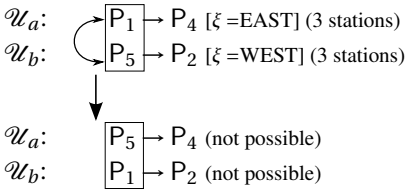
Figure 5.10(a) represents the scenario with separated directions and checking points (entrance and exit stations) depending on the direction of movement.



(a) Scenario with separated directions and stations. Two users moving opposite directions.



(b) Successful colluding attack without ξ parameter in the entrance ticket.



(c) The attack is no longer valid when ξ is included in the entrance ticket.

Figure 5.10: AFC scenario with separated directions and stations, with and without ξ parameter setting the direction of movement.

Consider two colluding users ($\mathscr{U}_a$ and $\mathscr{U}_b$) who exchange their respective entrance tickets $\left(\mathsf{t}_{\mathsf{in}_a}^{*} \text{ and } \mathsf{t}_{\mathsf{in}_b}^{*}\right)$. Therefore, as explained in Figure 5.10(b), $\mathscr{U}_a$ enters the system through P_1 and leaves it at P_4. Similarly, $\mathscr{U}_b$ goes the system in P_5 and leaves it at P_2. Then, if both users exchange their respective entrance tickets, the result is that they only will be charged for the distance between one single station instead to be charged by their real itinerary composed by three stations. As Figure 5.10(c) proves, if the direction parameter (ξ) is set in the entrance ticket, this colluding attack is no more successful because it is not possible to leave the service

by an earlier station than the entrance station (if separated directions and station are considered).

However, in case the scenario does not hold the requirement of separated directions and stations, the proposal continues to be vulnerable to the colluding attack explained above. Figure 5.11 shows the scenario in which users are allowed to choose the direction of their journey after to receive the entrance ticket. Therefore, the ξ parameter is no longer useful to avoid the colluding attack. As a result, we need to add more security measures to protect providers in a distance-based AFC system to be cheated by malicious and colluding users. The solution is to enforce a link between entrance and exit procedures in a way that the system should be able to check whether the user who has received the entrance ticket is the same one at the moment of leaving the service. In the next section we give a method to improve the service and avoid the colluding attack.

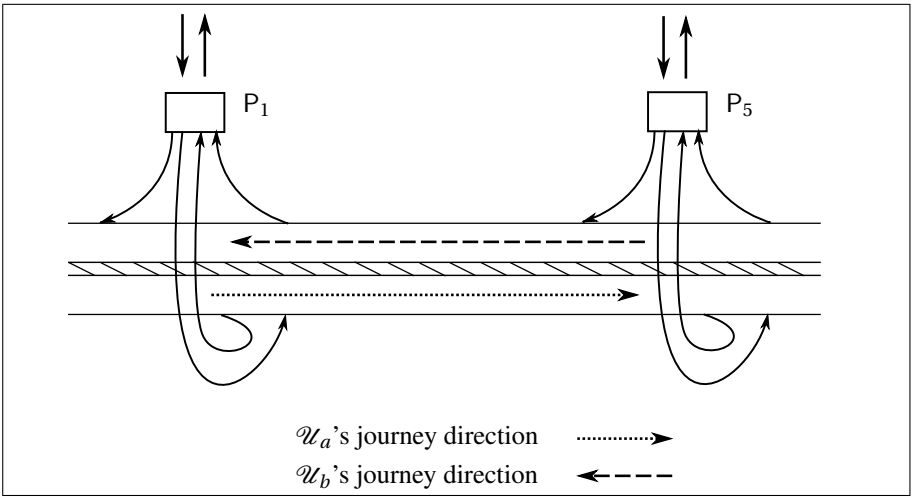


Figure 5.11: AFC scenario without separated directions and stations. Users choose the direction after entering the service.

5.7 Enhanced Distance-Based AFC

In this Section we provide an enhanced distance-based AFC scheme to avoid the aforementioned colluding attack. As to achieve a link to check whether the same user who has received the entrance ticket is in fact the user who are leaving the service, we propose a method to link two group signatures.

5.7.1 Linkability of Signatures

The BBS group signature reviewed in §2.3.3 defines a scheme to generate anonymous and indistinguishable signatures, even they are computed by the same user, thanks to randomization techniques. Therefore, the probability by an adversary to link two group signatures computed by the same user (even colluding with $\mathcal{T}_G$) is negligible (in $\mathtt{K}$), so that scheme generates unlinkable signatures. Thus, new processes should be considered to link two group signatures.

There are in the literature some proposals defining linkable group signatures (with different levels of linkability). In a linkable group signature scheme, or also called *link-but-not-trace* group signatures [99–101], signatures from the same signer can be linked, but its anonymity remains.

However, and with the consideration that only two group signatures has to be linked (we have to continue to avoid the linkability of group signatures made in different journeys, as required by the *Definition 5.2.6*), we propose in [102] two new procedures to add a simple linkability feature to the original BBS group signature scheme. We describe these two new procedures using the same notation already used to describe the full BBS group signature scheme in §2.3.3:

SignLink^G. The signer uses the same random pair (α, β) randomly picked up in a previous run of the *Sign*^{G} algorithm (see *Sign*^{G} specification in §2.3.3). This way, the same triple $(T_1, T_2, T_3) \leftarrow \left(u^{\alpha}, v^{\beta}, A_i h^{\alpha+\beta}\right)$ is obtained (or restored from storage). The procedure for a user i with private key (A_i, x_i) differs from the original *Sign*^{G} by the following (note that highlight lines allow being precomputable along user's journey before to arrive the destination station):

1. Picks new $r_{\alpha}^{'}, r_{\beta}^{'}, r_{x_i}^{'}, r_{\delta_1}^{'}, r_{\delta_2}^{'} \overset{R}{\leftarrow} \mathbb{Z}_p$
2. Calculate new $R_1^{'}, R_2^{'}, R_3^{'}, R_4^{'}, R_5^{'}$ but uses previous T_1, T_2, T_3, δ_1 and δ_2:

$$R_1^{'} \leftarrow u^{r_{\alpha}^{'}} \qquad R_2^{'} \leftarrow v^{r_{\beta}^{'}}$$

$$R_3^{'} \leftarrow e(T_3, g_2)^{r_{x_i}^{'}} \cdot e(h, \omega)^{-r_{\alpha}^{'} - r_{\beta}^{'}} \cdot e(h, g_2)^{-r_{\delta_1}^{'} - r_{\delta_2}^{'}}$$

$$R_4^{'} \leftarrow T_1^{r_{x_i}^{'}} \cdot u^{-r_{\delta_1}^{'}} \qquad R_5^{'} \leftarrow T_2^{r_{x_i}^{'}} \cdot v^{-r_{\delta_2}^{'}}$$

3. Using a hash function $\mathscr{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p$, the user i obtains the challenge $c^{'}$

$$c' = \mathcal{H}\left(M', T_1, T_2, T_3, R_1', R_2', R_3', R_4', R_5'\right) \in \mathbb{Z}_p$$

4. Using the challenge c' the user i obtains $s_\alpha', s_\beta', s_{x_i}', s_{\delta_1}', s_{\delta_2}'$:

$$s_\alpha' = r_\alpha' + c'\alpha \quad s_\beta' = r_\beta' + c'\beta \quad s_{x_i}' = r_{x_i}' + c' x_i$$
$$s_{\delta_1}' = r_{\delta_1}' + c'\delta_1 \quad s_{\delta_2}' = r_{\delta_2}' + c'\delta_2$$

5. The signature of knowledge is $\sigma' = \left(T_1, T_2, T_3, c', s_\alpha', s_\beta', s_{x_i}', s_{\delta_1}', s_{\delta_2}'\right)$, where $T_1, T_2, T_3 \in \mathcal{G}_1$ while $c', s_\alpha', s_\beta', s_{x_i}', s_{\delta_1}', s_{\delta_2}' \in \mathbb{Z}_p$ and are fresh.

VerifyLink^G. This algorithm outputs either *true* or *false* depending on whether the provided signatures σ and σ' have been computed by the same signer who has computed the triple (T_1, T_2, T_3). Note that this algorithm does not substitute the original *Verify*^{G}. Instead, it complements that algorithm because this new procedure only checks whether signature has been computed by the same user (verifier should execute *Verify*^{G} algorithm before *VerifyLink*^{G}).

1. Checks whether $\left(T_1 \stackrel{?}{=} T_1',\ T_2 \stackrel{?}{=} T_2' \text{ and } T_3 \stackrel{?}{=} T_3'\right)$.

If all verifications hold, verifier is convinced that σ and σ' has been computed by the same user i, thus *true* is output. Otherwise, the algorithm outputs *false*.

Note that this method to link two group signature only has to be executed once by journey in order to avoid linkability of different sessions (see *Definition 5.2.6*).

5.7.2 AFC System Specification

Armed with the ability to verify whether two group signatures has been computed by the same user $\mathcal{U}$ by means of the use of *SignLink*^{G} and *VerifyLink*^{G} algorithms described above, we provide the description of changes to apply to the general AFC scheme described in §5.5 and modified in §5.6 as to become it into a distance-based fare for the scenario in which directions and stations are not separated.

The major modification is that the k parameter and all steps in which it is involved have been replaced by the *SignLink*^{G} algorithm to compute a second and linkable group signature to the first one computed in the entrance. Similarly, the *VerifyLink*^{G} algorithm has to be added to the list of actions run by the destination provider ($\mathcal{P}_\mathcal{D}$), the payment manager ($\mathcal{T}_P$) and the group manager ($\mathcal{T}_G$). This way,

the scheme's security is increased but also the computational cost, due to the inclusion of a second group signature at the moment $\mathscr{U}$ leaves the AFC service.

The new `SysEntrance` protocol with aforementioned changes is depicted in Figure 5.12. As pointed out before, the only difference is that $\mathscr{U}$ does not generate either k or $\mathscr{H}(k)$, so she group signs only the tuple $(s_1, \delta_{\mathscr{U}})$ instead of the triple $(s_1, \delta_{\mathscr{U}}, \mathscr{H}(k))$ as before.

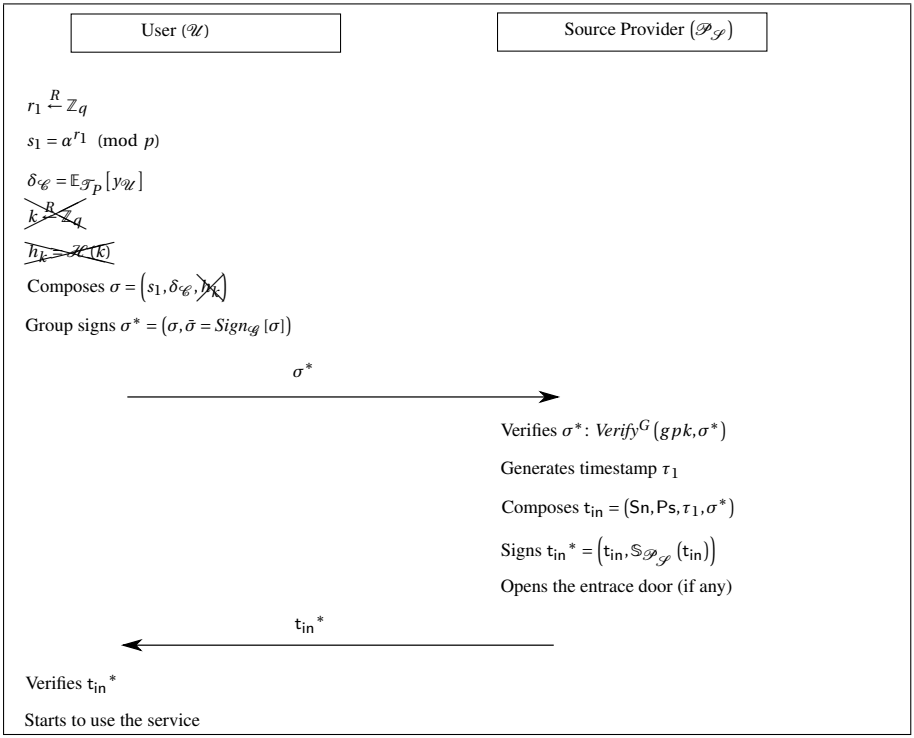


Figure 5.12: Enhanced `SysEntrance` protocol flow for the new scenario.

Figure 5.13 represents the enhanced `SysExit` protocol. Now, $\mathscr{P}_{\mathscr{D}}$ runs a previous step in which he sends $\mathscr{U}$ a random element Φ. Then, $\mathscr{U}$ has to group sign Φ (obtaining Φ^*) as to prove that she is actually the same user who has computed the group signature in the entrance by means of the $SignLink^G$ algorithm. $\mathscr{P}_{\mathscr{D}}$ should verify this fact by using the $VerifyLink^G$ algorithm instead of verify correctness of k.

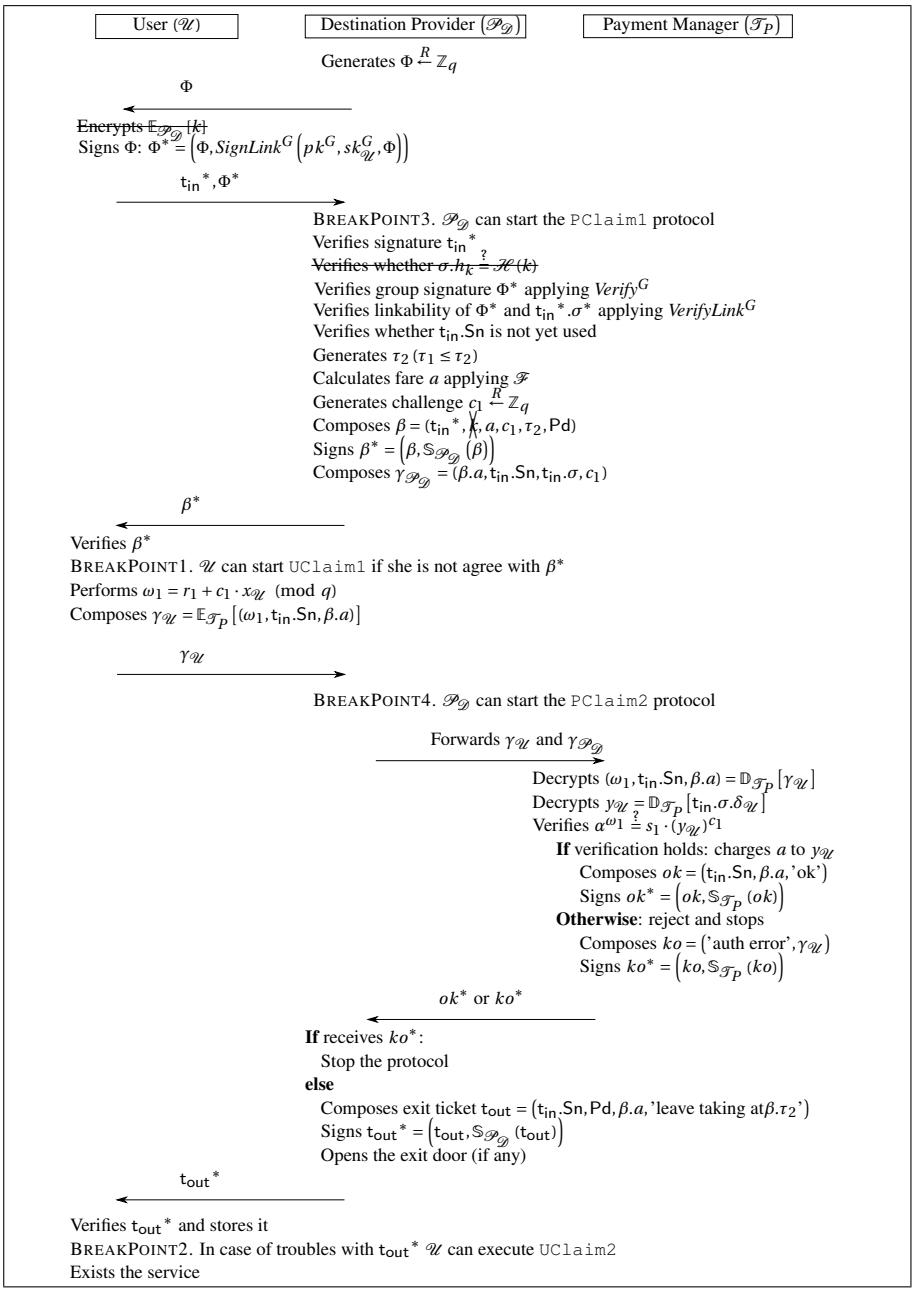


Figure 5.13: Enhanced SysExit protocol flow for the new scenario.

User and provider claims (`UClaim1`, `UClaim2`, `PClaim1` and `PClaim2`) have been also updated to include similar modifications. Thus, in the first step of both `UClaim1` and `UClaim2` protocols, $\mathscr{U}$ sends Φ^* instead of k. Then, $\mathscr{T}_P$ should verify the group signature Φ^* using $Verify^G$ and the linkability with the first group signature by applying $VerifyLink^G$. Finally, the same modification is applied to `PClaim1` and `PClaim2` protocols, but this time is $\mathscr{T}_G$ who verifies the linkability of both group signatures.

5.8 Conclusions

In this Chapter we have reviewed a previous AFC scheme originally appeared in [34] to charge users based on the distance parameter. Honest users remain anonymous but malicious users could suffer the revocation of their anonymity, thanks to the use of a group signature scheme. Due to the fact that the few differences between charge users based on distance or time, we have decided to generalize that proposal to fit both fare calculations. This way, we have provided a general solution to the AFC scenario. As result of a joint work with researchers from *Universitat Rovira i Virgili (URV)*, we have improved the scheme to avoid a colluding attack performed by users who exchange their entrance tickets in order to receive an smaller fare from service provider. The solution avoids that colluding attack in two different scenarios: either when checking stations (entrance and exit stations) are separated by direction or not. The former case is easy to resolve as it only requires to set the direction parameter in entrance tickets. Instead, the latter scenario requires to add more security measures at the moment users leave the service. In fact, the provided solution conveys a second group signature to check whether the user who had computed the first group signature at the moment to enter the service is actually the same as who leaves the service. In addition, we have added a complete security model with definitions for each security requirement that an AFC scheme should achieve. Finally, we address the reader to Chapter 7 (Section §7.3) where we prove in a formal language that the AFC scheme in fact achieves required security properties stated by the security model. Aside, we have implemented these schemes and measured the performance of both time-based and distance-based AFC schemes in Chapter 12 to prove that them are also efficient to be used by mobile devices.

Part III

Security Analysis of Privacy-Protecting Proposals

CHAPTER 6

CONTRIBUTION TO THE AUTOMATIC FORMAL ANALYSIS OF SECURE PROTOCOLS

The analysis of secure protocols is a challenging and often a costly step once researchers want to provide proofs to demonstrate that the released scheme is in fact secure. Despite there are different ways to conduct a security analysis, the formal analysis may provide stronger proofs than other means of security analysis. Due to the fact that a formal security analysis could be a tedious job, some automatic methods have been presented along the literature to facilitate this job. One of them is the CPN language, derived from the original Petri Net language which has been proved a useful tool to analyze concurrent and distributed protocols. In this Chapter we provide some interesting insights about the use of CPNs and how they should be applied to formally verify security properties. Moreover, we conduct a formal analysis of a contract signing scheme and thanks to the use of the proposed CPN method, we are ready to improve the security of the scheme.

6.1 Introduction

The security analysis of secure protocols, such as e-payment proposals, ticketing systems and even cryptographic primitives is a challenging topic due to the fact that

it is a tedious and repetitive job. There are different ways to perform that security analysis depending on their level of formalism. Typically, this field is divided into two groups: informal review and formal analysis of security properties. The former proves security properties in an informal way, meaning that the analysis is not intended to either cover all the cases or to run deep on cryptographic and mathematical assumptions. Instead, the latter uses mathematical analysis, often based on the underlying cryptographic assumptions, in order to formally proof the accomplishment of the analyzed security properties. It can be argued that the formal analysis is better to proof the security of protocols than the informal verification, but not always it is easy to conduct a formal analysis. Created as to resolve that issue, some automatic formal analysis methods have been published in the literature [35–37] and it constitutes an important research topic. In this Chapter, we focus our efforts on analyze the applicability of a formal methodology based on CPN [35] to automatically verify secure protocols to prove its resistance to attacks and vulnerabilities. CPNs are a formal language that extend the original Petri Net language to model complex and distributed systems in order to be analyzed and verified. Recently, CPN methodology has attracted attention to become a formal method to analyze some security properties from secure protocols.

The proposed CPN methodology that we explain in this Chapter is based on two previous works in which authors formally analyze two e-commerce schemes. The first one is the work due to Katsaros [38] who analyzed some properties of the Netbill micropayment scheme. The second work is due to Sornkhom and Permpoontanalarp [39] who have applied CPN to formally analyze the security of the ECS1 contract signing scheme due to Micali et al. [103]. Authors in [39] used the CPN methodology to prove that the ECS1 scheme was vulnerable to three different attacks proposed by Bao et al. in a previous work [104]. As expected, authors had found these three attacks but, surprisingly, the automatic model was able to found two new attacks not yet discovered to ECS1 scheme.

Armed with CPN, we want to describe and adapt the proposed methodology to apply it to conduct a formal security verification of the Ferrer-Payeras-Huguet (FPH) protocol [40] against the three attacks previously defined in [104] and two other vulnerabilities described in [39]. As a result, we have successfully created a new CPN model adapted to the FPH protocol. This model allows us to automatically analyze the FPH protocol and to conclude that it is not vulnerable to the previously mentioned attacks. In addition, we have found two new conflictive situations not yet discovered by the authors of FPH that affects the fairness of the scheme.

We discuss that CPN could be very useful to analyze the fairness property of security protocols that include some kind of message exchange, with a limited and static number of involved parties. It can be also useful to verify the accomplishment of other security requirements such as the non-repudiation property or the effectiveness. Moreover, CPN could be used to verify the correctness property and even to

perform some sort of theoretical performance analysis. Although it seems a promising method, its applicability may depend on the scheme to analyze. So, we must decide whether the method is useful after a careful study of the complexity of the scheme to verify.

We have organized the chapter as follows. Section 6.2 gives a brief introduction to different languages and some automatic model checkers for modeling processes. In Section 6.3 we define what is the Petri Net language while in Section 6.4 we present some key concepts of CPNs. Section 6.5 proposes how to use CPN language to model a secure protocol. In Section 6.6 we provide a short introduction to contract signing schemes together with a description of the FPH and the ECS1 contract signing schemes. Section 6.7 includes the description of proposed model made by using CPNs. Section 6.8 presents the formal fairness analysis of the FPH scheme and the results obtained by using different scenario configurations verifying whether FPH is vulnerable or not to the previously defined attacks. Moreover, in this Section we describe two situations found by the CPN model, previously undiscovered by FPH authors, where the scheme could become unfair. As to resolve these fairness problem, we propose a solution that makes the FPH scheme still more secure. As opposed to the successful use of the CPN methodology in the two-party FPH scheme version, in Section 6.9 we show that not always CPN is effective due to the fact that it has some limitations to model some kind of protocols, such as the multi-party version of the FPH scheme. Finally, Section 6.10 includes the conclusions about using the CPN method to formally verify protocols giving advises about when it is useful and where not based on the acquired experience.

6.2 Process Modeling Languages and Model Checkers

Within the field of modeling languages there are some methodologies created to model complex business processes in order to be analyzed and afterwards improved. The creation of models for business and industrial processes is a critical step in their design, planning and their corresponding implementation to detect flaws by analysis and inspection.

One of these languages is the Unified Modeling Language (UML), standardized as the ISO/IEC 19501:2005 recommendation [105]. It is defined as a general-purpose modeling language for visualizing, specifying, constructing and documenting processes mainly intended to be applied to the field of software engineering. It is a graphical language with plenty of software tools to create, manage and interpret UML diagrams, such as IBM Rational Rose, ArgoUML, Umbrello, etc. UML is very useful as a model tool to plan and describe business models during their software development. However, UML does not implement means to automatically check the correctness of the models in any way.

A different method with a mathematical background is the *applied π-calculus*, derived from the original *π-calculus* which is a way to formally model concurrent systems bringing a formal tool for the description of the interaction, communication and synchronization between processes. The *applied π-calculus* defines a formal notation to describe and to reason about cryptographic protocols and there exist some experimental verification tools based on this method, such as ProVerif [106] and Cryptographic Protocol Type Checker (CRYPTYC) [107]. Unfortunately, these tools seem to be more specific and limited to verify some concrete protocols and properties instead being able to implement and verify a large range of protocols and systems.

Until now, we have talked about modeling languages capable to model systems and processes, but not to simulate and automatically verify their correctness or formally analyze the accomplishment of certain properties. In this field, we have found three main projects or languages: the Automated Validation of Internet Security Protocols and Applications (AVISPA) project [36], the Simple Promela Interpreter (SPIN) tool [37] and the Petri Net language.

The AVISPA project was created to model and analyze large-scale Internet security-sensitive protocols and applications. AVISPA defines a complex language to define how protocols should be modeled. Then, armed with this definition, a software tool performs an automatic simulation and the results can be used to analyze the behavior of the analyzed protocol or application. Although it seemed a promising automatic modeling and verification tool, it was given up by developers, so that it is no more developed.

Similar to the aims of AVISPA, we have found the SPIN tool. SPIN is a tool to model and verify the correctness of distributed software models in an automated way. Protocols to be verified has to be described on terms of the PROcess ModEling LAnguage (PROMELA) [108] which is another special language that supports modeling of distributed algorithms. The SPIN tool can be used as a simulator either with or without human interaction. But as AVISPA does, SPIN fails on having a powerful graphical representation and to model concurrent systems [38].

As opposed to previous projects and tools, the Petri Net language is defined based on a strong mathematical backend and a graphical language. We define the Petri Net language and some interesting extensions to them in the following Section.

6.3 Petri Net Definition

A Petri Net [109] is a mathematical and graphical discrete-event modeling language for building models of concurrent and distributed systems and analyze their properties. As explained above, there are other modeling languages in the literature, but Petri Nets are the only methodology that is supported by a mathematical theory that

defines both their construction and their analysis. Informally speaking, a Petri Net is defined mainly by places, transitions, arcs and tokens:

- *Place*. It can store a discrete number of tokens.
- *Transition*. It is the action that changes the contents of a place. When a transition fires, the global state of the system changes.
- *Arc*. It connects places and transitions. A token is transferred between a place and a transition (or from a transition to a place) through an input arc (or an output arc).
- *Token*. It is an element that is transferred along the model. It can enable transitions whenever there are sufficient tokens at input arcs.

Once a transitions fires, the global state of the system is changed and configured by a new distribution of tokens over places. Each of these states are called *markings* and they can be seen as a photo of the token distribution along the model.

Formally, a Petri Net is a 4-tuple $(\mathscr{P},\mathscr{T},\mathscr{W},\mathscr{M}_0)$, where:

- $\mathscr{P}$ is a finite set of *places*.
- $\mathscr{T}$ is a finite set of *transitions*.
- $\mathscr{P}$ and $\mathscr{T}$ are disjoint, i.e. no object can be both a place and a transition.
- $\mathscr{W} : (\mathscr{P} \times \mathscr{T}) \cup (\mathscr{T} \times \mathscr{P}) \rightarrow \mathbb{N}$ is a multiset of arcs connecting a place with a transition but never connecting two places or to transitions.
- $\mathscr{M}$ is a *marking* which is a multiset of its places.
- $\mathscr{M}_0$ is the *initial marking* of a Petri Net.

There are many extensions to the original Petri Net language that adds or modifies some properties of them. Some of these extensions are the *Timed* Petri Nets (adding time notion to the original language), *Hierarchical* Petri Nets (adding hierarchy to support different levels of abstraction) and *Colored* Petri Nets (adding typed tokens).

6.4 Colored Petri Nets

CPNs [35] are an extension of the original Petri Net language adding some functionalities to it. In a Petri Net, tokens are indistinguishable whereas in CPN, each token has a concrete type, called *color*. Similarly to original Petri Nets, CPNs could be also useful to analyze distributed applications and services such as communication protocols, data networks, embedded systems, business processes, and so on.

CPNTools [110, 111] is the most popular tool to deal with CPN. CPNTools presents a simple but powerful graphical interface that allows us to create, edit, model, simulate and analyze complex and distributed systems using CPN. The application is bounded with a functional programming language, called standard ML [111] which is used to provide the primitives to define colors, describe data and its manipulation, allowing the creation of customized models. An interactive simulation provides a way to check whether a CPN model is properly implemented and whether it works as expected. Moreover, an automatic simulation tool is provided to execute the CPN model as fast as possible, without detailed human interaction and inspection. After a complete simulation, we can analyze all the states of the system using the *state space* tool provided by CPNTools. Therefore, we can extract the complete behavior through the analysis of the computed state space.

In addition to the elements that define a Petri Net such as places, transitions, arcs and tokens, CPN also add the following components to the language:

- *Color set*. Tokens that move through places and transitions have a value, called a color.
- *Variable*. It is an identifier whose value can be changed during the execution of the model.
- *Function*. CPNTools introduce some functions that can be used by a model to deal with color sets. Functions follow the syntax of the ML language and it is also possible to define and develop custom and powerful functions either to change the behavior of the model (e.g. a transition can only be fired under a condition specified by a function) or to inspect the state space (e.g. a query function traversing the complete state space looking for whether some tokens are present or not in some states, in order to also know their value at any time of the simulation).

Figure 6.1 shows a simple CPN example, where all of the above elements are depicted. There are two places (with names *place1* and *place2* and marked with an integer color), two transitions (called *t1* and *t2*) and four arcs connecting the places with the transitions. Each arc has a variable which will be filled by a token to transfer it along the model. Note that only one transition is enabled to be fired due

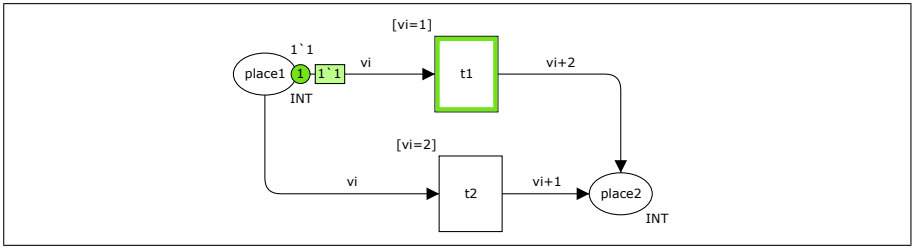


Figure 6.1: A Colored Petri Net example built using CPNTools.

to a condition (enclosed by square brackets). The initial marking is composed by a single token filling the *place1* with value *1* and an empty place *place2*. Finally, a simple function relied at the output arcs, causes that the input token value is increased by an integer value.

As stated before, both Petri Nets and CPN models are formal methods backed by a solid mathematical foundation, so they could be used to formally verify properties of distributed systems and communication protocols [35, 109].

6.5 General Assumptions and Methodology

In order to use the CPN language to model a secure protocol, such as an e-commerce system, we have to define some initial assumptions:

- The parties involved within the protocol must be defined as well as their role (i.e. defining if each party is either honest or malicious).
- It must be supposed that all the cryptographic algorithms (such as signature and encryption algorithms) are implicitly secure, since the model cannot check the security of these crytographic blocks (in order to do that, the model should implement the complete behavior of each used cryptographic tool and, of course, it is unfeasible and not really useful for the analysis we want to provide).
- The initial marking must be set. It means that each considered party must have a set of predefined values that they own in their corresponding databases, such as their unique identifier within the model, their cryptographic keys, some invariable values, etc. Moreover, it is supposed that each party already has the public keys of the other considered entities just before to start the model execution. In addition to the contents of parties' databases, initial markings of places which controls the right synchronization of the model must be specified.

Besides to these initial general assumptions, we have defined a process starting from the creation of the model up to the analysis of the results given in by the CPNTools:

- First of all, we should build the model following these basic steps (based on a bottom-up process):
  - To declare color sets to represent all the messages and the elements of the protocol.
  - To create a top-level net to model the parties.
  - To create an entity-level net to model the behavior of each party.
  - To create a process-level net for each entity to define all the actions performed by each entity.
  - To declare several variables within the model to store results and transfer tokens.
  - To declare a number of functions to include conditions and token transformations.
- Once the model is created, we have to set up the initial marking for each entity.
- After that, we can generate the full state space, using the corresponding tool from CPNTools (see Figure 6.2), obtaining all the available markings and combinations of them.
- Before analyzing the results, we have to create some query functions to look for markings that meet certain conditions.
- Finally, using these functions, we can extract a list of dead markings (markings where the execution of the protocol is already concluded) where probably an unfair state has taken place. If some unfair markings are detected, we can graphically explore the path from the initial marking to that dead marking to investigate the exact attack trace which has conducted to the unfair marking. As a result, we obtain a tree where each leaf represents a single marking and each of them are connected through branches. Each branch describes which is the transition fired to pass from the parent leaf to the child leaf (see Figure 6.3).

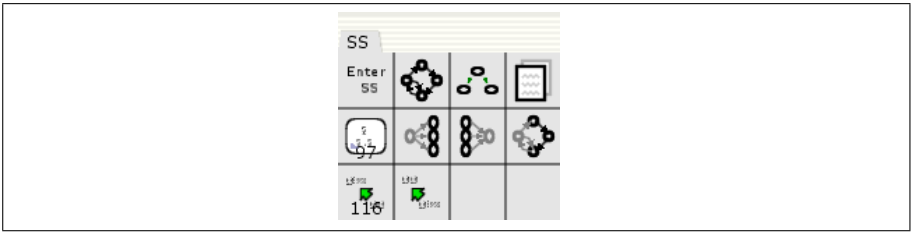


Figure 6.2: State space tool from the CPNTools software.

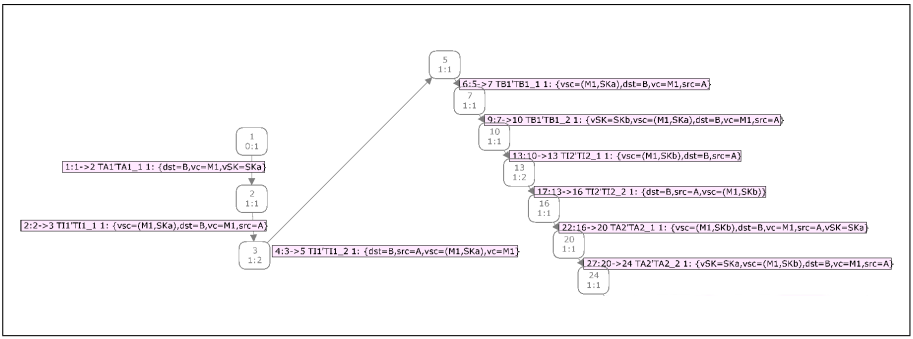


Figure 6.3: Marking tree generated by using the CPNTools.

6.6 Case Example I: FPH Contract Signing Scheme

In order to test whether the CPN methodology is useful to formally verify the fairness property in a fair exchange protocol, we have selected the FPH contract signing protocol as a case example. Before to model and analyze the scheme, we provide a brief description about the main features of the contract signing schemes as well as a description of the FPH and ECS1 schemes.

6.6.1 Overview on Contract Signing Schemes

Contract signing procedures, certified electronic mail or electronic purchases are good examples of fair exchange protocols. A fair exchange of values always provides an equal treatment to all users, and, at the end of the execution of the exchange, all parties have the element that wished to obtain, or the exchange has not been solved successfully (in this case, nobody has the expected element). These protocols make use of non-repudiation services, so they have to produce evidences to guarantee non-repudiation services. In case of dispute, an arbiter has to be able to evaluate the evidences and take a decision in favor of one party without any ambiguity. Contract signing schemes allow the signature of a previously accorded contract by either two

(two-party) or more signers (multi-party). The fair exchange protocol ensures that at the end of the exchange all the signers have the signed contract or none of them have it. Fair exchange protocols often use a TTP to help users to successfully complete the exchange. Several electronic contract signing schemes have been presented, with TTPs involved in different degrees. Among them there are a few proposals where the exchange can be finished in only three steps. The ECS1 scheme due to Micali et al. [103] and the FPH scheme due to Ferrer et al. are efficient systems involving an exchange protocol of three messages. Both schemes differ in the resolution protocol as well as in the elements exchanged in the three steps. However, they have some common aspects like the use of an off-line TTP, called *optimistic* approach. This concept of *optimistic protocol* was introduced in [74] by Asokan et al. In an optimistic fair exchange protocol the TTP only intervenes in case of problems to guarantee the fairness of the exchange.

6.6.2 Ideal Features of a Contract Signing Scheme

Practical solutions for contract signing require the existence and the possible involvement of a TTP. In order to obtain efficiency, the following three objectives are usually desired:

- Reduce the involvement of the TTP.
- Reduce the number of messages to be exchanged.
- The possible implication of the TTP should not require expensive operations, neither the storage of high volumes of information.

The first aim has been achieved in some proposals, the so called optimistic solutions [67, 74, 112–114], in which the TTP is not involved in every contract signing run. Regarding to the number of messages to exchange, the authors of [112] state that three is the minimum number of messages to exchange to complete successfully a contract signing scheme. Schemes for contract signing have to provide evidences to the parties to prove, at the end of the exchange whether the contract is really signed or not, together with the associated terms and conditions. Some additional properties must be achieved in optimistic protocols [67, 114]:

EFFECTIVENESS. If parties behave correctly, the TTP will not be involved in the contract signing.

FAIRNESS. No party will be in advantageous situation at any stage of a protocol run.

TIMELINESS. The parties can decide when to finish a protocol run.

NON-REPUDIATION. The involved parties cannot deny their actions performed during the contract signing.

VERIFIABILITY OF THE THIRD PARTY. If the TTP misbehaves, all parties will be able to prove the incorrect behavior.

6.6.3 Description of the FPH Contract Signing Scheme

The FPH scheme is an optimistic contract signing protocol that achieves all of the above security requirements. This scheme involves three parties: the originator *Alice* (**A**) and the recipient *Bob* (**B**) being two users who want to digitally sign an already agreed plaintext contract (C), and the TTP (**T**), who assures the fairness of the protocol in case of problems during the exchange.

The scheme is defined through three protocols. The first one is the protocol between **A** and **B** that is called *exchange protocol* and where **T** is not involved at all. If anything goes wrong with the *exchange protocol*, **A** can invoke the *cancel protocol* and **B** can run the *finish protocol*. Regarding to the communication channels, on one hand, the channel used between both signers (during the *exchange protocol*) is an unreliable channel, so it cannot be assumed that the messages sent through this channel arrive to their recipient. On the other hand, the channel between each signer and **T** (both *cancel* and *finish* protocol) is a resilient channel, i.e. messages will eventually arrive to their recipient, but the time of the arrival cannot be predicted. Both **A** and **B** exchange NR evidences directly, i.e. without the involvement of **T** through the use of the *exchange protocol* (see Table 6.2). Only in case they cannot get the expected items from the counterpart, **T** will be invoked, by initiating either the *cancel protocol* (see Table 6.3) or the *finish protocol* (see Table 6.4) depending on which is the party who wants to resolve the unexpected situation. The notation and the elements used along the protocol description are depicted in Table 6.1.

If the protocol run is completed, the originator **A** holds a NR evidence ($h_{\mathbf{B}}$), and the recipient **B** holds a pair of NR evidences ($h_{\mathbf{A}}, ACK_{\mathbf{A}}$). So the scheme meets the *effectiveness* requirement. If it is not the case, either **A** or **B**, or both, need to resolve the unfair situation starting either the *cancel* or the *finish protocol*, respectively, so that the situation returns to a fair position.

If **A** asserts (**A** could be trying to cheat or being in a wrong conception of the exchange state) that she has not yet received the second message from **B**, **A** could start the *cancel protocol* (Table 6.3) in order to ask for a signing cancellation. During the *cancel protocol*, **T** verifies the correctness of the information given by **A**. If the information received is not correct or incomplete, **T** sends an error message to **A**. Otherwise, it proceed in one of the two possible ways. If the variable *finished* is *true*, it means that **B** had previously contacted with **T** (see paragraph below), and **T**

Table 6.1: The notation used to describe the FPH scheme.

Element	Description
X, Y	Concatenation of two messages X and Y
$\mathscr{H}(X)$	Collision-resistant one-way hash function on the message X
$\mathbb{S}_{\mathscr{P}}(X)$	Signature on the message X with the private key of party $\mathscr{P}$
$\mathscr{P}_i \rightarrow \mathscr{P}_j : X$	$\mathscr{P}_i$ sends a message X to $\mathscr{P}_j$
$\mathtt{C} = (\mathbf{A}, \mathbf{B}, \mathscr{M})$	The contract $\mathtt{C}$ specifies the originator ($\mathbf{A}$), the recipient ($\mathbf{B}$) and the text of the contract ($\mathscr{M}$)
$h_{\mathbf{A}} = \mathbb{S}_{\mathbf{A}}(\mathtt{C})$	Signature of **A** on the contract $\mathtt{C}$. It is the first part of the non-repudiation evidence for **B**
$h_{\mathbf{B}} = \mathbb{S}_{\mathbf{B}}(\mathtt{C})$	Signature of **B** on the message $\mathtt{C}$. It is the non-repudiation evidence for **A**
$ACK_{\mathbf{A}} = \mathbb{S}_{\mathbf{A}}(h_{\mathbf{B}})$	Signature of **A** on $h_{\mathbf{B}}$; it is the acknowledgement that states that **A** knows that the contract is signed. It is the second part of the non-repudiation evidence for **B**
$ACK_{\mathbf{T}} = \mathbb{S}_{\mathbf{T}}(h_{\mathbf{B}})$	Signature of **T** on $h_{\mathbf{B}}$; this is the equivalent acknowledgement to that **A** should have sent
$h_{\mathbf{AT}} = \mathbb{S}_{\mathbf{A}}(\mathscr{H}(\mathtt{C}), h_{\mathbf{A}})$	Evidence that **A** has requested **T**'s intervention
$h_{\mathbf{BT}} = \mathbb{S}_{\mathbf{B}}(\mathscr{H}(\mathtt{C}), h_{\mathbf{A}}, h_{\mathbf{B}})$	Evidence that **B** has requested **T**'s intervention
$h'_{\mathbf{B}} = \mathbb{S}_{\mathbf{T}}(h_{\mathbf{B}})$	Signature of **T** on $h_{\mathbf{B}}$ to prove its intervention

Table 6.2: The *exchange protocol* between **A** and **B**.

1. $\mathbf{A} \rightarrow \mathbf{B}$:	$\mathtt{C}$, $h_{\mathbf{A}}$
2. $\mathbf{B} \rightarrow \mathbf{A}$:	$h_{\mathbf{B}}$
3. $\mathbf{A} \rightarrow \mathbf{B}$:	$ACK_{\mathbf{A}}$

Table 6.3: The *cancel protocol* between **A** and **T**.

IF (*finished=true*)	1'. $\mathbf{A} \rightarrow \mathbf{T}$:	$\mathscr{H}(\mathtt{C})$, $h_{\mathbf{A}}, h_{\mathbf{AT}}$
	2'. **T**:	retrieves $h_{\mathbf{B}}$
	3'. $\mathbf{T} \rightarrow \mathbf{A}$:	$h_{\mathbf{B}}, h'_{\mathbf{B}}$
ELSE	2''. $\mathbf{T} \rightarrow \mathbf{A}$:	$\mathbb{S}_{\mathbf{T}}(\text{"}canceled\text{"}, h_{\mathbf{A}})$
	3''. **T**:	Stores *canceled=true*

had given the NR evidence to **B** ($ACK_{\mathbf{T}}$). Now **T** has to give the NR evidence to **A**, so **T** retrieves the previously stored NR evidence ($h_{\mathbf{B}}$), and sends it to **A**, together with an evidence to prove its intervention $\left(h'_{\mathbf{B}}\right)$. But if **B** had not previously contacted with **T**, **T** sends a message to **A** to cancel the transaction, and **T** stores this information (i.e. *canceled = true*) in order to satisfy future requests from **B**. Whatever was the case, now, the contract signing is again a fair situation.

If **B** claims that he has not received the third message, **B** may initiate the *finish*

Table 6.4: The *Finish protocol* between **B** and **T**.

	2'. $\mathbf{B} \rightarrow \mathbf{T}$:	$\mathscr{H}(\mathtt{C}), h_{\mathbf{A}}, h_{\mathbf{B}}, h_{\mathbf{BT}}$
IF (*canceled*=*true*)	3'. $\mathbf{T} \rightarrow \mathbf{B}$:	$\mathbb{S}_{\mathbf{T}}("canceled", h_{\mathbf{B}})$
ELSE	3". $\mathbf{T} \rightarrow \mathbf{B}$:	$ACK_{\mathbf{T}}$
	4". **T**:	Stores *finished*=*true* and $h_{\mathbf{B}}$

protocol (Table 6.4). During the finish protocol, **T** verifies the correctness of the information given by **B**. If the verification fails, **T** sends an error message to **B**. Otherwise, **T** proceeds in one of two following ways. If the variable *canceled* is *true* (i.e. **A** had previously contacted with the **T** (see the paragraph above)), **T** had given a message to **A** to cancel the transaction, and now it has to send a similar message to **B**. Otherwise, **T** sends the NR evidence $ACK_{\mathbf{T}}$ to **B**. In the last case, **T** also stores the NR evidence $h_{\mathbf{B}}$ and assigns the value *true* to the *finished* variable, in order to satisfy future requests from **A**. As *cancel protocol* does, whatever was the case, the contract signing becomes again in a fair exchange.

As a conclusion, the FPH contract signing scheme is claimed to be fair by their authors. Moreover, it meets the effectiveness and the non-repudiation properties and it does not make any timing assumptions (i.e. the protocol is asynchronous).

Informal Analysis of the Fairness and the Non-Repudiation Properties

After a protocol run is completed (with or without the participation of **T**), some disputes can arise between both signers. We can face with two possible types of disputes: repudiation of **A** (while **B** is claiming that the contract is *signed*) and repudiation of **B** (while **A** is claiming that the contract is *signed*).

An external arbiter or judge (not part of the protocol) has to evaluate the evidences held and brought by the parties to resolve these two types of disputes. As a result, the arbiter will determine who says the truth. The arbiter has to know who is the originator and who is the recipient. This information can be reviewed by the judge using $\mathtt{C}$ because it is defined as $\mathtt{C} = (\mathbf{A}, \mathbf{B}, \mathscr{M})$.

In case of repudiation of **A**, **B** is claiming that he received the signature on the contract $\mathtt{C}$ from **A**. He has to provide the following information to an arbiter: $\mathtt{C}$, $h_{\mathbf{A}}$ and $ACK_{\mathbf{A}}$ or $ACK_{\mathbf{T}}$. The arbiter checks whether $h_{\mathbf{A}}$ is the **A**'s signature on $\mathtt{C}$, and if it is true, the arbiter assumes that **A** had sent her signature to **B**. Then, the arbiter checks whether $ACK_{\mathbf{A}}$ is the **A**'s signature on $h_{\mathbf{B}}$, or it checks if $ACK_{\mathbf{T}}$ is **T**'s signature on $h_{\mathbf{B}}$. If this verification holds, the arbiter assumes that either **A** or **T** had sent an acknowledgement to **B**. Therefore, the arbiter will side with **B**. Otherwise, if one or both of the previous verifications fail, the arbiter rejects the demand of **B**. If the evidence held by **B** proves he is right, and **A** holds a message like $\mathbb{S}_{\mathbf{T}}(\mathscr{H}("canceled"), h_{\mathbf{A}})$, it means that either **T** or **A** had acted improperly.

In case of repudiation of **B**, **A** is claiming that **B** had signed the contract C. She has to provide the following information to an arbiter: C and $h_{\mathbf{B}}$. The arbiter checks if $h_{\mathbf{B}}$ is the **B**'s signature on C. If it is positive, the arbiter assumes that **B** had received both C and $h_{\mathbf{A}}$, thus he is committed to obtain the acknowledgement (either $ACK_{\mathbf{A}}$ or $ACK_{\mathbf{T}}$). If the previous verification fails, the arbiter rejects **A**'s demand. If the verification holds, the arbiter should interrogate **B**. If **B** contributes with a *cancel* message, it means that **B** contacted with **T**, and **T** observed that **A** had already executed the *cancel protocol*. For this reason, **T** sends the *cancel* message to **B**. Now it is proved that **A** has tried to cheat. Therefore, the arbiter rejects **A**'s demand, and the arbiter will side with **B**. If **B** cannot contribute with the *cancel* message, the arbiter will side with **A**.

As a conclusion, the protocol meets the non-repudiation requirement. Moreover, the protocol also fulfills the property of verifiability of the TTP [40]. This informal analysis does not cover all the possible situations derived from the execution of the contract signing scheme. It is here where a CPN analysis makes sense in order to formally verify the fairness of the FPH scheme. So in the next Section, we create a CPN model that mirrors the three protocols being part of the FPH scheme.

6.6.4 ECS1 Contract Signing Scheme due to Micali et al.

The ECS1 scheme due to Micali et al. (Table 6.5) and FPH scheme are similar, so we will use the same notation to describe them. Moreover, we use $\mathbb{E}_{\mathscr{P}}[X]$ to denote the encryption using the public key of $\mathscr{P}$ on the message X. So, **A** is committed to the contract ($\mathscr{M}$) as the initiator if **B** has both $\mathbb{S}_{\mathbf{A}}(\mathscr{M}, Z)$ and $\mathscr{R}$, where $Z = \mathbb{E}_{\mathbf{T}}[\mathbf{A}, \mathbf{B}, \mathscr{R}]$ and $\mathscr{R}$ is a random message. Then, **B** is committed to $\mathscr{M}$ as the recipient if **A** has both $\mathbb{S}_{\mathbf{B}}(\mathscr{M}, Z)$ and $\mathbb{S}_{\mathbf{B}}(Z)$.

Table 6.5: Micali's ECS1 scheme definition.

	1. **A** → **B**:	$\mathbb{S}_{\mathbf{A}}(\mathscr{M}, Z)$
	2. **B** → **A**:	$\mathbb{S}_{\mathbf{B}}(\mathscr{M}, Z), \mathbb{S}_{\mathbf{B}}(Z)$
IF (both signatures are valid)	3. **A** → **B**:	$\mathscr{R}$
IF (**B** receives valid $\mathscr{R}$ such that $Z = \mathbb{E}_{\mathbf{T}}[\mathbf{A}, \mathbf{B}, \mathscr{R}]$)		The exchange is completed
ELSE	1'. **B** → **T**:	$\mathbf{A}, \mathbf{B}, Z, \mathbb{S}_{\mathbf{B}}(\mathscr{M}, Z), \mathbb{S}_{\mathbf{B}}(Z)$
	2'. **T** → **A**:	$\mathbb{S}_{\mathbf{B}}(\mathscr{M}, Z), \mathbb{S}_{\mathbf{B}}(Z)$
	3'. **T** → **B**:	$\mathscr{R}$

6.7 Modeling the FPH Scheme with CPN

In this Section we explain how we have created the new model for the formal and automatic analysis of the FPH protocol, similar to that used by Sornkhom and

Permpoontanalarp [39] but adapted to the requirements of the current security analysis. Once the scheme is modeled, we can formally prove its behavior in case of malicious users. Our first goal is to prove the fairness of the FPH; first we will do that in case of malicious signers, and then we have modeled a malicious intruder.

In our model, we have defined the three parties involved in the scheme as described in §6.6: Alice (**A**), Bob (**B**) and the Trusted Third Party (**T**). In addition, we have modeled a new entity called the Intruder (**I**) who is a malicious party who can act acquiring different roles. So, **I** can be an observer (like a man in the middle) without doing anything than reading exchanged messages. **I** can also *drop*, *store*, *forward* or *modify* messages sent by any party involved in the exchange. While **T** is honest, both **A** and **B** could take the role of a malicious party. As a result, **A** and **B**, acquiring a malicious role ($\mathbf{A}_m$ and $\mathbf{B}_m$ respectively), can either stop the exchange or contact to **T** when they are not allowed to do that according to the scheme definition. This way, we can model a misbehaving party trying to cheat another party in order to gain advantage over the counterpart.

In order to model the *drop* and *stop* events made by misbehaving parties (i.e. $\mathbf{A}_m$, $\mathbf{B}_m$ or **I**), the model has a mechanism to inform about these events to the other affected parties. When an event happens, the model forces a notification to be sent by the party who either drops the message or stops the exchange and it will be received by the other involved parties. This assumption helps us to avoid the use of a timeout on each party, simplifying the management of the model. When an event message is received, the party could act either contacting **T** or stopping the exchange depending on the protocol step currently being performed and the behavior of the party.

Another important consideration is that messages between **T** and any other party of the model will always be delivered to the intended destination without any modification, even they passes through the modeled intruder.

With the previous considerations, we are able to build a scenario that can be used to model the protocol using different attack sessions, where each session can involve an initiator (i.e. either **A** or $\mathbf{A}_m$) and a receiver (i.e. either **B** or $\mathbf{B}_m$) depending of the role we want to apply to each signing party. Note that **I** and **T** are implicitly present in every session trace. So, this way we can deploy four protocol sessions: (**A**, **B**), ($\mathbf{A}_m$, **B**), (**A**, $\mathbf{B}_m$) and ($\mathbf{A}_m$, $\mathbf{B}_m$), where (a, b) denotes who is the originator party, a, and who acts as the recipient, b.

The architecture of the model can be divided into three blocks, using a top-bottom technique: top, entity and process levels. The model synchronizes these levels by using tokens with colors. Thus, all the modeled messages exchanged among parties are tuple composed by three elements: the source, the destination and the message itself as a payload. Figure 6.4 shows how it looks the formal definition of messages using the ML language. The first line defines the list of possible values that the *Id* colset can take (the name of the party). The second line defines a general protocol message as the union of all the defined messages in the model. Finally, the

NET colset is defined as the combination of both previous colsets, generating this way a message with the identities of the originator and the recipient, together with the message to be transferred. So, the model knows where to deliver each exchanged message and which is the data to be transferred.

```
colset Id = with A | B | T | I | Am | Bm | X | Y;
colset PM = union pp:PMp + p1:PM1 + p2:PM2 + ... + pe:PMe;
colset NET = product Id * Id * PM;
```

Figure 6.4: Definition of tokens with type and value (color).

The top level (Figure 6.5) shows the basic interaction among all the involved parties belonging to the scheme and the message flow among them. In the top level, the model exposes the contents of each party's database. These databases contain the protocol messages sent and received by the corresponding party along with the state of the protocol run (such as the next step to be performed). In addition, the top level model allows us to control the content of the session. This variable controls which role is assumed by each party during the current session (i.e. it sets whether the party behaves either honestly or maliciously). Moreover, Figure 6.5 reflects that messages always are intercepted by **I** in their transit among parties.

The entity level defines a more detailed model of the protocol and denotes all the steps that each party can execute. Figure 6.6 shows the entity level of **A** and her two roles: honest and malicious behaviors. In fact, honest transitions model protocols in the same way as they are defined by the scheme, while malicious transitions model actions out of the definition of the scheme, i.e. they model misbehaving actions such as stop the protocol where it is not allowed, contact **T** when it is not authorized, etc. Transitions TA_1 to TA_4 are the transitions corresponding to her honest role, while TAm_1 to TAm_4 are the transitions of her malicious role. The first transitions of **A**, i.e. TA_1 and TAm_1, generate the first *exchange protocol* message and send it to **B**. The transitions TA_2 and TAm_2 receive and verify the second *exchange protocol* message sent by **B** and they also send back to **B** the last *exchange protocol* message. TA_3 and TAm_3 have the responsibility to contact with **T** mirroring the behavior of the *cancel protocol*. The last transitions, TA_4 and TAm_4, receive the resolution from **T** to the *cancel protocol* request. Note that the selection of the transitions to be executed by the model is controlled by the session configuration which tells whether the party is either honest or malicious.

Regarding to the **B**'s entity level, as shown in Figure 6.7, it is build using the same concepts and notation as **A**'s entity level. Therefore, the honest role is implemented by transitions from TB_1 to TB_3, while the malicious role is controlled by transitions from TBm_1 to TBm_3. TB_1 and TBm_1 receive and verify the first message of the

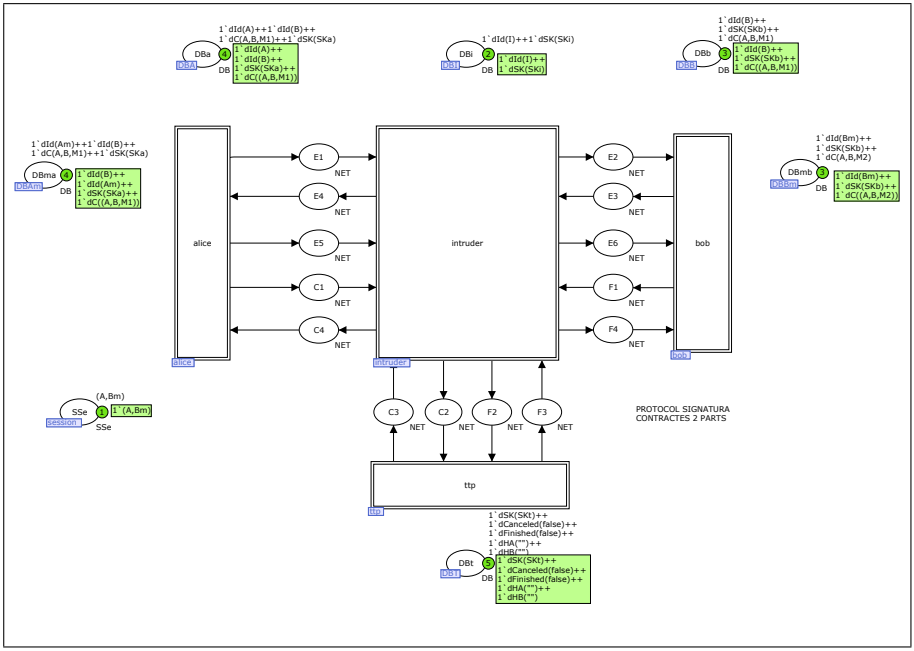


Figure 6.5: Top level scheme.

exchange protocol and they also send the corresponding second message. TB_2 and TBm_2 receive and verify the last *exchange protocol* message. If it is needed, both TB_2 and TBm_2 implements the call to the *finish protocol* to request **T** a resolution of the exchange. Lastly, TB_3 and TBm_3 receive the resolution from **T** to the *finish protocol* request.

The **T** entity level has two different processes, covering both the *cancel protocol* and the *finish protocol*. Figure 6.8 shows the model to represent the *cancel protocol*. The model receives the cancellation request from **A** and resolves the state of the protocol depending on the value of both *canceled* and *finished* internal variables. Finally **T** sends back to **A** the final resolution and also updates the value of his internal variables.

Thus, the process level implements all the actions that parties can run and it also specifies how the relations between the entities are. The actions performed by every process are atomic, i.e. only one process can be executed at the same time. It means that along the complete CPN model, only one transition can fire at once. It is so challenging because the model is very big and there are a lot of interactions among many transitions. This problem has been resolved by using an special place with a single token, which is shared between all the parties of the model. It is

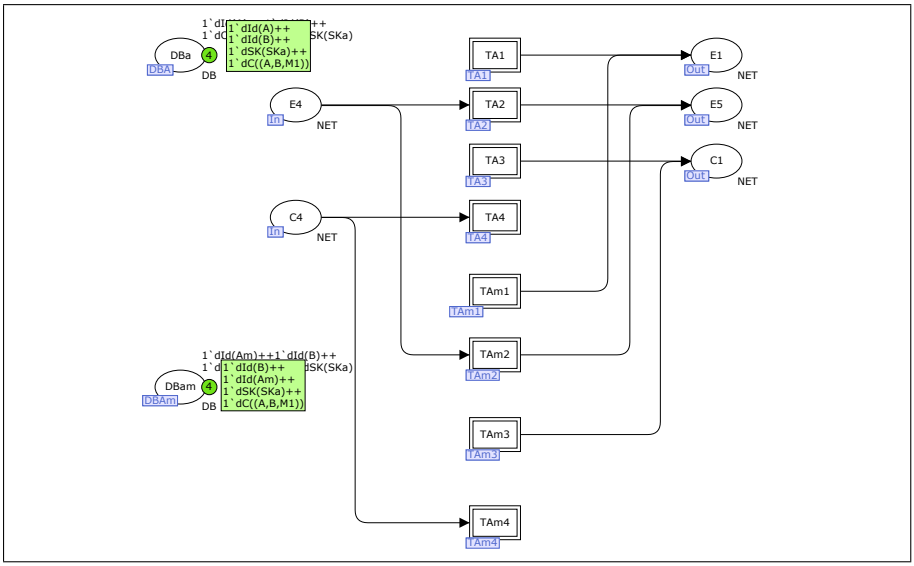


Figure 6.6: A's entity level.

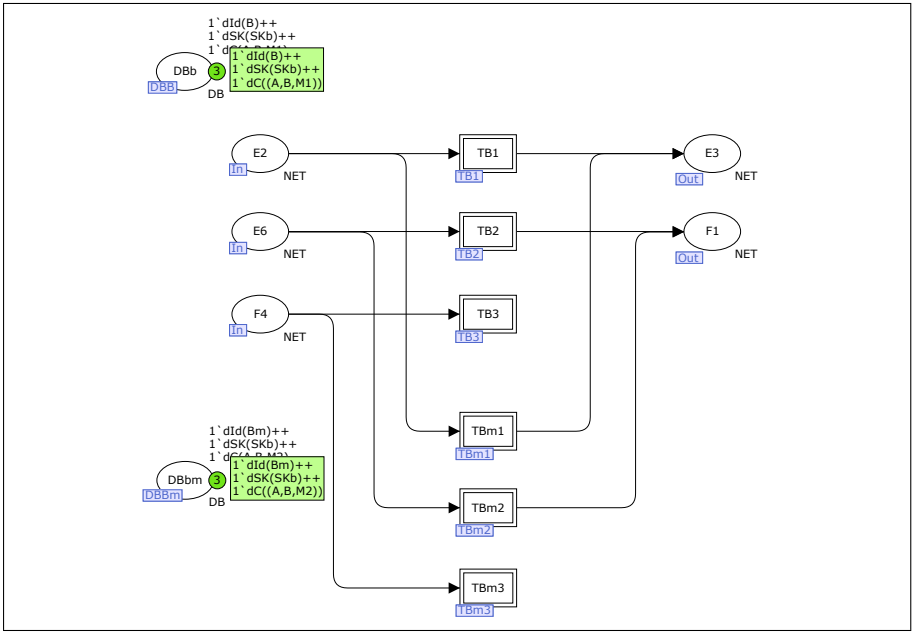


Figure 6.7: B's entity level.

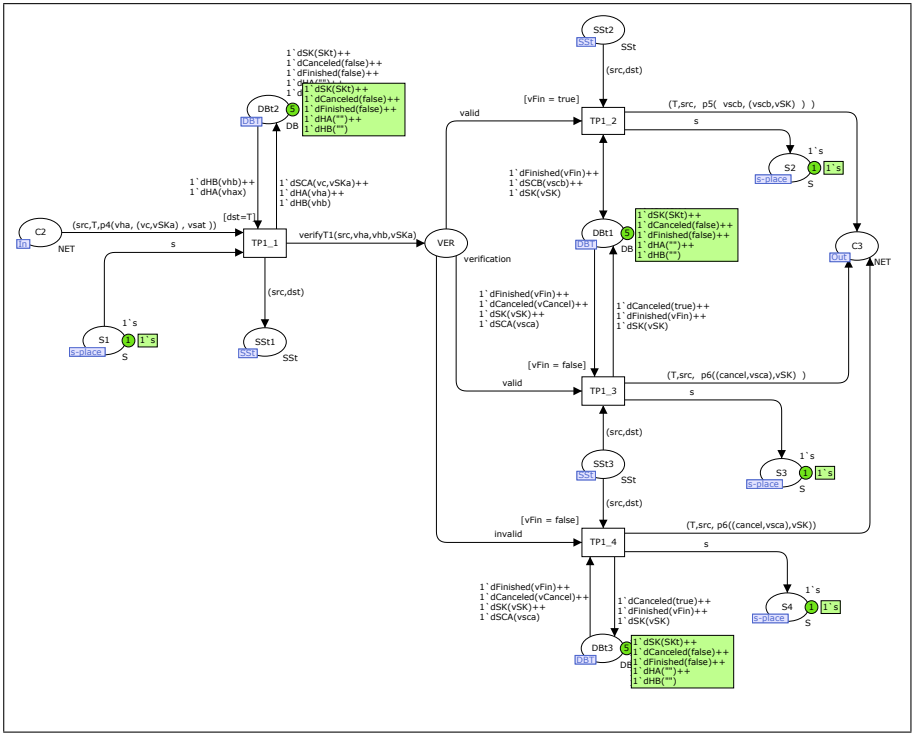


Figure 6.8: Process level of **T** party modeling the *cancel* protocol.

captured by each party when a process starts, and it will be released when the process finishes. This way, only a process can be fired at once. Moreover, each process level is controlled by another session flow control mechanism, internal to each process. This mechanism is defined as a token traveling through parties where at every step, they change its contents. This way, the token controls that all the actions runs in the right order within every process level. Therefore, it controls that only a single party can execute an action at the same time (i.e. to fire CPN transitions) and that action are performed either before or after than another action (e.g. the verification step being run after the corresponding receipt of the message).

6.7.1 Query Functions

In order to extract a list of attack scenarios from the state space, we have developed a set of query functions. Figure 6.9 shows one of these queries. It is developed to find special contents in each party database searching in the complete list of markings

```
fun SearchCommits( ack:DB, id:Id ) : Node list
  = PredAllNodes(
      fn n =>
      let
          val dba = Mark.Top'DBa 1 n
          val dbb = Mark.Top'DBb 1 n
          val dbi = Mark.Top'DBi 1 n
          val dbam = Mark.Top'DBma 1 n
          val dbbm = Mark.Top'DBmb 1 n
      in
          if (id=A) then
             cf(  ack ,  dba ) > 0
          else if (id=B) then
             cf( ack , dbb ) > 0
          else if (id=Am) then
             cf( ack, dbam ) > 0
          else if (id=Bm) then
             cf( ack, dbbm ) > 0
          else (* id=I *)
             cf( ack , dbi ) > 0
      end
)

fun SearchCommitsTerminalNodes( ack:DB, id:Id ) : Node list
   = PredNodes ( (SearchCommits(ack,id)) ,
                    fn n => (Terminal n) andalso (FullyProcessed n),
                    NoLimit)
```

Figure 6.9: Search query functions developed in order to search for commits into the party's databases.

contained in the state space generated after a complete and successful simulation. The main function is *SearchCommitsTerminalNodes(ack,id)*, where *ack* is the element or the commit we would like to search in the database of the party identified by *id* (being *id* one of the following values: **A**, $\mathbf{A}_m$, **B**, $\mathbf{B}_m$, **T** or **I**). The query returns a list of markings fulfilling some conditions. The function has been build with the help of the standard query function *PredNodes(p1,p2,p3)* from the ML language. The first parameter requested by the last function is another custom query called *SearchCommits(ack,id)*, where *ack* and *id* have the same meaning as in the previous query. It is capable to take up the contents of the desired database of party *id* telling if the *ack* element is either present in the database or not. The second parameter limits the search to find the leaf markings, i.e. those markings containing a full protocol run. Finally, the last parameter (*NoLimit*) tells the query it should check all the available markings and it should return all the results.

Thus, the *SearchCommitsTerminalNodes(ack,id)* query allows analyzing the compliance of the fairness property. In order to do this, the query is applied against the parties involved in the exchange (depending on the session configuration) to search for the desired commit, eventually returning a list of terminal markings meeting the proposed conditions. Thus, if we set the input variable *ack* with a commitment that

must not be present in the database of the party *id*, and the query function returns some markings, it means that the simulated contract signing run with the predefined session is not fair by some reason. Otherwise, we can assure that the exchange has been fair.

The analysis can be more complete doing a deep inspect on the contents of the markings returned by the query function. To do that, CPNTools provide a tool to analyze all the states and transitions among marking (see Figure 6.3 from §6.5). It is very interesting due to the fact that it means that we can examine all the protocol run from the first transition to the last one. Thus, it helps to clarify the reasons why the simulated exchange fails to be fair and thus to know which is the step that contributes to make the exchange unfair.

6.8 Formal Evaluation of FPH Using the CPN Methodology

Until today, several attacks to contract signing schemes have been described. Bao et al. [104] found three attacks to the ECS1 scheme (see §6.6.4). Later Sornkhom and Permpoontanalarp [39] found these vulnerabilities together with two new attacks to the same scheme applying a similar method with CPN. The consequence of these attacks is the loss of fairness of the contract signing scheme. For this reason, we have used the CPN model just described to evaluate whether the FPH scheme is resistant to these attacks. In order to do that, we list how it is performed each attack to the ECS1 scheme and then, we explain how to set the CPN model to prove whether FPH is resistant to these attacks. Moreover, after this analysis, we present a complete FPH formal fairness verification in order to obtain all the possible cases where the exchange could become unfair due to misbehaving signers.

6.8.1 Bao's First Attack

In the ECS1 scheme this attack (see in the Table 6.6) can be done if **A** sends a false Z in the first step of the protocol, where $Z = \mathbb{E}_{\mathbf{T}}[\mathbf{A}, \mathbf{B}, \mathscr{R}]$. In this case, **A** can always obtain the **B**'s commitment, but **B** will not have the **A**'s commitment. The attack is possible because **B** cannot verify the contents of the element Z received during the step 1 (see Table 6.5).

Table 6.6: Bao's first attack trace to the ECS1 scheme.

$\mathbf{A} \rightarrow \mathbf{B}$:	$\mathbb{S}_{\mathbf{A}}(\mathscr{M}, Z)$ where $Z = \mathbb{E}_{\mathbf{T}}[\mathbf{A}, \mathbf{B}, \mathscr{R}]$
$\mathbf{B} \rightarrow \mathbf{A}$:	$\mathbb{S}_{\mathbf{B}}(\mathscr{M}, Z)$, $\mathbb{S}_{\mathbf{B}}(Z)$
$\mathbf{A} \rightarrow \mathbf{B}$:	Nothing
$\mathbf{B} \rightarrow \mathbf{T}$:	$\mathbf{A}$, $\mathbf{B}$, Z, $\mathbb{S}_{\mathbf{B}}(\mathscr{M}, Z)$, $\mathbb{S}_{\mathbf{B}}(Z)$
$\mathbf{T} \rightarrow \mathbf{A}$:	Nothing
$\mathbf{T} \rightarrow \mathbf{B}$:	Nothing

In order to detect the attack in the model, we have generated a session with $\mathbf{A}_m$ (**A** acting maliciously) and **B**, as Figure 6.10 shows. In this attack, $\mathbf{A}_m$ builds a false contract $\mathscr{M}_2$ and she also sets a fake pair of originator and recipient (X, Y) (instead of the pre-agreed $\mathtt{C} = (\mathbf{A}, \mathbf{B}, \mathscr{M})$, she composes $\mathtt{C} = (X, Y, \mathscr{M}_2)$). The first query searches for the commitment $h_{\mathbf{A}}$ in the $\mathbf{A}_m$'s database. It returns four cases, corresponding to the markings 20, 21, 22 and 37. The second query searches the same element ($h_{\mathbf{A}}$) in the **B**'s database and as Figure 6.10 proves, **B** never receives this element. It is due to the fact that the verification stage fails and thus **B** never stores the received message. The last two queries search for the response of **T** into the $\mathbf{A}_m$'s database. Figure 6.10 shows that $\mathbf{A}_m$ only receives a *cancel* message (marking 37) while she never obtains the NR evidence from **T**.

Therefore, the FPH protocol is not vulnerable against the *first attack* of Bao et al. because **B** verifies the elements received during the step 1, so he no longer computes the message of the step 2. Thus, the *exchange protocol* finishes without success. If **A** tries to contact **T** to resolve the situation, **T** sends a cancellation proof and sets the *canceled* variable to *true*. **B** does not contact **T** because he does not have any valid element received from **A**.

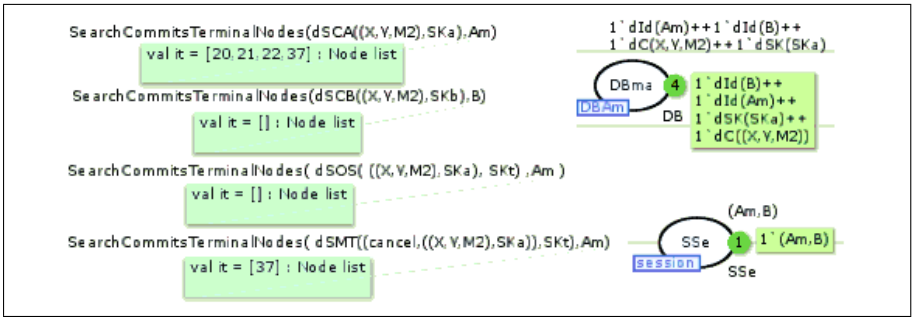


Figure 6.10: Bao's first attack query results, $\mathbf{A}_m$ database contents and session configuration.

6.8.2 Bao's Second Attack

In the ECS1 scheme, this attack (see Table 6.7) can be performed if a malicious **A** colludes with another party $\mathbf{A}'$ who acts as the originator of the exchange. $\mathbf{A}'$ sends a false Z element, where $Z = \mathbb{E}_{\mathbf{T}}\left[\mathbf{A}', \mathbf{B}, \mathscr{R}\right]$. In this case, the malicious **A** always obtains the **B**'s commitment on a contract between $\mathbf{A}'$ and **B**, but **B** does not get anything useful from the exchange. This attack is possible because **B** cannot verify the identity of the party **A** contained in the element received in the first step, so he cannot check that the real identity contained in Z is $\mathbf{A}'$ instead of **A**.

Table 6.7: Bao's second attack trace to the ECS1 scheme.

$\mathbf{A} \rightarrow \mathbf{B}$:	$\mathbb{S}_{\mathbf{A}}(\mathscr{M}, Z)$ where $Z = \mathbb{E}_{\mathbf{T}}\left[\mathbf{A}', \mathbf{B}, \mathscr{R}\right]$
$\mathbf{B} \rightarrow \mathbf{A}$:	$\mathbb{S}_{\mathbf{B}}(\mathscr{M}, Z)$, $\mathbb{S}_{\mathbf{B}}(Z)$
$\mathbf{A} \rightarrow \mathbf{B}$:	Nothing
$\mathbf{B} \rightarrow \mathbf{T}$:	$\mathbf{A}$, $\mathbf{B}$, Z, $\mathbb{S}_{\mathbf{B}}(\mathscr{M}, Z)$, $\mathbb{S}_{\mathbf{B}}(Z)$
$\mathbf{T} \rightarrow \mathbf{A}$:	Nothing
$\mathbf{T} \rightarrow \mathbf{B}$:	Nothing

The *Bao's second attack* can be simulated in the model using the same session configuration ($\mathbf{A}_m$,**B**) used to test the *Bao's first attack*, but using a different contract. In this case, we have built a false contract with a confabulated originator (X), the intended recipient (**B**) and the previously agreed contract ($\mathscr{M}$). So the contract looks as $\mathtt{C} = (X, \mathbf{B}, \mathscr{M})$ Figure 6.11 shows that the query results are the same as in the *first attack*, i.e. **B** never builds the second message of the *exchange protocol*. Then, if we search for the **T**'s resolution inside the $\mathbf{A}_m$'s database, we prove that $\mathbf{A}_m$ never obtains the NR evidence from **B**. Instead, she only could have a cancellation proof from **T** (marking 37).

So, the FPH protocol is not vulnerable against this attack due to the fact that **B** verifies the elements received in the step 1 of the *exchange protocol*. In case of the *second attack*, **B** does not send the message of step 2, in a similar way as in the *Bao's first attack*. Therefore, the *exchange protocol* is stopped, so **A** never obtains the corresponding **B**'s commitment. If **A** tries to conclude the exchange contacting **T**, she receives a cancellation proof. **B** does not contact with **T** because he does not want to finish the exchange as he knows that the element already sent in step 1 is false.

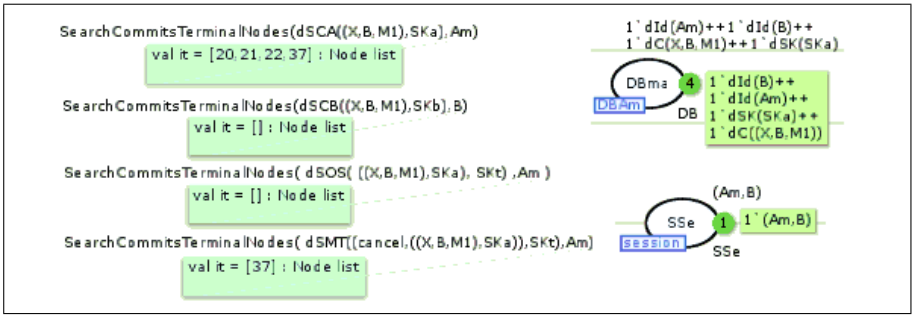


Figure 6.11: Bao's second attack query results, $\mathbf{A}_m$ database contents and session configuration.

6.8.3 Bao's Third Attack

In the ECS1 scheme, this attack (see Table 6.8) can be performed if **B**, after the receipt of a valid message in the step 1, contacts with **T** to start the *finish protocol* to resolve the exchange. During this request, **B** includes a fake contract. In this case, the malicious **B** always gets the **A**'s commitment on the original contract, but **A** obtains the **B**'s commitment on the fake contract (selected by the malicious **B**). This attack is possible because **A** cannot request the resolution of the exchange. Instead, she obtains from **T** the elements resulting of the resolution already started by **B**.

Table 6.8: Bao's third attack trace to the ECS1 scheme.

$\mathbf{A} \rightarrow \mathbf{B}$:	$\mathbb{S}_{\mathbf{A}}(\mathscr{M}, Z)$ where $Z = \mathbb{E}_{\mathbf{T}}\left[\mathbf{A}', \mathbf{B}, \mathscr{R}\right]$
$\mathbf{B} \rightarrow \mathbf{T}$:	Z, $\mathbb{S}_{\mathbf{B}}\left(\mathscr{M}', Z\right)$, $\mathbb{S}_{\mathbf{B}}(Z)$ for a false contract $\mathscr{M}'$
$\mathbf{T} \rightarrow \mathbf{A}$:	$\mathbb{S}_{\mathbf{B}}\left(\mathscr{M}', Z\right)$, $\mathbb{S}_{\mathbf{B}}(Z)$
$\mathbf{T} \rightarrow \mathbf{B}$:	$\mathscr{R}$

The *Bao's third attack* can be verified with the model using a session configuration where **A** is the honest originator and $\mathbf{B}_m$ is the malicious recipient $(\mathbf{A}, \mathbf{B}_m)$. $\mathbf{B}_m$ builds a contract containing a false plain text but using the real originator and recipient. So the contract is defined as $(\mathtt{C} = (\mathbf{A}, \mathbf{B}, \mathscr{M}_2))$. As Figure 6.12 exposes, when $\mathbf{B}_m$ receives the first message, he changes its contents by setting a contract with a different text $(\mathscr{M}_2)$. Then, we have searched for whether a false $h'_{\mathbf{B}}$ sent by $\mathbf{B}_m$ is inside the **A**'s database and, effectively, it is in the marking 63. Although **A** stores

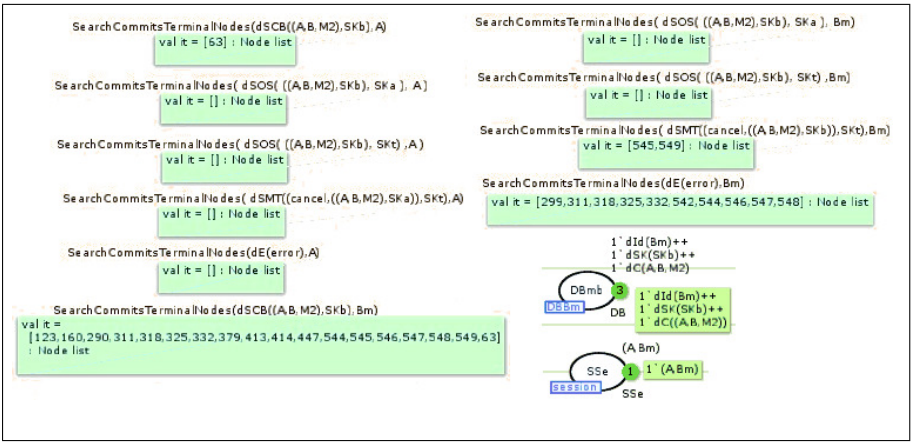


Figure 6.12: Bao's third attack query results, $\mathbf{B}_m$ database contents and session configuration.

the message, she have checked that the message is wrong so she does not generate the last message of the *exchange protocol*. Then, **A** can contact **T**, but she would ask for the original and true contract using the *cancel protocol*. Thus, **T** sends her a cancellation proof. Finally, we can search for **T**'s responses in $\mathbf{B}_m$'s database returned by the execution of the *finish protocol*, finding that he never obtains the alternative proof. Moreover, he can only obtain the cancellation proof and an error message because **T**'s verification fails.

The FPH scheme, rather than the ECS1 signing protocol, when **A** receives a false $h'_{\mathbf{B}} = \mathbb{S}_{\mathbf{B}}\left(\mathbf{A}, \mathbf{B}, \mathscr{M}'\right)$ during the step 2, she detects the attack, stops the exchange and contacts **T**. If **B** contacts **T** in the first place and the request contains a false $h_{\mathbf{B}}$, **T** has been able to detect that $h_{\mathbf{A}}$ and $h_{\mathbf{B}}$ are not related to the same contract. Then, when **A** starts a resolution request using the *cancel protocol*, **T** sends her a cancellation proof, so the contract has not been signed. Otherwise, if **A** contacts **T** first, she obtains a cancellation proof. So, the FPH scheme is not vulnerable to the proposed attack.

6.8.4 Sornkhom's First Attack

The *Sornkhom's first attack* was described first in [39]. It is possible because the ECS1 scheme has an incomplete definition of the **B**'s commitment $(\mathbb{S}_{\mathbf{B}}(\mathscr{M}, Z), \mathbb{S}_{\mathbf{B}}(Z))$ since this evidence is not linked to the identity of the initiator. So, anybody who has it can in fact claim to be the originator of the contract committed by **B**.

The *Sornkhom's first attack* can be verified in the model using a session between two honest parties: **A** and **B**. Figure 6.13 shows that the databases of both **A** and **B** contain the previously committed contract. In this case, we would search states where the intruder **I** eavesdrops messages. So, the first query finds a single state in the **B**'s database where **I** have changed the originator of the contract (i.e. **I** instead of **A**). Finally, by using the third query it is proved how **B** never builds his commitment ($h_{\mathbf{B}}$) on the wrong contract with **I** as the originator because he can verify the received NR evidence and realize that it had been manipulated.

As opposed to the ECS1 scheme, the FPH protocol has linked the commitment of **B** to the contract. The evidence is the commitment $h_{\mathbf{B}} = \mathbb{S}_{\mathbf{B}}(\mathtt{C})$, however, holding this evidence is not enough for anyone to prove that **B** has committed himself to the contract `C`. It is due to the fact that FPH specifies that `C` must contain the contract to be signed ($\mathscr{M}$) together with the identity of who is the originator (**A**) and who is the recipient (**B**). Thus, the FPH protocol is resistant to this attack.

6.8.5 Sornkhom's Second Attack

In the *Sornkhom's second attack* (see trace in Table 6.9) described in [39], a malicious **A** can obtain the **B**'s commitment on a contract between **B** as an originator and any colluding party $\mathbf{A}_r$ as a recipient. But **B** will not get anything. So, this attack

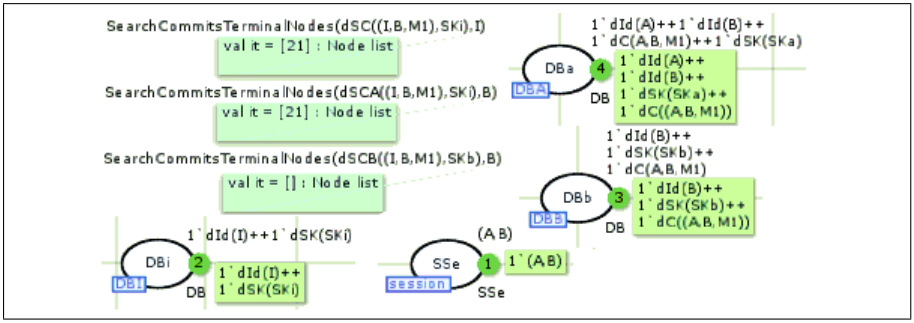


Figure 6.13: Sornkhom's first attack query results, parties' database contents and session configuration.

consists on swapping the initiator and the recipient roles as to get the commitment of **B** on a contract where he is in fact the originator instead of the recipient. In order to perform the attack, **A** involves **B** in the protocol so as to exchange the commitments on a contract. But **A** builds a fake element Z with the identity of **B** as the initiator and a colluding party $\mathbf{A}_r$ as the recipient ($Z = \mathbb{E}_{\mathbf{T}}[\mathbf{B}, \mathbf{A}_r, \mathscr{R}]$). Finally **A** gives $\mathbb{S}_{\mathbf{B}}(\mathscr{M}, Z)$ and $\mathscr{R}$ to $\mathbf{A}_r$. **T** cannot send anything to neither **A** nor to **B** because the item Z does not fulfill the protocol specifications. Now, $\mathbf{A}_r$ can successfully claim to have the commitment of **B** on the contract being **B** the originator of the exchange and **B** does not have any useful evidence.

Table 6.9: Sornkhom's second attack trace to the ECS1 scheme.

$\mathbf{A} \rightarrow \mathbf{B}$:	$\mathbb{S}_{\mathbf{A}}(\mathscr{M}, Z)$ where $Z = \mathbb{E}_{\mathbf{T}}[\mathbf{B}, \mathbf{A}_r, \mathscr{R}]$
$\mathbf{B} \rightarrow \mathbf{A}$:	$\mathbb{S}_{\mathbf{B}}(\mathscr{M}, Z)$, $\mathbb{S}_{\mathbf{B}}(Z)$
$\mathbf{A} \rightarrow \mathbf{B}$:	Nothing
$\mathbf{B} \rightarrow \mathbf{T}$:	$\mathbf{A}$, $\mathbf{B}$, Z, $\mathbb{S}_{\mathbf{B}}(\mathscr{M}, Z)$, $\mathbb{S}_{\mathbf{B}}(Z)$
$\mathbf{T} \rightarrow \mathbf{A}$:	Nothing
$\mathbf{T} \rightarrow \mathbf{B}$:	Nothing

The verification of this attack using the CPN model (see Figure 6.14) considers a session composed by $\mathbf{A}_m$ and **B**. In this case, $\mathbf{A}_m$ changes the contents of the contract, swapping the party's roles but using the right previously agreed contract ($\mathscr{M}$). So this time the contract is composed by $\mathtt{C} = (\mathbf{B}, \mathbf{A}, \mathscr{M})$ The way to apply the query functions to search for suspicious markings is the same as in the Bao's *first* and *second attacks*. The first query searches for the first element $h_{\mathbf{A}}$ in the $\mathbf{A}_m$'s database. The second query searches for the second message allocated in the **B**'s database, without any result, so we prove that **B** does not build it. Finally, the third and fourth queries try to search for responses provided by **T** in the $\mathbf{A}_m$'s database proving that she only obtains a cancellation proof (marking 37).

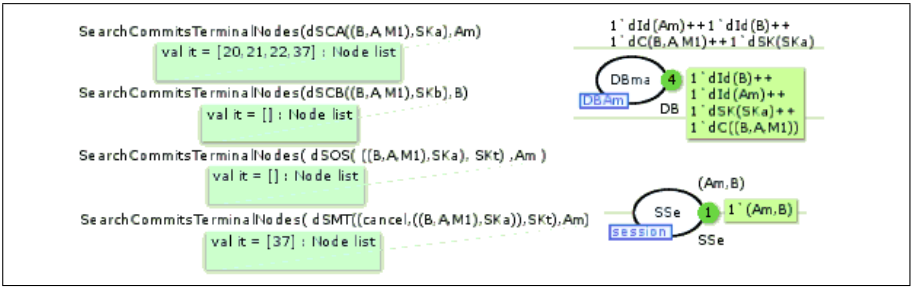


Figure 6.14: Sornkhom's second attack query results, $\mathbf{A}_m$'s database contents and session configuration.

In fact, in the FPH scheme, as explained before, **B** verifies the commitment received during the step 1 of the *exchange protocol*. Thus, if **A** has made improper changes in the message, **B** detects it as soon as he verifies the received NR evidence. Therefore, he decides to stop the *exchange protocol*. So, the model proves that the attack has not been successful.

6.8.6 Full Formal Analysis of the FPH Scheme with CPN

In [40], authors described a conflicting situation where **A** could obtain the NR evidence from **B** ($h_{\mathbf{B}}$) and a *cancel* message from **T**, while **B** only obtains the NR evidence from **A** ($h_{\mathbf{A}}, ACK_{\mathbf{A}}$). **A** can do it, for instance, invoking the *cancel protocol* after the end of a successful execution of the *exchange protocol*. Therefore, **A** has two types of evidences, so she can affirm that the contract is either *signed* or *canceled*, depending on her usefulness. Instead, **B** only possesses the NR evidence proving that the contract is signed. As stated in [40], it means that the evidences generated by this protocol are not transferable, so an arbiter must contact both signers to solve a dispute, to know the final state of the exchange and to guarantee the non-repudiation property. When the evidences are not transferable it is called that the scheme has *weak fairness* [73].

Our first objective with the CPN model is to discover this conflicting situation. Surprisingly, we find out more situations in which signers have contradictory evidences. So, in this Section we complete the formal fairness analysis of FPH and we describe these conflicting situations together with other situations in which signers have more than a single NR evidence even if they are not contradictory. Finally, we give a guideline to the arbiter which solves all kinds of conflicting situations derived from the execution of the contract signing.

Regarding to the CPN model, we have configured it to give us all the possible behaviors of both signers. Therefore, the model has been configured to use a ses-

sion with a malicious originator and a malicious recipient ($\mathbf{A}_m$,$\mathbf{B}_m$). Using the already known query functions, Figure 6.15 shows we have searched into each party's database the desired commits. In this case, we have searched the second and the third messages of the *exchange protocol* together with all the responses provided by **T**.

Studying the list of markings obtained from each query, we build the Table 6.10 with the conflicting cases. In each case, we denote the state of the contract signing execution from the point of view of each involved party, either as *signed* (S) or *canceled* (C). Moreover, we also denote whether $\mathbf{A}_m$ or $\mathbf{B}_m$ have contacted **T**, either maliciously (M) or honestly (H). As described in the Table 6.10, using the model we have located three cases in which both $\mathbf{A}_m$ and $\mathbf{B}_m$ have contradictory evidences even we have detected four possible scenarios, because *case 3* appears twice. The following list explains in detail every situation and how parties could obtain more than a single NR evidence.

- Case 1. It happens on marking 488, when $\mathbf{A}_m$ obtains the NR evidence from $\mathbf{B}_m$ (so **A** has the evidence that the contract is *signed*), but she contacts **T** in order to cancel the exchange. This is a malicious behavior, because $\mathbf{A}_m$ should not contact **T** to cancel an exchange that is already finished. **T** resolves the situation and sends the *cancel* message to $\mathbf{A}_m$. As a result, $\mathbf{A}_m$ could affirm

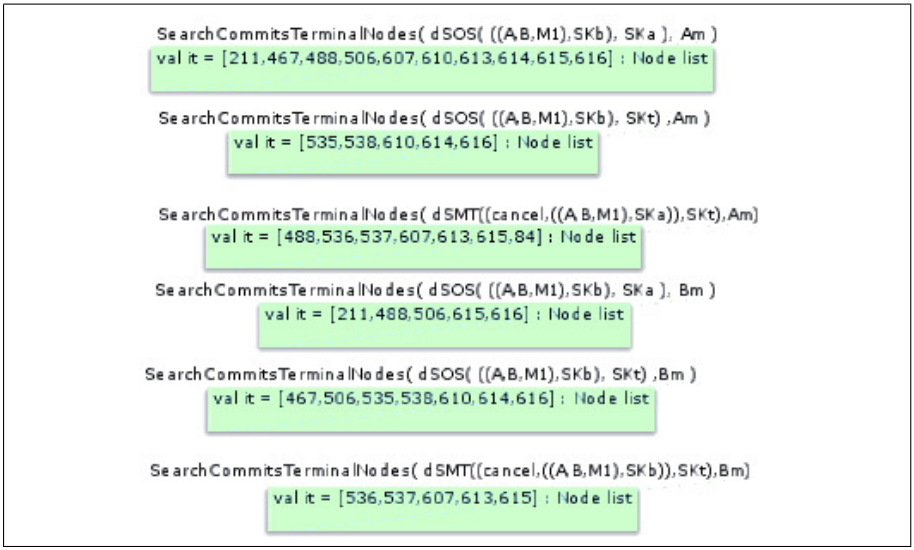


Figure 6.15: Query functions results over the model with ($\mathbf{A}_m$,$\mathbf{B}_m$) session.

Table 6.10: List of the markings corresponding to the three cases with contradictory evidences (where two markings correspond to the same case).

Case	Marking	$\mathbf{A}_m$ has NR	$\mathbf{B}_m$ has NR	$\mathbf{A}_m$ contacts **T**	$\mathbf{B}_m$ contacts **T**
1	488	S & C	S	Yes (M)	No
2	615	S & C	S & C	Yes (M)	Yes (M)
3	607	S & C	C	Yes (M)	Yes (H)
3	613	S & C	C	Yes (M)	Yes (H)

that the contract is either *signed* or *canceled*. $\mathbf{B}_m$ receives the NR from $\mathbf{A}_m$ and he does not need to contact **T**, so he only has the NR evidence that the contract is *signed*.

- Case 2. In this situation, corresponding to the marking 615, the *exchange protocol* finishes successfully for each party. However, $\mathbf{A}_m$ contacts **T** after transferring the NR evidence to $\mathbf{B}_m$, in order to obtain a *cancel* message. Once $\mathbf{B}_m$ receives the NR evidence from $\mathbf{A}_m$, he also contacts **T** and he obtains a *cancel* message. So, both $\mathbf{A}_m$ and $\mathbf{B}_m$ have a malicious behavior because they contact **T** when they should not. As a result, it seems that both signers can state that the contract is either *signed canceled* depending on her usefulness.

- Case 3. The situation is detected twice on the model. In both scenarios, $\mathbf{A}_m$ obtains the NR evidence from $\mathbf{B}_m$ but $\mathbf{B}_m$ never receives the third message of the *exchange protocol*. On one hand, during the first scenario (marking 607), $\mathbf{A}_m$ decides to maliciously stop the exchange and she does not send the third message to $\mathbf{B}_m$. In the other hand, the second scenario (marking 613) is the result of a *drop* event on the third message by **I**. In each scenario, $\mathbf{A}_m$ executes maliciously the *cancel protocol* before **B** calls to the *finish protocol*. Therefore, she receives a *cancel* evidence from **T**. From the point of view of $\mathbf{B}_m$, both scenarios are the same. He asks **T** in order to resolve the situation, obtaining a *cancel* proof in both situations (because **A** has contacted **T** before **B**). As a result, **A** has the NR evidence received from **B** stating that the contract is *signed* and the cancellation proof generated by **T**. Note that if **B** had contacted with **T** first, it had sent him the NR evidence proving the contract was *signed*. Thus, the malicious **A** has changed the final resolution issued by **T** due to her misbehavior.

In order to solve these conflicting situations an arbiter must always contact both parties. We have established the right resolution that the arbiter has to choose to

solve these situations with contradictory evidences as follows:

- Case 1: **A** can state that the contract is either *signed* or *canceled*, but **B** has the NR evidence proving that the contract is *signed*. If **A** tries to use the *cancel* proof, the arbiter will know she is in fact a cheating party, so the arbiter must side with **B**.
- Case 2: If **B** shows the NR evidence proving that the contract is *signed*, the arbiter must state that the contract is in fact *signed*. Instead, if **B** shows the *cancellation* proof, the arbiter must state that the contract is *canceled*. This way, due to the fact that **A** is always the first cheating party, if the arbiter sides always with **B**, the protocol will discourage **A** to act fraudulently.
- Case 3: Once again, **A** has acted fraudulently, and if the arbiter sides with **B**, he must state that the contract is in fact *canceled*.

Thus, we have detected the previously defined conflicting situation and we have also discovered two additional cases thanks to the use of the CPN model. All the cases are due to the fraudulent behavior of **A**. To solve these situations, an arbiter must contact both parties and in case of conflict, the arbiter must always side with **B**. This way, the protocol becomes fair and in addition, the fraudulent behavior of signers is discouraged. So, the solution is easy to apply and, as a result, the contract signing scheme has been improved without any modification to the core of the scheme.

Table 6.11: Scenarios without contradictory evidences.

Marking	$\mathbf{A}_m$ has NR	$\mathbf{B}_m$ has NR	$\mathbf{A}_m$ contacts **T**	$\mathbf{B}_m$ contacts **T**
84	C	Nothing	Yes (H)	No
211	S	S	No	No
467	S	S	No	Yes
506	S	S & S (by **T**)	No	Yes
535	S (by **T**)	S (by **T**)	Yes (H)	Yes (H)
536	C	C	Yes (H)	Yes (H)
537	C	C	Yes (H)	Yes (H)
538	S (by **T**)	S (by **T**)	Yes (H)	Yes (H)
610	S & S (by **T**)	S (by **T**)	Yes (M)	Yes (H)
614	S & S (by **T**)	S (by **T**)	Yes (M)	Yes (H)
616	S & S (by **T**)	S & S (by **T**)	Yes (M)	Yes (M)

Besides to the above conflicting cases, there are other situations detected by the model in which parties could have repeated proofs, even if they are not contradictory evidences. It is due to the fact that parties may contact **T** when the protocol has

been successfully finished. Table 6.11 displays the markings without contradictory evidences but with multiple repeated NR evidences. The notation to describe every case is the same as the used above in the Table 6.10.

The most interesting cases displayed in the Table 6.11 are both markings 84 and 616. In the marking 84, $\mathbf{B}_m$ has nothing from $\mathbf{A}_m$ because an intruder **I** has executed a *drop* event on the first message of the *exchange protocol*. $\mathbf{B}_m$ cannot execute the *finish protocol* because he does not have any valid element from $\mathbf{A}_m$. $\mathbf{A}_m$ resolves the contract executing the *cancel protocol*, obtaining a *cancel* message from **T**. The marking 616 happens when both $\mathbf{A}_m$ and $\mathbf{B}_m$ act maliciously contacting **T** after the *exchange protocol* has successfully finished. It is similar to the case 615 (see Table 6.10), but in this particular case, $\mathbf{B}_m$ contacts in the first place **T**, obtaining the corresponding NR evidence. Then, if $\mathbf{A}_m$ tries to cancel, **T** sends her a NR evidence that states the contract has been already *signed*.

As a conclusion, the CPN methodology has been proved to be very useful to formally verify the fairness and the non-repudiation properties of the FPH scheme. Moreover, the model returns a list of situations where parties have more than a single NR evidence, even among them there are only three conflictive situations where signers could have contradictory NR evidences that can be used by them as to gain advantage. Armed with this previously undetected cases, we have enhanced the FPH contract signing scheme in a sense that an arbiter can resolve these situations to recover the fairness of the protocol lost by the misbehavior of some malicious parties. Therefore, the presented CPN method has been proved useful to produce formal analysis of properties such as the fairness and the non-repudiation of other schemes, at least similar ones. However, the successful application of the method must be studied case by case, as it will depend on the complexity of the scheme or protocol being analyzed.

6.9 Case Example II: Multi-Party Contract Signing Scheme

After analyzing the two-party version of the FPH scheme, we have decided to try to verify the fairness of the multi-party version of the same protocol [40]. Summarizing the multi-party FPH scheme, as in the two-party version, there are a set of signers who exchange NR evidences directly without the intervention of **T** during the *exchange protocol*. Every party wants to be bound to the contract if all parties are bound at the end of the exchange, i.e. nobody wants a partially signed contract. Signers build a signing ring where each of them have a predefined order which is specified by the contract to be signed, as well as in the two-party version. Then, each signer i receives NR evidences from the prior signer in the ring (i.e. the $i-1$ signer), processes this information and sends the corresponding NR evidence to the subsequent signer in the ring (i.e. the $i+1$ signer). At the end of the *exchange pro-*

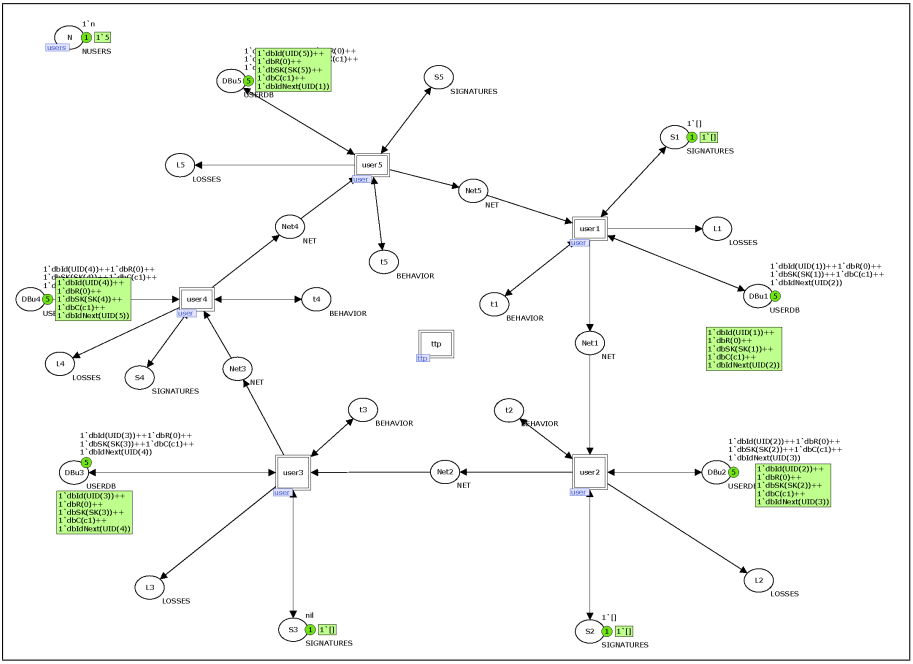


Figure 6.16: Top level CPN modeling the multi-party FPH scheme.

tocol, if every party behaves honestly, the contract will be signed by all of them. The protocol needs up to $3n-1$ rounds and exchanges 2^n NR evidences, where n is the number of signers. In case signing parties cannot get the expected NR evidences from the other parties, **T** is invoked by starting either the *cancel* or the *finish protocol*.

Figure 6.16 shows the top level CPN model of the multi-party FPH scheme, where five signers exchange NR evidences making up a signing ring. Every signer can be a malicious party, interrupting the *exchange protocol* and requesting the contract resolution to **T** (requesting either the cancellation or the finalization of the contract signing). Figure 6.17 models the entity level of **T**. It was difficult to model the behavior of **T**, because of the great number of verifications and state variables that enable **T** to issue the right response about the state of the contract signing (whether the contract is *signed* or *canceled*). This entity level covers up to five process levels modeling all the possible resolutions defined by the multi-party FPH scheme.

After the CPN model has been completed, the simulation and state space generation can be started in order to obtain all the available cases through the use of the corresponding tool from CPNTools. Unfortunately, taking into account only an exam-

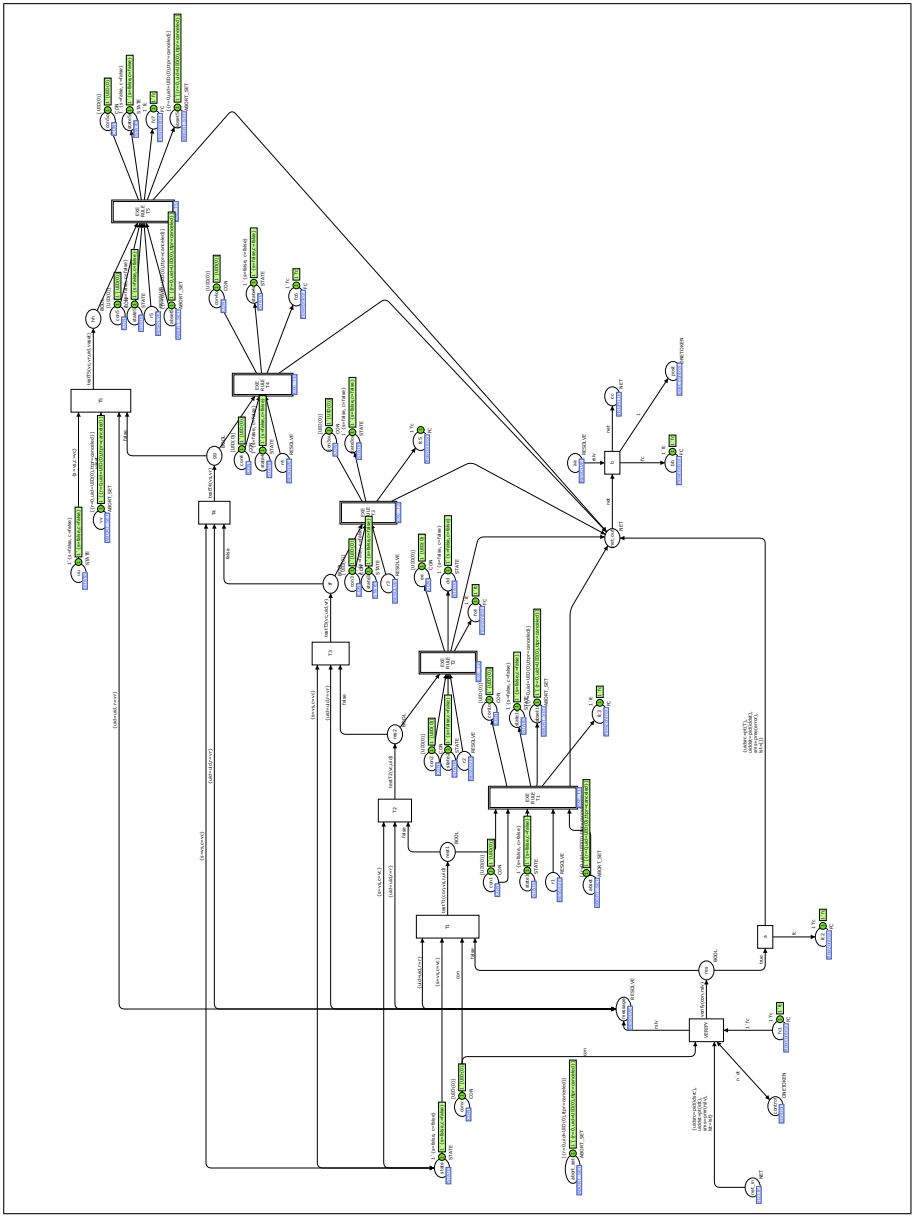


Figure 6.17: Entity level of **T** modeling resolution in the multi-party FPH scheme.

ple with five signers with malicious behavior, a problem called *state-explosion* has appeared making the verification unfeasible. The *state-explosion* problem emerges when the number of generated markings becomes impractical to manage by the CPNTools application and it stops to work. This trouble can be reduced if we consider less misbehaving signers instead to consider that all of them are malicious. However, it is not a good solution due to the fact that we are not be able to cover all the possible states, so the method becomes useless.

The other main problem with the multi-party version is the fact that every time we change the number of signers, the model has to be changed and the variables has to be updated. Recall that the number of rounds to execute and the number of NR evidences to be exchanged and carried in a single message depends on the number of signers, as stated in the scheme description. Unfortunately, the ML language does not have means to implement this behavior, so we are forced to update the model every time we want to verify the scheme with a different number of signers, making this process tedious and not useful at all.

6.10 Conclusions

In this Chapter we have studied different ways to automatize the formal analysis of secure protocols. There are different languages and frameworks to model and verify business and distributed applications. These frameworks have different features and they should be used depending on the objectives we want to reach with the analysis. However, we claim that the CPN language, derived from the original Petri Net modeling language, is the best option to automatically verify security properties of secure protocols and schemes, as stated by previous works [38, 39].

We have presented a methodology to model secure protocols (although it can be applied to other types of protocols without it being secure protocols) and to formally analyze the accomplishment of some security properties. In order to test and prove the applicability of the CPN method, we have selected a contract signing scheme, the FPH scheme due to Ferrer et al. [40] to test whether it is resistant to some attacks previously discovered in [39] on a similar contract signing scheme, the ECS1 scheme due to Micali et al. [103]. This way, we have familiarized with CPN and at the same time, we have conducted a formal analysis of a secure contract signing scheme. Surprisingly, the developed CPN model has been proven so useful as it has been able to discover new situations in which the scheme could present unfair states due to misbehaving signers. These situations have not been found by FPH's authors and they are not yet discovered. So, using the model, we have been able to formally verify the fairness and non-repudiation requirements of the FPH contract signing scheme and in addition, we have improved the original contract signing scheme to protect it from more unfair situations.

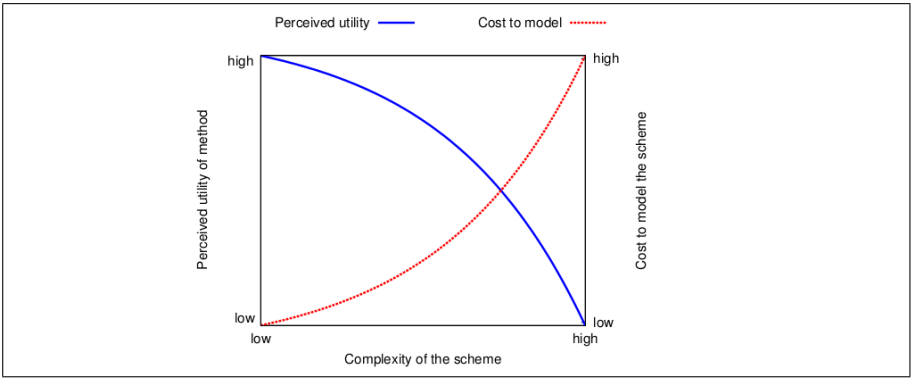


Figure 6.18: The perceived utility and the cost to model a secure scheme depending on its underlying complexity.

Although CPN has been proved its benefits to be applied in the field of automatic verification of security properties, not always it provides tangible advantages, as stated in §6.9. In the exposed case, we have tried to model and automatically analyze the multi-party version of the FPH scheme. However, due to some limitations of the CPN language, such as the difficulty to model the number of rounds depending on the number of signers, the method is not as useful than it was for the two-party version. In fact, the described CPN method will be useful depending on the complexity of the scheme to analyze. Figure 6.18 shows in a graphic way that the perceived utility of the method is inversely proportional to the cost to model the scheme. Therefore, before to proceed with the development of the CPN model, we advise that it is better to conduct a deep inspection of the scheme or the protocol to be modeled in order to be sure whether the method can be applied without major problems. So, it is necessary to conduct an analysis of the scheme's complexity before to model it due to the fact that it can be perceived as a trade off between the cost to model and the final utility of the model.

CHAPTER 7

SECURITY ANALYSIS OF PROPOSED PRIVACY-PROTECTING SCHEMES

In this Chapter we are going to analyze the security of the proposed privacy-protecting schemes presented in Part II in order to validate that they accomplish their respective security requirements, as stated by their respective security models. In Section 7.1 we provide the formal security analysis of the *μEasyPay* micropayment scheme for low-value purchases presented in Chapter 3. Moreover, we present a formal and automatic analysis of the fairness property by using the CPN method described in Chapter 6. In Section 7.2 we describe the formal security analysis of the $\mathscr{MC}-2\mathscr{D}$ multicoupon scheme with multi-merchant scenario described in Chapter 4. Finally, a formal security analysis of the AFC proposal presented in Chapter 5 is conducted in Section 7.3.

7.1 Formal and Automatic Security Analysis of the Micropayment Scheme

Along this Section we provide a formal analysis of our micropayment scheme described in §3.4 to analyze the achievement of the required security properties as stated in the security model described in §3.2.3.

7.1.1 Unforgeability

Theorem 7.1.1 *Taking into account the underlying RSA factorization problem from the partially blind signature and the one-wayness and collision resistance of a cryptographic hash function, the micropayment scheme explained in §3.4 is unforgeable w.r.t. the Definition 3.2.4.*

Proof. As stated by the unforgeability *Definition 3.2.4*, there are some forge instance that should be verified in order to prove that the micropayment scheme meets the unforgeability requirement. Summarizing, to forge the micropayment scheme, an adversary has either to forge the partially blind signature or to find a collision in the cryptographic hash function.

Concerning the unforgeability game 0, as described in the *Definition 3.2.4*, several forge scenarios could meet the condition that $\mathcal{A}_0$ tries to spend a microcoupon not really belonging $\mathbb{C}$ ($\mathtt{mc} \notin \mathbb{C}$). The first considered case may happen when $\mathcal{A}_0$ tries to forge the micropayment spending a microcoupon $\left(\mathtt{mc} = \left[\omega_{(i+1)\mathscr{C}}, \omega_{i\mathscr{C}}\right]\right)$ beyond the limits set by $\mathbb{C}$. It means that $\mathcal{A}_0$ uses a microcoupon whose index is either $i > 2N$ or $i \leq 0$. The coin identifier $(\omega_{0\mathscr{C}}, \omega_{0\mathscr{M}})$ is bound to the common information $\left(\Gamma = \left(2N, v, \tau_{exp}, \Delta\tau_{op}, \Delta\tau_r\right)\right)$ during the `Withdrawal` protocol by using the partially blind signature scheme. The common information contains the number of microcoupons held by $\mathbb{C}$ (i.e. $2N$ hash identifiers, thus N microcoupons $\mathtt{mc}$). So, either the merchant or the bank can verify whether the received index and the corresponding microcoupon are within the limits of $\mathbb{C}$. It is clear that if the index is $i > 2N$ or $i \leq 0$, both merchant and bank does not accept the microcoupon. Related to the former case, $\mathcal{A}_0$ could try to use a microcoupon not belonging $\mathbb{C}$ but providing an index within the limits specified by the common information ($i \leq 2N$ or $i > 0$). In that case, applying the hash chain properties, the verifier can easily determine that the provided index does not belong the provided microcoupon. As before, the verifier does not accept the microcoupon because it does not belong $\mathbb{C}$ ($\mathtt{mc} \notin \mathbb{C}$).

Another forge scenario may take place if $\mathcal{A}_0$ has found and used a microcoupon really not belonging $\mathbb{C}$ while the verifier does not detect it. It would mean that $\mathcal{A}_0$ has actually found a collision in the hash function. However, this case has negligible probability to happen as we use a secure cryptographic one-way hash function with resistance to collisions (in $\mathtt{K}$).

The third case consists on an adversary $\mathcal{A}_0$ trying to modify some of the elements described by the common information. Recall that the common information field contains the number of microcoupons, their value and a list of time marks. To modify any content within the common information, $\mathcal{A}_0$ must modify $\mathbb{C}$, in particular the partially blind signature ($\mathbb{C}_{\mathscr{PBS}}$). However, $\mathbb{C}_{\mathscr{PBS}}$ cannot be modified without either merchant of bank detecting it due to the fact that the partially blind signature is unforgeable (in $\mathtt{K}$) (refer to §3.3 from [55]).

The last unforgeability case is defined as the self-withdrawn $\mathbb{C}$. Since the auto-generated $\mathbb{C}$ has not been created by a recognized bank (i.e., $\mathbb{C}$ has not been signed by a private key whose owner should be a recognized bank and certified by a trusted CA), a verifier does not accept $\mathbb{C}$ as he only accepts genuine coins correctly issued from a recognized bank.

Therefore, w.r.t. the game 0, the micropayment scheme is unforgeable under the collision resistance of the cryptographic one-way hash function and the unforgeability of the partially blind signature scheme. Besides, an adversary cannot create self-withdrawn coins as they will not be accepted by either merchant or bank since they have been withdrawn by an unrecognized bank.

Regarding to the unforgeability game 1, $\mathscr{A}_1$ may try to deposit more coupons that the number he had received from honest customers. However, $\mathscr{A}_1$ cannot do it as it would mean that $\mathscr{A}_1$ has successfully forged either the partially blind signature or a microcoupon and we have proved above that it is unfeasible for an adversary with non-negligible probability (in $\mathtt{K}$).

In a formal way and following the *Definition 3.2.4* together with the above proofs, if an adversary $\mathscr{A}$ ($\mathscr{A}_0, \mathscr{A}_1$) can forge a coin $\mathbb{C}$, it means that the adversary $\mathscr{A}$ has success in one of the forge instances: forge the partially blind signature or find a collision of a cryptographic one-way hash function. However, both instances have negligible probability to happen (in $\mathtt{K}$). On one hand, $\mathscr{A}$ has to factorize the modulus used by the partially blind signature scheme. As long as the partially blind signature scheme is based on the discrete logarithm problem, it is assumed that it is unfeasible for $\mathscr{A}$ to find a solution to this problem (find the modulus factorization) with non-negligible probability (in $\mathtt{K}$). On the other hand, $\mathscr{A}$ cannot find with non-negligible probability (in $\mathtt{K}$) a collision, since it is not feasible under the fact that we have used a secure one-way collision resistant hash function. As a result, neither $\mathscr{A}$ ($\mathscr{A}_0, \mathscr{A}_1$) can win any forge game with non-negligible probability (in $\mathtt{K}$).

7.1.2 Anonymity of Customers

Lemma 7.1.2 *The micropayment scheme described in §3.4 protects the anonymity of customers w.r.t. the Definition 3.2.5 because the coin $\mathbb{C}$ does not contain any data about the customer's identity and besides, during a call to the* `Spend` *protocol, no information is exchanged that could be used to infer the customer's identity.*

Proof. During the `Withdrawal` protocol, $\mathscr{C}$ has to authenticate herself in order to be allowed by $\mathscr{B}$ to withdraw money and issue the coin $\mathbb{C}$. Indeed, in the first step of the `Withdrawal` protocol, $\mathscr{C}$ sends $\mathscr{B}$ the common information signed ($\mathbb{S}_{\mathscr{C}}(\Gamma, \alpha)$) along with her digital certificate from a trusted CA ($Cert_{\mathscr{C}}$). This way, $\mathscr{B}$ authenticated $\mathscr{C}$ while she also states that she wants to withdraw a coin with a value equivalent to $v_{\mathbb{C}} = (\Gamma.N \cdot \Gamma.v)$. Despite the fact the `Withdrawal` protocol is not an

anonymous process by construction, the final result (i.e. the coin $\mathbb{C}$) is anonymous. The withdrawn $\mathbb{C}$ conveys two types of data: the chained list of microcoupons ($\mathbb{C}_\omega$) and the partially blind signature ($\mathbb{C}_{\mathscr{P}\mathscr{B}\mathscr{S}}$). Neither field have any information about the identity of $\mathscr{C}$ even if during the `Withdrawal` protocol $\mathscr{C}$ provides $\mathscr{B}$ with her real identity to obtain the $\mathbb{C}_{\mathscr{P}\mathscr{B}\mathscr{S}}$ element. Therefore, during the process of `Withdrawal`, $\mathscr{B}$ issues coins without any customer identification.

Concerning the anonymity game 0 from the *Definition 3.2.5*, an adversary $\mathscr{A}$ acting as a malicious $\mathscr{M}$, cannot successfully identify the customer who runs the `Spend` protocol, examining data in $\mathcal{V}_{\mathscr{A}}^{\texttt{Spend}}$. $\mathscr{C}$ sends a single or a set of microcoupons to $\mathscr{A}$ during the exchange of the `Spend` protocol. As stated before, no information about her identity is carried by microcoupons since they are only hash identifiers derived from a random seed element, applying the hash chain properties. Besides, $\mathbb{C}_{\mathscr{P}\mathscr{B}\mathscr{S}}$ neither contains information about the identity of $\mathscr{C}$.

As a result, $\mathscr{A}$ cannot identify $\mathscr{C}$ while she is spending microcoupons by using the `Spend` protocol.

7.1.3 Unlinkability

Theorem 7.1.3 *Under the unlinkability of the partially blind signature scheme, the micropayment scheme described in §3.4 is 2N-unlinkable w.r.t. the Definition 3.2.6.*

Proof. Microcoupons from different coins (i.e. $\omega_{i\mathscr{C}} \in \mathbb{C}_\omega$ and $\omega'_{i\mathscr{C}} \in \mathbb{C}'_\omega$, $\forall i$) cannot be linked. It is due to the fact that chain identifiers from two different coins are generated iteratively from a unrelated and random seed identifier ($\omega_{2N\mathscr{C}}$). Indeed, by hash chain properties, it can be proved that from two different and random seed identifiers, the process outputs two different and unrelated coin identifiers $\left(\omega_{0\mathscr{C}} \neq \omega'_{0\mathscr{C}}\right)$. So, it means that even two microcoupons from the same coin can be linked by hash chain properties (i.e. $\omega_{i\mathscr{C}}, \omega_{(i+j)\mathscr{C}} \in \mathbb{C}$ can be linked to the same $\mathbb{C}$ applying $\mathscr{H}^{i+j-1}\left(\omega_{(i+j)\mathscr{C}}\right) = \omega_{i\mathscr{C}}$), two microcoupons from different coins cannot be linked.

In case of unlinkability game 0, if the adversary $\mathscr{A}_0$ can be able to link two `Spend` ($\mathscr{A}_0$ has two views $\mathcal{V}_{\mathscr{A}_0}^{\texttt{Spend}_0}$ and $\mathcal{V}_{\mathscr{A}_0}^{\texttt{Spend}_1}$) executions to the same customer, it means that $\mathscr{A}_0$ has broken the unlinkability of the underlying partially blind signature. However, it only happens with non-negligible probability thanks to the use of a secure partially blind signature (refer to §3.3 from [55] to show the proof).

In case of unlinkability game 1, we should prove that an adversary $\mathscr{A}_1$ cannot link $\mathbb{C}$ (in particular $\mathbb{C}_{\mathscr{P}\mathscr{B}\mathscr{S}}$) to the `Withdrawal` protocol in which $\mathbb{C}$ had been withdrawn. As defined by the game 1, $\mathscr{A}_1$ has obtained a view $\mathcal{V}_{\mathscr{B}}^{\texttt{Withdrawal}}$ from an execution of a `Withdrawal` protocol by a corrupted $\mathscr{B}$. This view contains the messages exchanged between $\mathscr{C}$ and $\mathscr{B}$ when they had been engaged in a previous `Withdrawal` protocol, so it also contains the digital certificate of $\mathscr{C}$. However,

due to the way $\mathscr{C}$ completes locally the generation of $\mathbb{C}_{\mathscr{PBS}}$ (remember that $\mathscr{C}$ unblinds the last message received by $\mathscr{B}$ during the `Withdrawal` protocol, obtaining actually the final $\mathbb{C}_{\mathscr{PBS}}$), $\mathscr{B}$ neither knows the final coin identifier $(\omega_{0\mathscr{C}}, \omega_{0\mathscr{M}})$. So, the latter element is not present in the provided view. Consider that $\mathscr{A}_1$ executes a `Spend` protocol with a customer that he does not know certainly whether she is the same customer as who had executed the `Withdrawal` protocol. Then, if $\mathscr{A}_1$ can link $\mathbb{C}_{\mathscr{PBS}}$ with the corresponding `Withdrawal` run in which it had been withdrawn, $\mathscr{A}_1$ has success in breaking the unlinkability of the underlying partially blind signature scheme. However, it only happens with non-negligible probability (in `K`) under the assumption that the partially blind signature is unlinkable, similarly as in the game 0 (refer also to §3.3 from [55]).

As a result, no p.p.t. adversary $\mathscr{A}$ $(\mathscr{A}_0, \mathscr{A}_1)$ can win any unlinkability game with non-negligible advantage over random guess (in `K`). Thus, the micropayment scheme is $2N$-unlinkable.

7.1.4 Untraceability

Theorem 7.1.4 *Under both the anonymity of customers and the unlinkability proofs, the micropayment scheme presented in §3.4 meets the untraceability property as defined by the Definition 3.2.7.*

Proof. Considering the proofs of both Lemma 7.1.2 and the Theorem 7.1.3, a p.p.t. adversary $\mathscr{A}_0$ acting as a malicious merchant (even colluding with a corrupted $\mathscr{B}$) can neither discover the identity of the customer who runs the `Spend` protocol nor can link two `Spend` transactions made by using different coins ($\mathbb{C}_1$ and $\mathbb{C}_2$) with non-negligible advantage over random guess (in `K`). Moreover, although using microcoupons from the same coin $\mathbb{C}_1$, $\mathscr{A}_0$ cannot know the real identity of $\mathscr{C}$ even microcoupon can be linked easily by hash chain properties.

It is also worthy to remark that although $\mathscr{C}$ is merchant-specific, $\mathscr{B}$ does not know the merchant a customer wishes to use the coin with. So, if the bank issues n coins for a particular customer, $\mathscr{B}$ cannot assure whether it has issued n coins for n different merchants or for the same merchant.

As a consequence, we can remark the following:

***Remark* 7.1.1** *Customers are protected from profiling made by malicious parties, in particular malicious merchants.*

7.1.5 Microcoupon Reuse Avoidance

Theorem 7.1.5 *Under the unforgeability of the partially blind signature scheme (thus, the unforgeability of the micropayment scheme itself) and the one-wayness and*

collision resistance of the cryptographic hash function, the micropayment scheme described in §3.4 detects and avoids the microcoupon reuse w.r.t. the Definition 3.2.8.

Proof. Although coins are self-contained, in a sense that any party can verify $\mathbb{C}$ because it carries all the required information for its verification, both $\mathscr{M}$ and $\mathscr{B}$ have to maintain their own database to store partially and still valid coins together with the microcoupons already used belonging that coins. So, every row of that databases encloses a tuple $(\mathbb{C}, \omega_{i\mathscr{C}}, i)$ with the aim to control the last received microcoupon $(\omega_{i\mathscr{C}})$ along with its position within the microcoupon chain (i) and the corresponding coin $(\mathbb{C})$. With checking this tuple of elements, both $\mathscr{B}$ and $\mathscr{M}$ are able to determine whether a single microcoupon had already been used before.

As exposed in §3.2.8, three types of microcoupon reuse instances can be defined based on who tries to cheat. First, during the microcoupon reuse game 0, an adversary $\mathscr{A}_0$ can try to execute the `Spend` protocol using an already used microcoupon with the honest merchant. Secondly, during the microcoupon reuse game 1, an adversary $\mathscr{A}_1$ can try to run the `Deposit` protocol with $\mathscr{B}$ sending an already deposited microcoupon. Finally, and similarly to the above case, an adversary $\mathscr{A}_2$ can try to perform the `Refund` protocol trying to cheat $\mathscr{B}$ using a microupon already used, deposited or refunded before. In every case, the verifier can check whether the coupon has already spent, deposited or refunded earlier checking whether the provided microcoupon is either in $\mathscr{M}$'s database or in $\mathscr{B}$'s database. Based on the proof of the unforgeability Theorem 7.1.1, no adversary $\mathscr{A}$ $(\mathscr{A}_0, \mathscr{A}_1, \mathscr{A}_2)$ has success on playing any microcoupon reuse game without it being detected and avoided by the corresponding verifier (for instance, either $\mathscr{B}$ or $\mathscr{M}$) with non-negligible probability (in $\mathtt{K}$). Note that, when $\mathscr{C}$ tries to cheat $\mathscr{M}$, he cannot know her identity so he only is able to deny the current `Spend` transaction. Instead, as both $\mathscr{C}$ and $\mathscr{M}$ has to be authenticated as to access their bank accounts, $\mathscr{B}$ can know immediately who are the cheaters trying to reuse a microcoupon already deposited or refunded, so $\mathscr{B}$ can undertake measures against them. Finally, as deposit and refund periods are totally disjoint it is not possible that $\mathscr{C}$ asks for a refund of a set of already used microcoupons but not yet deposited by $\mathscr{M}$.

7.1.6 Overspending Protection

Lemma 7.1.6 *The debit-based micropayment scheme described in §3.4 avoids customers overspend w.r.t. the Definition 3.2.9, because there is no customer able to spend more money than the funds she currently holds in her bank account at the moment of withdrawn.*

Proof. $\mathscr{B}$ is in charge of storing bank accounts of customers $(\mathtt{acc}_{\mathscr{C}})$. So, $\mathscr{B}$ always knows the remaining funds at every customer account $\left(\mathtt{v}_{\mathscr{C}}^{\mathtt{acc}}\right)$. When a

customer executes the `Withdrawal` protocol with $\mathscr{B}$, she has to provide the pre-agreed common information that contains the value and the number of microcoupons she want to withdrawn. It is trivial to prove that whether the value she is willing to withdrawn is larger than the amount of money she currently holds in her bank account $\left(\mathrm{v}_{\mathscr{C}}^{\texttt{Withdrawal}} > \mathrm{v}_{\mathscr{C}}^{\mathrm{acc}}\right)$, $\mathscr{B}$ denies the withdrawn and returns an error describing the situation with an anotation reporting the remaining funds.

7.1.7 Fairness

Lemma 7.1.7 *The micropayment scheme presented in §3.4 meets the controlled fairness requirement w.r.t. the Definition 3.2.13, because the scheme can either assure a the fairness with a resolution procedure implemented in the* `Deposit` *and* `Refund` *protocols or it can provide an improved efficiency without resolution procedures but providing a limited and controlled risk of loss.*

Proof. The `Spend` protocol has been designed in such a way that, once it finishes, both $\mathscr{C}$ and $\mathscr{M}$ will have the desired items from the other party, if they honestly follow each protocol step. Indeed, if involved parties in the `Spend` protocol behave correctly, then the exchange becomes fair. So, $\mathscr{M}$ gets the required microcoupon from $\mathscr{C}$ while $\mathscr{C}$ obtains the item she has willing to purchase from $\mathscr{M}$.

However, the exchange could became unfair whether $\mathscr{C}$ or $\mathscr{M}$ decides to not follow the `Spend` protocol honestly.

The first case happens whether it is $\mathscr{C}$ who does not act in accordance with the `Spend` protocol. Certainly, if $\mathscr{C}$ does not send the third message of the `Spend` protocol with the proof microcoupon $\left(\omega_{(i+1)\mathscr{C}}\right)$, $\mathscr{C}$ has the desired item while $\mathscr{M}$ does not have the corresponding proof microcoupon. However, in order to repair the fairness of the purchase transaction, $\mathscr{M}$ can run the `Deposit` protocol within the available period (see Figure 3.2 from §3.4.1) defined by the corresponding time mark $\left(\tau_d = \tau_{exp} + \Gamma.\triangle\tau_{op}\right)$ allocated inside the common information field and partially blind signed by $\mathscr{B}$. During the `Deposit` protocol, $\mathscr{M}$ sends the last received payment microcoupon $(\omega_{i\mathscr{C}})$ which index position within the microcoupon chain has odd value i, together with the corresponding product previously sent by $\mathscr{M}$ to $\mathscr{C}$. $\mathscr{B}$ executes the *Case1* from the `Deposit` protocol resolution. Therefore, $\mathscr{B}$ stores the product for the exception case explained in the next claim and $\mathscr{B}$ deposits the correct amount into the merchant's bank account. Thus, $\mathscr{M}$ is paid by the product sold and as a result, the fairness is restored.

The second situation is due to the malicious behavior of $\mathscr{M}$ during the `Spend` protocol. In fact, if $\mathscr{M}$ does not send the second message of the `Spend` protocol with the item requested by $\mathscr{C}$, she will be actually involved in an unfair situation if $\mathscr{M}$ executes the `Deposit` protocol. As stated in the previous case, $\mathscr{M}$ can get the full value of a microcoupon using this protocol even he has not received the proof

information. In this case, $\mathscr{B}$ executes the *Case1* of the `Refund` protocol. As a result, the last index stored by $\mathscr{B}$ in its database is odd and $\mathscr{B}$ has the item sent by $\mathscr{M}$, as said before. So, $\mathscr{B}$ can provide the item to $\mathscr{C}$ and it can refund the rest of unused microcoupons from $\mathbb{C}$ to $\mathscr{C}$. Therefore, $\mathscr{C}$ has received the item and whether she has had unused microcoupons, she has received the corresponding refund. So, the fairness has been restored again.

We have analyzed the fairness taking into account a procedure split between `Deposit` and `Refund` protocols to recover the fairness from an unfair situation due to a malicious behavior. However, this resolution implies the participation of $\mathscr{B}$ storing a provided item by $\mathscr{M}$ during the `Deposit` protocol to sent it afterward to the harmed $\mathscr{C}$ during the `Refund` protocol. As stated in §3.2.10 and §3.4.1, there are some services in which a *controlled fairness* should be assumed either to improve the efficiency or simply because the resolution does not have sense for the concrete service provisioning (e.g. a streaming service). For these cases, our micropayment scheme allows relaxing the initial fairness avoiding the previous resolutions. Therefore, under the *controlled fairness* property (see *Definition 3.2.13* from §3.2.10), the scheme remains somehow fair but with a controlled and limited risk of loss. The risk the involved parties have to assume is the loss of a single microcoupon per coin $\mathbb{C}$. Since we are dealing on a micropayment for low-value items, in which a single microcoupon represents a small amount of money (up to few cents), the risk is completely reasonable. Besides, if a single party presents a repeated malicious behavior causing that in most of the transactions it tries to cheat the other party, the malicious one has the risk to suffer a partial loss of his reputation. We refer the reader to the formal and automatic analysis of the fairness by the use of the CPN methodology in §7.1.8 to be convinced that harmed party in the last supposition only may loss a single microcoupon. Moreover, we complete the analysis of the first micropayment solution with resolutions with the same formal method.

***Remark* 7.1.2** *The micropayment scheme allows tweaking its efficiency in exchange of a controlled and limited risk of loss.*

7.1.8 Automatic Formal Fairness Verification with Colored Petri Net

In spite fairness has been discussed and proved during the above security discussion, we want to provide further proofs to demonstrate how the micropayment scheme from §3.4 meets the *controlled fairness* property as described by the *Definition 3.2.13*. To do it, we present an automatic and formal analysis of this property using the knowledge acquired in Chapter 6. So, first we describe the CPN model that allows us to perform the automatic analysis by using CPNTools. Therefore, we have applied the CPN method to our micropayment proposal taking into account both available versions: either considering the resolutions with the support of $\mathscr{B}$ or

without them. Moreover, we give some insights about how the CPN method allows proving also the correctness of the scheme detecting a little flaw that was corrected in the final version depicted in §3.4.

The Considered CPN Model

Figure 7.1 contains the representation of the top-level CPN net that models the micropayment scheme. In this figure, we have modeled the three involved entities. Note that the `Withdrawal` protocol has not been modeled due to the fact that we want to prove both the correctness and the fairness of spending transactions. Because of these, we have implemented the `Spend`, `Deposit` and `Refund` protocols.

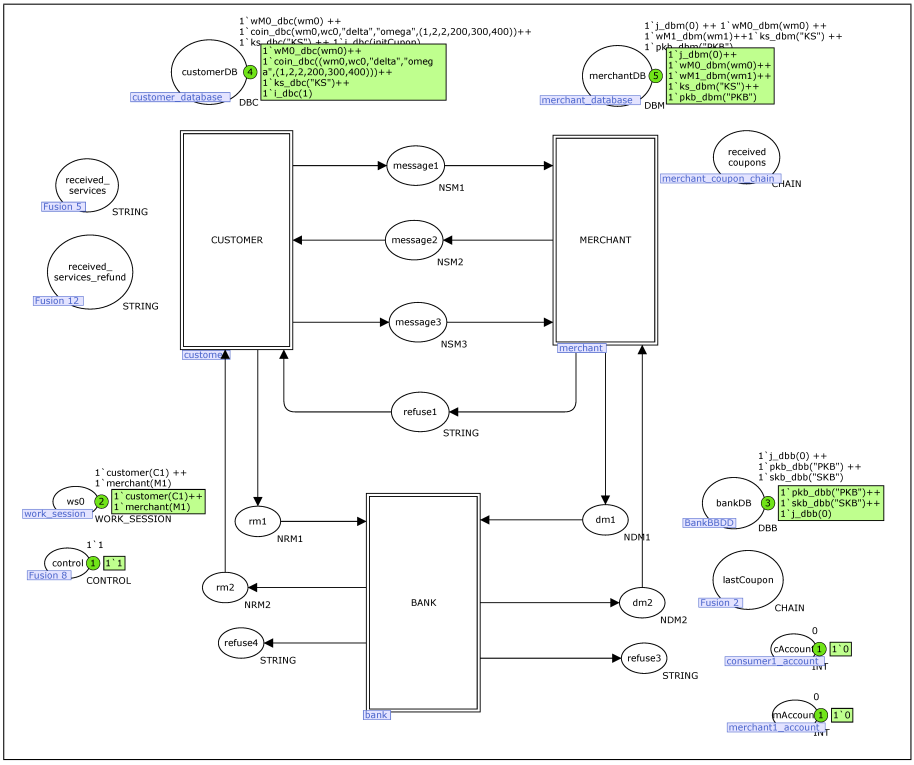


Figure 7.1: Top-level net modeling the micropayment scheme.

Many colors and places have been defined in order to model all types of messages transferred during the protocols and also to store tokens of different colors (such as microcoupons, purchased items, etc.) in the databases of each entity. As already

explained in Chapter 6, every rectangle in the top-level net (Figure 7.1) represents a substitution transition, i.e. a big transition that covers the concrete behavior of the corresponding party. Figure 7.2 shows part of the CPN model of the customer entity.

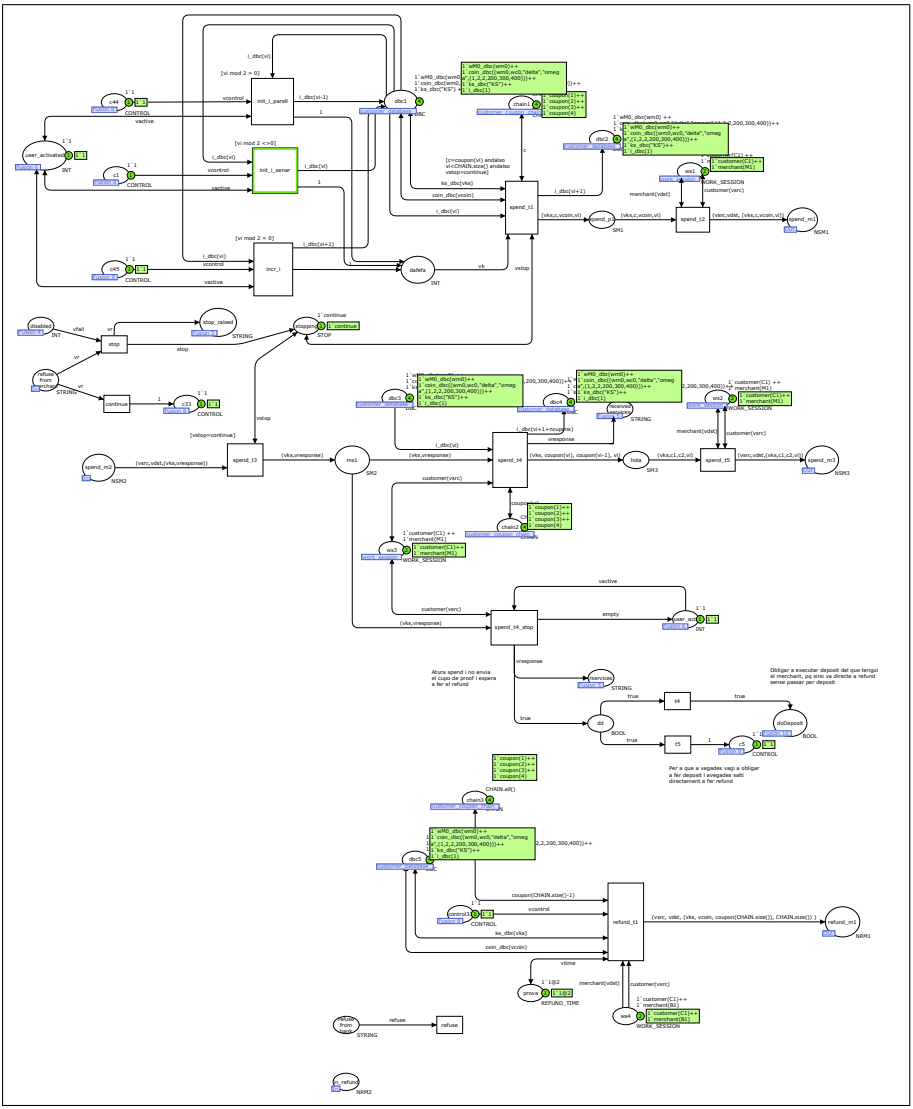


Figure 7.2: CPN modeling part of the customer entity.

Once the model has been modeled, we have to add all the initial markings to the model, such as the key pairs and the initialization of the coin with the number of microcoupons belonging it. We have decided to evaluate the model taking into account two microcoupons, i.e. the chain has four hash identifiers ($\omega_{4_{\mathscr{C}}}$ ... $\omega_{1_{\mathscr{C}}}$) together with the chain identifier ($\omega_{0_{\mathscr{C}}}$). So the automatic model will execute the `Spend` protocol at most twice. However, the considered number of microcoupons can be increased but the number of states will become impractical to manage.

Step-by-step Model Checking: Discovering and Amending a Flaw

The next stage to perform is checking the model step-by-step verifying whether it works as expected. It is during that stage in which we have detected a flaw in the way $\mathscr{B}$ updates the indexs of microcoupons received from both merchants (during the `Deposit` protocol) and customers (during the `Refund` protocol). Figure 7.3 compares the corrected `Deposit` protocol with the older one. It shows how indexes were updated improperly during the `Deposit` protocol, causing an inconsistency in the database of $\mathscr{B}$ and how the corrected protocol resolves it. Figure 7.4 presents the corrected `Refund` protocol with lots of changes compared with the older version. So, before to perform the automatic simulation to obtain the complete state space of the system, we have corrected an undetected problem in the scheme. Thus, the CPN methodology has been proved very useful to check also the correctness of the scheme even without using the automatic simulation tool.

Case1. i odd:	**Case1.** i odd:
1. Verifies whether PRD is provided, otherwise stops	1. Verifies whether PRD is provided, otherwise stops
2. Deposits value between chain identifiers $\left[w_{(i+1)_{\mathscr{C}}}, w_{j_{\mathscr{C}}}\right]: \lfloor \frac{i-j+1}{2} \rfloor$	2. Deposits value between chain identifiers $\left[\omega_{(i+1)_{\mathscr{C}}}, \omega_{j_{\mathscr{C}}}\right]: \lfloor \frac{i-j+1}{2} \rfloor$
3. Stores PRD for a later refund call	3. Stores PRD for a later refund call
4. $j' = i+1$ (the new j becoming as an even index)	4. $j' = i$
Case2. i even:	**Case2.** i even:
1. Deposits value between chain identifiers $\left[w_{i_{\mathscr{C}}}, w_{j_{\mathscr{C}}}\right]: \lfloor \frac{i-j}{2} \rfloor$	1. Deposits value between chain identifiers $\left[\omega_{i_{\mathscr{C}}}, \omega_{j_{\mathscr{C}}}\right]: \lfloor \frac{i-j}{2} \rfloor$
2. $j' = i$	2. $j' = i$
Otherwise. Denies the request	**Otherwise.** Denies the request

Figure 7.3: Corrected `Deposit` protocol after step-by-step checking with CPN (left) vs. flawed `Deposit` protocol (right).

Case1. k odd.

1. Refunds value between chain identifiers $\left[w_{(k-1)\mathscr{C}}, w_{j\mathscr{C}}\right]: \lfloor\frac{k-j-1}{2}\rfloor$
2. $j' = k-1$ (the new j becomes even)
3. If $\mathscr{M}$ has claimed in `Deposit`: $\mathscr{B}$ sends res to $\mathscr{C}$

Case2. k even.

1. Refunds value between chain identifiers $\left[w_{k\mathscr{C}}, w_{j\mathscr{C}}\right]: \lfloor\frac{k-j}{2}\rfloor$
2. $j' = k$
3. If $\mathscr{M}$ has claimed in `Deposit`: $\mathscr{B}$ sends res to $\mathscr{C}$

Otherwise. Denies the request.

Case1. j' odd:

1. Refunds value between chain identifiers $\left[\omega_{k\mathscr{C}}, \omega_{(j'+2)\mathscr{C}}\right]: \lfloor\frac{k-j'-2}{2}\rfloor$
2. $j' = k$
3. Sends lost PRD to $\mathscr{C}$

Case2. j' even:

1. Refunds value between chain identifiers $\left[\omega_{k\mathscr{C}}, \omega_{(j'+1)\mathscr{C}}\right]: \lfloor\frac{k-j'-1}{2}\rfloor$
2. $j' = k$

Otherwise. Denies the request.

Figure 7.4: Corrected `Refund` protocol after step-by-step checking with CPN (left) vs. flawed `Refund` protocol (right).

Automatic Model Checking: Fairness Formal Verification

When we are sure that the model works as predicted and it is correct, we can start the automatic simulation process. As a result, we obtain the complete state space of the system.

We have developed four query functions to search for markings containing probably unfair executions of the micropayment in which either customer or merchant has advantage in front of the other. For example, Figure 7.5 shows the code for the query called `CustomerAdvantage`. It explores the full state space to find markings in which customer $\mathscr{C}$ has advantage in front of merchant $\mathscr{M}$, i.e. executions in which $\mathscr{C}$ has obtained more items than $\mathscr{M}$. The other three queries work in a similar way as `CustomerAdvantage`.

We have begun to formally analyze the micropayment version with resolutions. Figure 7.6 shows that it is fair due to the fact that the query functions applied over the resulting state space do not find any unfair marking. Thus, we have successfully verified formally that the micropayment scheme with resolutions is fair for both customers and merchants.

After the above analysis, we have to analyze the micropayment without considering the resolutions. As a result, Figure 7.7 lists the markings where either the customer or the merchant acquires, at least initially, certain advantage over the other one. Table 7.1 summarizes those situations and outlines who are the parties with advantage in each depicted case.

The following enumeration explains in detail the protocol flow that conducted to the unfair situation of each case shown in Table 7.1:

```
fun CustomerAdvantage(): Node list
= PredAllNodes(
  fn n =>
  let
    val cAccount = ms_to_col(Mark.main'cAccount 1 n)
    val mAccount = ms_to_col(Mark.main'mAccount 1 n)
    val nServicesReceived =size(Mark.main'received_services 1 n)
    val nServicesRefund = size(Mark.main'received_services_refund 1 n)
    val cond2 = cAccount + nServicesReceived
    val totalAccounts = cAccount + mAccount
    val coupons = 2
  in
    if ((Terminal n) andalso (FullyProcessed n)
andalso mAccount<nServicesReceived ) then
      true
    else
      false
  end
)
```

Figure 7.5: Query `CustomerAdvantage`.

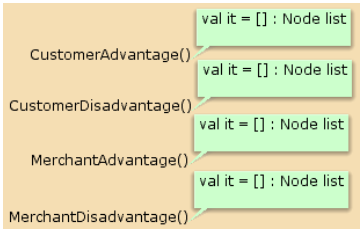


Figure 7.6: Queries do not detect any unfair marking in the micropayment including resolutions and support by $\mathscr{B}$.

Figure 7.7: Queries output some unfair markings in the micropayment without considering resolutions.

Marking 172. Malicious $\mathscr{C}$ stops the first `Spend` transaction after receiving the requested service. $\mathscr{M}$ cannot run the `Deposit` protocol because he only has the first payment microcoupon. $\mathscr{C}$ executes the `Refund` protocol and she obtains a reimbursement equivalent to two microcoupons (the whole coin value). Thus, $\mathscr{C}$ gains advantage over $\mathscr{M}$.

Table 7.1: List of likely unfair states output by CPNTools.

Marking	$\mathtt{v}_{\mathscr{C}}^{\mathtt{acc}}$	$\mathtt{v}_{\mathscr{M}}^{\mathtt{acc}}$	$\mathscr{C}_{rs}$	Advantage for
172	2	0	1	Customer
175	2	0	1	Customer
176	1	1	2	Customer
118	0	2	1	Merchant
177	1	1	0	Merchant

$\mathtt{v}_{\mathscr{C}}^{\mathtt{acc}}$: Value of the customer's account in $\mathscr{B}$.
$\mathtt{v}_{\mathscr{M}}^{\mathtt{acc}}$: Value of the merchant's account in $\mathscr{B}$.
$\mathscr{C}_{rs}$: Number of received services by $\mathscr{C}$ from $\mathscr{M}$.

Marking 175. $\mathscr{M}$ maliciously stops the first `Spend` transaction before sending the service. Then, $\mathscr{M}$ tries to execute the `Deposit` protocol, but he only has the first payment microcoupon, so $\mathscr{B}$ denies it. $\mathscr{C}$ starts a second `Spend` transaction using the third hash identifier (the second payment microcoupon). $\mathscr{M}$ sends the requested service but $\mathscr{C}$ does not send the corresponding proof microcoupon. $\mathscr{M}$ forgets to run the `Deposit` protocol within the allowed time. Finally, $\mathscr{C}$ obtains a refund of both microcoupons (the whole coin value). So, $\mathscr{C}$ gains advantage over $\mathscr{M}$ even though the latter first has played maliciously but afterward $\mathscr{M}$ has harmed himself because he has not run the `Deposit` protocol in time.

Marking 176. The first `Spend` transaction finishes correctly. Then, $\mathscr{M}$ executes the `Deposit` protocol and he obtains the value of the first microcoupon. During the second instance to the `Spend` protocol, $\mathscr{C}$ maliciously stops the exchange after receiving the desired service (she does not send the corresponding proof microcoupon). Therefore, $\mathscr{M}$ cannot request a `Deposit` using only the payment microcoupon without the corresponding proof microcoupon. Finally, $\mathscr{C}$ executes the `Refund` protocol and she obtains a reimbursement of the second microcoupon. So, $\mathscr{C}$ gains advantage over $\mathscr{M}$.

Marking 118. Malicious $\mathscr{M}$ stops the first `Spend` transaction after receiving the payment microcoupon. $\mathscr{C}$ starts a new `Spend` transaction and it finish successfully ($\mathscr{C}$ receives the requested service and $\mathscr{M}$ obtains the microcoupon). $\mathscr{M}$ runs the `Deposit` protocol and obtains a deposit of two microcoupons (the whole coin). Finally, $\mathscr{C}$ executes the

`Refund` protocol but she does not receive a reimbursement as a compensation to the lost service during the first spend transaction. So, $\mathscr{M}$ gains advantage over $\mathscr{C}$.

Marking 177. Malicious $\mathscr{M}$ stops both `Spend` protocol before sending her the requested service. So, $\mathscr{M}$ has up to the third hash identifier (i.e. the payment part from the second microcoupon) without sending any service to $\mathscr{C}$. Then, $\mathscr{M}$ runs the `Deposit` protocol using the third hash identifier and he obtains a deposit of the first microcoupon. $\mathscr{C}$ runs the `Refund` protocol and $\mathscr{B}$ reimburses her with the value of the second microcoupon. Therefore, $\mathscr{M}$ obtains some advantage over $\mathscr{C}$, even though $\mathscr{C}$ is able to recover the value of the last microcoupon.

Analyzing carefully each case, we have detected that there are cases where unfair situations take part due to the malicious behavior of either $\mathscr{C}$ or $\mathscr{M}$. On one hand, in markings 172, 175 and 176, $\mathscr{C}$ obtains advantage over $\mathscr{M}$. However, only markings 172 and 176 are due to the malicious behavior of $\mathscr{C}$ who stops the `Spend` protocol after receiving the desired service from $\mathscr{M}$. Instance 175 is different, because in this case, $\mathscr{M}$ stops the `Spend` protocol (so $\mathscr{M}$ acts maliciously) without sending the service, but $\mathscr{M}$ forgets to execute the `Deposit` protocol within the allowed time. As a result, $\mathscr{C}$ obtains an advantage when she refunds an already used microcoupon. On the other hand, both markings 118 and 177 contain two executions in which $\mathscr{M}$ follows a malicious behavior, stopping the `Spend` protocol after receiving the payment microcoupon. Thus, $\mathscr{M}$ gains advantage over $\mathscr{C}$ in both cases.

However, as stated by the proof to the Lemma 7.1.7, the micropayment scheme without resolutions meets the *controlled fairness* property. As formally proved, in any case in which a single party acquires advantage over the other one can only gain at most a single microcoupon (even we have used two microcoupons along the automatic verification, the *controlled fairness* property will be accomplished in the same way no matters the number of microcoupons belong the coin). So, if we consider acceptable the lose of a single microcoupon with low value in exchange of a simplification and an improvement of the efficiency, the micropayment follows the *controlled fairness* with controlled and limited risk of loss.

***Result* 7.1.1** *µEasyPay solution for purchasing low value goods or services described in Chapter 3 accomplishes all security requirements w.r.t. the micropayment security model described in §3.2.*

7.2 Formal Security Analysis of the Multicoupon Scheme

In this Section we present a formal analysis of our $\mathscr{MC}$-$2\mathscr{D}$ scheme to verify that it fulfills the desired security requirements under the security model defined in §4.2.2.

7.2.1 Unforgeability

Theorem 7.2.1 *Under the RSA factorization problem, the one-wayness and collision resistant of a cryptographic hash function and the SDH assumption of the BBS group signature, the multicoupon scheme described in §4.4 is unforgeable w.r.t. the Definition 4.2.3.*

Proof. We prove that the scheme meets all the considered instances of multicoupon forge provided by the unforgeability games described in §4.2.3. According to that, to forge a multicoupon it is necessary: to forge the partially blind signature scheme, to find a collision in the cryptographic hash function or to forge the group signature scheme.

In order to simplify the proof notation related to the game 0, let us consider a *prover* to refer to the adversary $\mathscr{A}_0$ who tries to use a forged coupon (a malicious customer who redeems a forged coupon) and a *verifier* to denote the honest entity who receives a coupon from the *prover* (a merchant who verifies a redeemed coupon or an issuer who checks a coupon received from a merchant).

Besides, let us suppose the *prover* tries to forge a multicoupon using a coupon $(\omega_{i,j})$ beyond the limits of the multicoupon, i.e. a coupon such as its index is either $i > N_j$ or $i \leq 0$. By construction, the multicoupon identifiers $\left(\left[\omega_{0,j}\right]_{j=1}^{J}\right)$ are blinded and linked to the common data during the `Issue` protocol. The common data contains the number of coupons between a given multicoupon identifier $(\omega_{0,j})$ and the corresponding seed $(\omega_{N_j,j})$, as well as other parameters. $\mathscr{I}$ signs these data using a partially blind signature scheme and as a result, $\mathscr{C}$ obtains locally $\mathbb{MC}^{2D}$. Then, the *verifier* can compare the provided index with the number of coupons bounded in $\mathbb{MC}^{2D}$. It is easy to prove that whether the presented index is $i > N_j$ or $i \leq 0$, the *verifier* does not accept the coupon.

Another forge attempt arises whether the *prover* tries to use a coupon $(\omega_{i,j})$ beyond the limits of the multicoupon, as before, but providing an index within the allowed range $(0 < i \leq N_j)$. By hash chain properties, the *verifier* can check whether the provided coupon is included in the multicoupon applying iteratively i times the hash function on it. The *verifier* compares the result with the corresponding multicoupon identifier $(\omega_{0,j})$ and if they are not the same, the *verifier* concludes that the provided coupon does not belong $\mathbb{MC}^{2D}$.

The *prover* could try to use a coupon not belonging $\mathbb{MC}^{2D}$ without the *verifier* detecting it. It would mean that the hash function has been broken, due to the fact that

the *prover* has been found a collision. Since our scheme uses a secure cryptographic one-way hash function with collision resistance, this forge instance, based on finding a collision or a preimage of a hash value, has negligible probability to happen.

Additionally, the *prover* could try to modify some of the parameters bounded in the common data, e.g. increase the number of coupons in a multicoupon, modify their value or extend some of the provided time marks. The modification of the common data implies the modification of $\mathbb{MC}^{2D}$. Thus, when the *verifier* checks the partially blind signature through the use of $Verify^{\mathcal{PBS}}$, the *verifier* detects that the $\mathbb{MC}^{2D}$ has been modified, since the equation does not hold. So, the *prover* cannot modify the $\mathbb{MC}^{2D}$ without it being detected by the *verifier* due to the fact that the partially blind signature scheme is unforgeable.

Finally, let us suppose a *prover* trying to use a coupon from a self-issued $\mathbb{MC}^{2D}$. In this case, the *verifier* can detect that $\mathbb{MC}^{2D}$ has been issued by an unrecognized $\mathcal{I}$. Thus, the *verifier* does not accept $\mathbb{MC}^{2D}$.

Regarding to the game 1, let us first remember that during the `Issue` protocol, $\mathcal{I}$ knows how many coupons have been issued ($\mathcal{Z}_I$), since one of the parameters included in every public common data is the number of coupons within each multicoupon (and their corresponding value, $\left[(N_j, v_j)\right]_{j=1}^{J}$). $\mathcal{A}_1$ can try to modify the common data in order to increase the number of coupons available in $\mathbb{MC}^{2D}$. However, since the common data is partially blind signed during the `Issue` protocol, it cannot been modified without being detected using the $Verify^{\mathcal{PBS}}$ procedure, unless the partially blind signature scheme has been forged. But it is not possible to happen with non-negligible probability due to the fact that the used partially blind signature is unforgeable based on the RSA factorization problem (see §3.3 from [55]). Thus, it is not possible with non-negligible probability that an adversary $\mathcal{A}_1$ can claim more coupons than the number of coupons issued by $\mathcal{I}$.

Moreover, $\mathcal{A}_1$ cannot claim more coupons than the number of coupons redeemed by honest customers ($\mathcal{Z}_R$), since each coupon received from a particular $\mathcal{C}$ is group signed by the use of the group signature scheme. So, $\mathcal{A}_1$ should either modify the group signature (to fake and increase the number of coupons redeemed by honest customers) or find a collision in the hash function (to find a coupon not rightfully allowed). On one hand, the former forge instance is not possible due to the use of the BBS group signature, which is unforgeable under the SDH assumption (see §3.1 and Definition 3.1 from [115]). On the other hand, the latter forge attempt is also not possible because of collision resistance property of the cryptographic hash function.

Formally, if an adversary $\mathcal{A}(\mathcal{A}_0, \mathcal{A}_1)$ can forge $\mathbb{MC}^{2D}$, as exposed in §4.2.3, it means that the adversary $\mathcal{A}$ has success trying to forge the partially blind signature, trying to find a collision of a cryptographic hash function or trying to forge the BBS group signature. In order to forge the used $\mathcal{PBS}$ scheme, since it is based on the RSA factorization problem, the adversary have to know the factorization of n. It

means that $\mathscr{A}$ needs to break the discrete logarithm problem, which it is assumed to be unfeasible (in K). As we use a secure one-way collision resistant hash function, it is not possible by an adversary to compute a collision in the hash function with non-negligible probability (in K). Finally, the forge of the group signature scheme implies to breaking the SDH assumption, which is considered hard if group settings are carefully selected (refer to [46]). Then, it follows that $\mathscr{A}(\mathscr{A}_0, \mathscr{A}_1)$ cannot win any forge game with non-negligible probability.

***Remark* 7.2.1** *$\mathscr{C}$ cannot redeem more coupons than they have been rightfully allowed, issue new multicoupons or modify its content, and $\mathscr{M}$ cannot claim more coupons than the number of issued coupons by $\mathscr{I}$ or claim more coupons than the number of coupons redeemed by customers.*

7.2.2 Unlinkability

Theorem 7.2.2 *Under both the Decision Linear problem from the BBS group signature scheme and the unlinkability of the partially blind signature scheme, the multicoupon scheme presented in §4.4 is 2D-unlinkable w.r.t. the Definition 4.2.4.*

Proof. First, different $\mathbb{MC}^{2D}$ structures have different and unrelated multicoupon identifiers ($[\omega_{0,j}]_{j=1}^{J}$). Thus, two different coupons from different $\mathbb{MC}^{2D}$ ($\mathbf{c}_j(m) \in \mathbb{MC}_{\omega}$ and $\mathbf{c}_j^{'}(m) \in \mathbb{MC}_{\omega}^{'}$, $\forall j$) cannot be linked. It is easy to prove, due to the fact that multicoupons are generated applying iteratively the hash chain procedure over different and randomly chosen seed identifiers ($\omega_{N_j,j}$ are uniformly distributed over $\mathbb{Z}_r$). As a result, their corresponding multicoupon identifiers will be different and unrelated ($[\omega_{0,j}]_{j=1}^{J} \neq [\omega_{0,j}^{'}]_{j=1}^{J}$).

Regarding to the unlinkability game 0, we need to take into account that $\mathscr{C}$ signs each set of released coupons during the `Multiredeem` protocol using the BBS group signature scheme, with her group member private key ($sk_{\mathscr{C}}^{G}$). So, even coupons from different multicoupons are not related due to the above proof, it must be also unfeasible to link two group signed coupons released by the same $\mathscr{C}$ who uses, obviously, the same group member private key. Note that elements output by the $Sign^G$ algorithm are computed based on other elements randomly chosen from $\mathbb{Z}_q$ (the size of $\mathbb{Z}_q$ depends on the size of the security parameter K), so all of the elements of a signature are uniformly distributed along the input set $\mathbb{Z}_q$. By the Decision Linear assumption (we refer the reader to §3.2 and Theorem 4.1 from [115]), it can be proven that elements contained in two different group signatures computed by the $Sign^G$ algorithm are indistinguishable. In a formal way, if the adversary $\mathscr{A}_0$ win the unlinkability game 0, it means that $\mathscr{A}_0$ can link two group signatures applied on two coupons released by $\mathscr{C}$. However, it is not possible due to no p.p.t adversary

$\mathcal{A}_0$ can win the game with non-negligible linkability advantage under the Decision Linear problem (in K).

Concerning the unlinkability game 1, we prove that the multicoupon scheme does not allow the linkability of an `Issue` run with a `Multiredeem` run. It is the same as proving that the $\mathbb{MC}_{\mathcal{PBS}}$ element, used during the `Multiredeem` execution, cannot be linked to the corresponding `Issue` process. Let us consider that $\mathcal{A}_1$ has access to an issuing view provided by a corrupted issuer ($\mathcal{V}_{\mathcal{I}}^{Issue}$). Note that only $\mathcal{C}$ knows the final result of an `Issue` protocol: the $\mathbb{MC}_{\mathcal{PBS}}$ is computed locally by $\mathcal{C}$, thanks to the last step of the protocol (unblinding). Thus, $\mathcal{I}$ does not know the $\mathbb{MC}^{2D}$ identifier ($[\omega_{0,j}]_{j=1}^{J}$). As a consequence, $\mathcal{A}_1$ neither knows this identifier, since it is not included within $\mathcal{V}_{\mathcal{I}}^{Issue}$. Now, consider $\mathcal{A}_1$ has also executed a `Multiredeem` with a customer $\mathcal{C}'$ (without knowing whether it is the same as $\mathcal{C}$ or not), obtaining a new view $\mathcal{V}_{\mathcal{A}_1}^{Multiredeem}$. $\mathcal{A}_1$ can explore data collected in $\mathcal{V}_{\mathcal{I}}^{Issue}$ and $\mathcal{V}_{\mathcal{A}_1}^{Multiredeem}$, trying to found a link between $\mathbb{MC}_{\mathcal{PBS}}$ and the corresponding `Issue`. If $\mathcal{A}_1$ can link $\mathbb{MC}_{\mathcal{PBS}}$ with its `Issue` execution, it means that $\mathcal{A}_1$ can break the unlinkability of the underlying partially blind signature. However, it only happens with negligible probability (see unlinkability proof §3.4 from [55]), so we have proved that no p.p.t. adversary $\mathcal{A}_1$ can win the unlinkability game 1 with non-negligible advantage (in K).

***Remark* 7.2.2** *Hence, we achieve an $\mathcal{MC}$-2$\mathcal{D}$-unlinkable scheme, since an adversary $\mathcal{A}(\mathcal{A}_0,\mathcal{A}_1)$ cannot link neither two `Multiredeem` instances nor an `Issue` together with a `Multiredeem` run with non-negligible advantage (in K).*

7.2.3 Anonymity of Customers

Theorem 7.2.3 *Under the full-anonymity (CPA-full-anonymity[1]) of the underlying BBS group signature, the multicoupon scheme described in §4.4 is anonymous w.r.t. the Definition 4.2.5.*

Proof. $\mathbb{MC}_{\mathcal{PBS}}$ conveys two types of data: common information (expiration time, number and value of coupons), which can be read by anyone and it does not contain data about its owner; and blind data (multicoupons identifiers $[\omega_{0,j}]_{j=1}^{J}$). Then, no customer identification data is contained in the $\mathbb{MC}_{\mathcal{PBS}}$.

Firstly, let us consider an adversary $\mathcal{A}_0$ (operating as a malicious $\mathcal{I}$) who tries to identify a customer exploring data in $\mathcal{V}_{\mathcal{A}_0}^{Issue}$. It is trivial to prove that $\mathcal{A}_0$ cannot obtain any useful data to do that (with non-negligible probability), since $\mathcal{C}$ does not provide any type of information related to her identity during the `Issue` protocol.

[1]The BBS short group signature is proven under the CPA-full-anonymity model defined in [115], where an adversary cannot query the openning oracle.

Secondly, consider an adversary $\mathscr{A}_1$ (acting as a malicious $\mathscr{M}$ or a coalition of them) trying to identify a customer processing the $V_{\mathscr{A}_1}^{Multiredeem}$ filled during a run of the `Multiredeem` protocol. When $\mathscr{C}$ sends either a single coupon or a set of coupons (either A_{l_j,j,n_j} or $B_{p_j,j}$) to $\mathscr{A}_1$, she provides $\mathscr{A}_1$ these coupons group signed by her group private key (sk_c^G). Therefore, $\mathscr{A}_1$ can only know whether the provided signature is valid and made by a customer who belongs a group of customers using the $Verify^G$ algorithm. It is due to the definition and the proof provided by Theorem 5.2 of [115], stating that there is no adversary who can break the group signature anonymity (since it is not possible to break the semantic security of Linear encryption) with non-negligible probability.

As a result, neither $\mathscr{A}_0$ nor $\mathscr{A}_1$ can infer the identity of an honest $\mathscr{C}$ with non-negligible probability. Note that, $\mathscr{G}$ is the only entity who can revoke the anonymity and disclose the identity of $\mathscr{C}$ using the $Open^G$ procedure from the BBS group signature scheme.

7.2.4 Coupon Reuse Avoidance

Theorem 7.2.4 *Under the unforgeability of the BBS group signature scheme the multicoupon scheme presented in §4.4 allows detecting and avoiding coupon reuse w.r.t. the Definition 4.2.6.*

Proof. $\mathscr{I}$ does not require to store any information about the $\mathbb{MC}^{2D}$ during the `Issue` protocol, because the $\mathbb{MC}^{2D}$ has been signed with $\mathscr{I}$'s RSA private key and it also contains all the required information for its verification. However, when $\mathscr{M}$ claims to $\mathscr{I}$ in exchange to a set of redeemed coupons from customers, $\mathscr{I}$ stores data about these already used coupons. Now, when $\mathscr{C}$ redeems some coupons in a particular $\mathscr{M}$, she sends $\mathscr{M}$ the set of data associated to that coupons, that is, the payment and proof information along the pair of merchant and redeem identifiers (i.e. $[id_{\mathscr{M}}, id_r]$). All this information is group signed by $\mathscr{C}$ $\left(\text{both } \mathbb{SG}_{\mathscr{C}}\left(A_{l_j,j,n_j}, id_{\mathscr{M}}, id_r\right) \text{ and } \mathbb{SG}_{\mathscr{C}}\left(B_{p_j,j}, id_{\mathscr{M}}, id_r\right)\right)$, and it is used by $\mathscr{M}$ to obtain each individual coupon and check whether any single coupon has been already redeemed, i.e., if any coupon is either in the $\mathscr{M}$'s local database or in the $\mathscr{I}$'s global database (on-line verification).

At this point, we can differentiate two cases. First, taking into account the game 0 (as defined in §4.2.6), an adversary $\mathscr{A}_0$ can try to redeem an already spent coupon with an honest merchant $\mathscr{M}$ (with identifier $id_{\mathscr{M}}$). In this case, $\mathscr{M}$ can detect it and prove the misbehavior of $\mathscr{A}_0$, because each run of the redeem protocol is uniquely identified by the pair of identifiers $[id_{\mathscr{M}}, id_r]$. Thus, the use of the same coupons information $(A_{l_j,j,n_j} - B_{p_j,j})$ at the same merchant in a different redeem run ($id_r \neq id_r^{'}$), proves that $\mathscr{A}_0$ is misbehaving, with non-negligible probability. Similarly, if

$\mathcal{A}_0$ tries to reuse the same set of coupons in different merchants, the system detects it because the information (different $id_{\mathcal{M}}$ and the same coupons information) is group signed by the adversary, and thus $\mathcal{A}_0$ is the dishonest participant.

Secondly, in the game 1 (as defined in §4.2.6), an adversary $\mathcal{A}_1$ (with identifier $id_{\mathcal{M}}$) attempts to reuse a set of coupons during a `Claim` protocol run. Now, $\mathcal{A}_1$ must provide to $\mathcal{I}$ with the set of coupons along the pair of identifiers $[id'_{\mathcal{M}}, id'_r]$ signed by the honest $\mathcal{C}$. If $id'_{\mathcal{M}} = id_{\mathcal{M}}$ and $id'_r = id_r$, it proves $\mathcal{A}_1$ misbehavior. Hence, if $\mathcal{A}_1$ tries to modify id'_r to involve $\mathcal{C}$ in an attempt of coupon reuse, $\mathcal{A}_1$ must break the unforgeability of the scheme. As proved by Theorem 7.2.1, $\mathcal{A}_1$ cannot forge either a multicoupon or a coupon with non-negligible probability (in `K`). $\mathcal{A}_1$ could collude with another malicious $\mathcal{M}'$, but in this case $\mathcal{M}'$ should prove he has assigned the identifier $id'_{\mathcal{M}} = id_{\mathcal{M}}$ bound to the presented coupons. Therefore, in any case, through the anonymity revocation mechanism, $\mathcal{C}$ cannot be charged with the coupon reuse (see §7.2.5).

7.2.5 Anonymity Revocation of Misbehaving Customers

Theorem 7.2.5 *Under both the unforgeability and the full-traceability property of the underlying BBS group signature scheme, the multicoupon scheme presented in §4.4 allows the anonymity revocation and the correct identification of customers trying to redeem either forged coupons or coupons already redeemed, w.r.t. the Definition 4.2.7.*

Proof. Due to the unforgeability (already discussed in §7.2.1) and the *full-traceability* property of the BBS group signature (according to the Theorem 5.3 and the corresponding proof provided in [115]), our multicoupon scheme ensures that all signatures made on coupons, even those created by an adversary $\mathcal{A}$ composed by the collusion of multiple customers (including probably also a malicious merchant or the group manager), trace to the cheating customer of the coalition. Thus, using the algorithm $Open^G$ provided by the BBS group signature and executed by the group manager, an adversary $\mathcal{A}$ can only avoid to be correctly identified with negligible probability, even they are trying to unfairly blame an honest customer. As a consequence, the multicoupon scheme also provides a non-frameability protection to honest customers.

7.2.6 Merchant disaffiliation

Lemma 7.2.6 *Due to the use of an affiliation revocation list ($\mathcal{AL}$) and the fact that merchants do not share sensitive information with the issuer about both customers and the data required to generate multicoupons, the system presented in §4.4 al-*

lows a secure either voluntary or forced disaffiliation of a merchant from the issuer, without it compromising the security of the system, w.r.t. the Definition 4.2.8.

Proof. $\mathscr{M}$ only needs to know $\mathscr{I}$'s public key (e, n) and group's public key (pk^G) in order to work in the system. Therefore, merchants do not have either information about customers or another sensitive information, such as the private key used to issue $\mathbb{MC}^{2D}$. As stated in the definition, two types of disaffiliation can be defined and proved secure as follows.

On one hand, if $\mathscr{M}$ is no more interested in belonging the affiliation and accepting multicoupons issued by $\mathscr{I}$, he can request his disaffiliation. The process is as simple as removing the merchant identifier $id_{\mathscr{M}}$, assigned to $\mathscr{M}$ at the affiliation stage, from the affiliation list ($\mathscr{AL}$), due to the fact that merchants do not either have information about customers or share data with other merchants from the affiliation.

On the other hand, considering an adversary $\mathscr{A}$ as a misbehaving merchant, $\mathscr{I}$ can force easily the revocation of the affiliation of $\mathscr{A}$. $\mathscr{I}$ can remove $id_{\mathscr{M}}$ from the affiliation list ($\mathscr{AL}$), and include $id_{\mathscr{M}}$ in a affiliation revocation list ($\mathscr{ARL}$).

In both cases, when $\mathscr{M}$ tries to claim a set of coupons, he must be authenticated by $\mathscr{I}$. Moreover, the data provided by $\mathscr{M}$ contains his $id_{\mathscr{M}}$ signed by $\mathscr{C}$ at the redeem phase. Thus, $\mathscr{I}$ can check the affiliation list ($\mathscr{AL}$) and if it is applicable, $\mathscr{I}$ denies the claim to $\mathscr{M}$.

Whether the disaffiliation has been voluntary or forced, if a merchant is still accepting coupons from customers after his disaffiliation, this merchant cannot claim more coupons, since $\mathscr{I}$ will deny the claim request due to the fact that $\mathscr{M}$ is no longer affiliated. However, it will be only a loss for $\mathscr{M}$ not for customers. As a result, merchants can leave the system without it being compromised regardless of the disaffiliation was either voluntarily or forced.

7.2.7 Unsplitting

Lemma 7.2.7 *Due to the hash chain used in the coupon generation of a same multicoupon, the system presented in §4.4 discourages customer to split and share coupons, and to receive split coupons from another customer without trust each other, because this procedure could compromise the anonymity of a not dishonest customer, according to the Definition 4.2.9.*

Proof. To prove this lemma, we follow both games described in *Definition 4.2.9*. For the sake of simplicity and without loss of generality, we assume in the following proofs that $\mathbb{MC}^{2D}$ is composed by a single multicoupon, i.e. j will be always $j = 0$.

During the game 0, an adversary $\mathscr{A}_0$ obtains a valid $\mathbb{MC}^{2D}$ from $\mathscr{I}$. Then, $\mathscr{A}_0$ extracts a set of coupons $\mathscr{L}_s$ from $\mathbb{MC}^{2D}$ (i.e. $\mathscr{L}_s \equiv [\omega_{i,0}]_{i=k}^{k+s}$) and gives this list to an honest customer $\mathscr{C}_0$. Before $\mathscr{C}_0$ can redeem some of the received coupons, $\mathscr{A}_0$

redeems a single coupon $\omega_{r,0} \in \mathscr{L}_s$. Now, $\mathscr{C}_0$ tries to redeem some of the received coupons, one of them the coupon $\omega_{r,0}$ recently used by $\mathscr{A}_0$. The merchant who receives the coupons checks their validity contacting with the issuer who detects the reuse of $\omega_{r,0}$ (see §4.2.6). As $\omega_{r,0}$ is group signed by the honest customer $\mathscr{C}_0$, it is clear that from the point of view of $\mathscr{I}$, $\mathscr{C}_0$ is guilty of coupon reuse (see §7.2.5) even though $\mathscr{A}_0$ is who really behaved improperly. As a result, the revocation mechanism of the group signature could declare $\mathscr{C}_0$ unfairly guilty of reusing a coupon.

During the game 1, an adversary $\mathscr{A}_1$ receives a set of coupons $\mathscr{L}_s$ from an honest customer $\mathscr{C}_1$, who previously obtained a valid $\mathbb{MC}^{2D}$ from $\mathscr{I}$. $\mathscr{A}_1$ could try to spend a valid coupon not included in the list of shared coupons, $\omega_{r,0} \notin \mathscr{L}_s$, but included in the multicoupon from which the coupons were extracted, i.e. $\omega_{r,0} \in \mathbb{MC}^{2D}$. In this context, we can distinguish two cases.

In the first one, $\mathscr{A}_1$ should obtain a coupon with index $i > k + s$, but this process implies that $\mathscr{A}_1$ has forged a coupon. However, $\mathscr{A}_1$ has negligible probability of success, as already discussed in §7.2.1.

In the second one, $\mathscr{A}_1$ could try to use a coupon with index $0 \leq i < k$. By hash chain properties, we know $\omega_{k,0}$ is obtained applying iteratively s times the hash function on $\omega_{k+s,0}$. So, as $\mathscr{A}_1$ knows $\omega_{k,0}$ she can easily generate coupons with index $0 \leq i < k$. Therefore, $\mathscr{A}_1$ could redeem a valid coupon included in $\mathbb{MC}^{2D}$ but not really shared by $\mathscr{C}_1$. Later, $\mathscr{C}_1$ tries to redeem legitimately this not shared coupon, but already used improperly by $\mathscr{A}_1$. The merchant who receives the coupon checks its validity contacting with the issuer, who detects the reuse of that coupon (see §4.2.6). As this coupon is group signed by $\mathscr{C}_1$, as in the game 0, the anonymity revocation procedure will declare $\mathscr{C}_1$ unfairly guilty of coupon reuse (see §7.2.5) even though who behaves improperly is $\mathscr{A}_1$.

Note that this second case can be avoided if $\mathscr{C}_1$ shares with another customer a set of coupons where the first shared coupon ($\omega_{k,0}$) is the first unused coupon by $\mathscr{C}_1$ (i.e. $\omega_{k-1,0}$ is the last used coupon from $\mathbb{MC}^{2D}$ by $\mathscr{C}_1$).

Hence, if customers want either to split and share some coupons from their $\mathbb{MC}^{2D}$, or to receive split coupons, they assume the risk to be unfairly accused of fraudulent behavior (through the anonymity revocation of the group signature over a set of reused coupons) if they do not trust each other. As a result, our scheme discourages the multicoupon splitting as *weak unsplittability* does.

***Result* 7.2.1** *$\mathscr{MC} - 2\mathscr{D}$ proposal for the multi-merchant scenario to acquire goods or services with probably discounts or gifts described in Chapter 4 fulfills all security properties w.r.t. the electronic multicoupon security model described in §4.2.*

7.3 Security Analysis of the Automatic Fare Collection System

In this Section we describe the security analysis of the AFC scheme described in Chapter 5 to prove that it achieves the required security properties from the security model presented in §5.2.2. Due to the fact that the AFC and the $\mathscr{MC}-2\mathscr{D}$ multicoupon schemes (the latter validated in §7.2) share the use of a group signature scheme (such as the BBS scheme), some of the following proofs utilize previous arguments already given in proofs related to the $\mathscr{MC}-2\mathscr{D}$ scheme.

7.3.1 Unforgeability

Theorem 7.3.1 *Under the use of a secure public key crytosystem and the SDH assumption of the BBS group signature, the general AFC scheme described in §5.4, as well as the modified and enhanced versions defined in §5.6 and §5.7, are unforgeable w.r.t. the Definition 5.2.3.*

Proof. Providers sign both entrance and exit tickets with their own private keys from a secure public key cryptosystem, i.e. $\mathsf{t_{in}}^* = \left(\mathsf{t_{in}}, \mathbb{S}_{\mathscr{P}_\mathscr{J}}(\mathsf{t_{in}})\right)$ and $\mathsf{t_{out}}^* = \left(\mathsf{t_{out}}, \mathbb{S}_{\mathscr{P}_\mathscr{D}}(\mathsf{t_{out}})\right)$.

First, consider that $\mathscr{A}_0$ plays the unforgeability game 0, in which she tries to modify some parameters enclosed in a ticket received from a provider. Then, if $\mathscr{A}_0$ is able to validate the ticket without verifier (i.e. providers) detecting it, it means that $\mathscr{A}_0$ has broken the public key cryptosystem. However, it is not possible to happen with non-negligible probability under the assumption we use a secure public key cryptosystem for signing (in $\mathtt{K}$).

Secondly, regarding to the unforgeability game 1, $\mathscr{A}_1$ tries to self-issue an entrance ticket using her own credentials. However, the probability providers do not detect the fact that the presented ticket is signed by an unrecognized entity is negligible considering the use of a secure signing algorithm (in $\mathtt{K}$).

Similarly, consider the unforgeability game 2, in which $\mathscr{A}_2$ tries to modify messages coming from users. That messages are group signed by her group credentials, so they are also unforgeable due to the use of a secure group signature scheme. Indeed, to forge a message group signed, it is necessary to break the SDH assumption, which is considered hard under a correct choice of group settings (as stated in §7.2.1). So, the probability for an adversary to forge a group signature is negligible (in $\mathtt{K}$).

***Remark* 7.3.1** *Due to the fact that entities sign messages they issue (providers and managers use a secure public key cryptosystem for signing, while users sign by using a secure group signature scheme), messages are authentic and cannot be modified without verifier detecting it, so messages achieve also the integrity property.*

7.3.2 Non Repudiation of Origin

Theorem 7.3.2 *Under the use of a secure public key cryptosystem (a digital signing algorithm and the BBS group signature), the general AFC scheme described in §5.4, as well as the modified and enhanced versions defined in §5.6 and §5.7, provide NRO w.r.t. the Definition 5.2.4.*

Proof. Service providers (either $\mathscr{P}_{\mathscr{S}}$ or $\mathscr{P}_{\mathscr{D}}$) sign every issued ticket with their own private key from a secure public key cryptosystem. Each public key and the attached identity of the signing entity is properly certified by a CA trusted by all involved parties. Therefore, only authorized and recognized parties can issue valid tickets.

Similarly, even users computing group signatures are anonymous, if their identity is revoked by $\mathscr{T}_G$, then they cannot deny the signature authorship due to the fact that the revocation algorithm only gives the incorrect identity of the signer with negligible probability (in $\mathtt{K}$).

Thus, by non repudiation properties of a secure electronic signature scheme, the entity who has computed a signature cannot deny its authorship. If an adversary can deny the authorship, it would mean that the signature scheme has been compromised or the ticket has been forged. As already proved in §7.3.1, there is no p.p.t. adversary that can forge a ticket with non-negligible probability (in $\mathtt{K}$) under the use of secure public key cryptosystem. Similarly, any p.p.t. adversary can deny the authorship of a group signature due to the anonymity revocation algorithm. So, it follows that there is no p.p.t. adversary able to deny the authorship of a signature with non-negligible probability (in $\mathtt{K}$).

7.3.3 Anonymity

Theorem 7.3.3 *Under the full-anonymity (CPA-full-anonymity) of the BBS group signature, the AFC scheme described in §5.4, as well as the modified and enhanced versions defined in §5.6 and §5.7, provide anonymity to users w.r.t. the Definition 5.2.5.*

Proof. Neither entrance ticket nor exit ticket carry some information related to the real identity of $\mathscr{U}$.

First, consider an adversary $\mathscr{A}_0$ acting as a malicious provider (either source or destination provider) trying to identify the user who executes the `SysEntrance` and `SysExit` protocols. Information provided by $\mathscr{U}$ is group signed by her own group key pair. Therefore, $\mathscr{A}_0$ can only know whether $\mathscr{U}$ belongs a group of users. In case $\mathscr{A}_0$ has obtained the identity of $\mathscr{U}$ it would mean that $\mathscr{A}_0$ has had success in breaking the group signature anonymity. However, it only happens with non-negligible probability (in $\mathtt{K}$) (as proved in §7.2.3).

Besides, the information related to $\mathscr{U}$ is encrypted with the payment manager public key. Therefore, under the use of a secure encryption algorithm, $\mathscr{A}_0$ has negligible probability to decrypt this information.

Secondly, consider an adversary $\mathscr{A}_1$ with the role of a corrupted payment manager. As already stated above, information provided by $\mathscr{U}$ to authorize the payment of the fare is encrypted by the public key of $\mathscr{T}_P$. So, $\mathscr{T}_P$ can decrypt it and charge the right user. However, $\mathscr{T}_P$ cannot identify the real identity of $\mathscr{U}$ because $\mathscr{T}_P$ only has the pseudonym $y_{\mathscr{U}}$. $\mathscr{U}$ proves that she is a valid user through a secure ZKP algorithm (such as Schnorr ZKP). During this process, $\mathscr{U}$ proves knowledge of the secret element $x_{\mathscr{U}}$ $\left(\text{such that } y_{\mathscr{U}} = \alpha^{x_{\mathscr{U}}} \pmod{p}\right)$ while she is remaining anonymous in front of $\mathscr{T}_P$. The only entity who can link the real identity of $\mathscr{U}$ with her pseudonym is $\mathscr{T}_G$.

Thus, it follows that there are no p.p.t. adversary $\mathscr{A}(\mathscr{A}_0, \mathscr{A}_1)$ who can know the real identity of users using the service, with non-negligible probability (in K).

7.3.4 Unlinkability

Theorem 7.3.4 *Under the Decision Linear problem from the BBS group signature scheme, the enhanced AFC scheme described in §5.7 only allows the linkability of two group signatures w.r.t. the Definition 5.2.6.*

Proof. Even the BBS group signature scheme generates unlinkable group signatures, because the use of the Decision Linear assumption (as proved in §7.2.2), the enhanced AFC scheme has to be able to link two group signatures: the first one computed during the `SysEntrance` protocol and the second one during the `SysExit` protocol. The method proposed in §5.7.1 allows it by reusing some random elements computed by the first group signature instance. However, it is remaining not possible for a p.p.t. adversary to link two group signatures performed in different journeys with non-negligible advantage over random guess (in K). Note that group signatures generated for different journeys always use fresh random elements.

***Remark* 7.3.2** *As providers can link the exit ticket with the corresponding entrance ticket, and verify whether both group signatures have been computed by the same user (even she remains anonymous), the enhanced AFC scheme described in §5.7 avoids committing fraud with the colluding attack exposed in §5.6.1.*

7.3.5 Untraceability

Theorem 7.3.5 *Under both the anonymity of users and unlinkability of different journeys proofs, the AFC scheme described in §5.7 is untraceable w.r.t. the Definition 5.2.7.*

Proof. It is not possible for any party in the system (but $\mathscr{T}_G$) to link actions made by an user and at the same time trace her real identity to that actions. The proof of untraceability follows from above proofs about anonymity of users and unlinkability of tickets made in different journeys.

Note that even $\mathscr{T}_P$ can link different payments made by a single user using the pseudonym $y_{\mathscr{U}}$, it is unfeasible for $\mathscr{T}_P$ to assign journeys to the real identity of the user who owns the pseudonym $y_{\mathscr{U}}$, because $\mathscr{T}_P$ has no other additional information to infer her real identity.

7.3.6 Overspending Avoidance

Lemma 7.3.6 *The AFC scheme presented in §5.4, as well as the modified and enhanced versions defined in §5.6 and §5.7, avoid users either to reuse a ticket already validated or to use a ticket out of the allowed validity time w.r.t. the Definition 5.2.8.*

Proof. On one hand, if $\mathscr{U}$ tries to overspend an entrance ticket, the serial number will be marked as already used. If this behavior can be proved, the group manager $\mathscr{T}_G$ can include $\mathscr{U}$ to the revocation list.

On the other hand, the destination provider $\mathscr{P}_{\mathscr{D}}$ receives the entrance ticket from $\mathscr{U}$ in order to be verified. In this verification, the current time is compared to the validity time τ_v of the entrance ticket $\mathsf{t_{in}}^*$ (signed by $\mathscr{P}_{\mathscr{S}}$). If ticket is no longer valid, the AFC system can take actions, such as anonymity revocation, apply a fine, etc.

7.3.7 Anonymity Revocation of Misbehaving Users

Theorem 7.3.7 *Under both the unforgeability and the full-traceability property of the BBS group signature scheme, the AFC scheme described in §5.4 as well as the modified and enhanced versions defined in §5.6 and §5.7, allows the anonymity revocation and the correct identification of users trying to validate forged tickets, tickets already validated or trying to difficult the process of charging the computed fare, w.r.t. the Definition 5.2.9.*

Proof. On one hand, when $\mathscr{U}$ computes group signatures by the use of the BBS group signature scheme, we refer to the anonymity revocation proof given in §7.2.5 (related to the $\mathscr{MC}-2\mathscr{D}$ multicoupon scheme), due to the fact that it can be applied also to the AFC scheme in that circumstances.

On the other hand, consider the case in which $\mathscr{U}$ misbehaves sending wrong information in order to difficult the process of charging her account in $\mathscr{T}_P$. As already pointed out by `PClaim1` and `PClaim2` protocols, if $\mathscr{U}$ has a malicious behavior, $\mathscr{T}_P$ cannot know what is the account related to that user. However, $\mathscr{U}$ had provided her pseudonym $y_{\mathscr{U}}$ to the group manager $\mathscr{T}_G$ when she had registered to the

system. In case of aforementioned problems, $\mathscr{T}_P$ can ask $\mathscr{T}_G$ for the anonymity revocation of $\mathscr{U}$. As $\mathscr{T}_G$ has the link between $\mathscr{U}$'s identity, her private group key and her pseudonym $y_{\mathscr{U}}$, $\mathscr{T}_G$ can provide the pseudonym $y_{\mathscr{U}}$ (after executing the $Open^G$ algorithm) to $\mathscr{T}_P$ in order to proceed to charge the cheating $\mathscr{U}$.

***Result* 7.3.1** *AFC solutions for time-based and distance-based fares for charge users according to the use of services presented in Chapter 5 accomplish all security requirements w.r.t. the Automatic Fare Collection security model described in §5.2. In addition, the distance-based AFC scheme is not vulnerable to the colluding attack in scenarios with entrance and exit stations not separated by direction.*

Part IV

Implementations and Performance Analysis

CHAPTER 8

CONSIDERATIONS ON THE DEPLOYMENT OF SOLUTIONS AND IMPLEMENTATION FRAMEWORK

The main objective that researchers should achieve, in addition to prove whether their solutions are actually secure, is also to prove the practicability and utility of these solutions when they have to be used by users. Therefore, real software implementations and deployments should be very important in order to demonstrate whether the proposed schemes are in fact suitable to be used and performs well in scenarios with real and current devices and communications. In this Chapter we give some important considerations that should be kept in mind when e-commerce solutions in general are object of implementation and deployment to a real scenario. Then, we also describe the implementation framework used to develop the code for the privacy-protecting solutions presented in Part II.

8.1 Introduction

The final objective of researchers after proposing solutions and proving if they are actually secure, is to also prove if these solutions are in fact practical when they are used in a real deployment scenario. In fact, we are confident with the idea that implementations and performance evaluations are almost as important as security

analysis, and even the solution itself. It is due to the fact that even if a solution is proved more secure than previous ones, in case it cannot be proved either practical or implementable, this solution will not be used by anyone. Thus, our objective with performance evaluations of presented privacy-protecting solutions in this dissertation is to prove that they are also efficient and affordable to be used in real scenarios composed by current mobile devices.

In this Chapter we present some interesting considerations to take into account when general e-commerce solutions (not only for our solutions) have to be implemented and deployed to a real scenario, such as the server and client platforms, the communication technologies and the encoding and syntax of transferred messages. After these general considerations, we provide our choices among all the available options to describe the implementation framework to afterward develop and evaluate the performance of the privacy-protecting solutions presented in Part II.

This Chapter is organized as follows. In Section 8.2 we give some considerations to keep in mind when an e-commerce solution should be deployed. We also select technologies and solutions that fit better to implement and evaluate our privacy-protecting schemes. Then, we describe in Section 8.3 the common implementation and testing framework used in the following implementation and performance analysis Chapters. After this introductory Chapter, in Chapter 9 we will explain the implementation and performance analysis of the BBS and ACJT group signature schemes together with a comparison between them. Then, the implementations of our privacy-protecting solutions and their corresponding performance analysis will be provided: the μ*EasyPay* scheme in Chapter 10, the $\mathscr{MC}-2\mathscr{D}$ solution in Chapter 11 and the AFC system in Chapter 12.

8.2 Considerations on the Deployment of e-Commerce Solutions

In this Section we give some important considerations to take into account at the time to develop, deploy and evaluate e-commerce solutions, not just for schemes similar to those presented in Part II, but also to put into production e-commerce solutions in general.

In order to build the production scenario, we must select solutions to cover every part of the scenario: the server side, the customer application, the way to exchange messages among entities and how to encode these messages. To give a decision, we must analyze the available and current solutions for each part based on the following key features: *a*) performance; *b*) security; *c*) easy deployment; *d*) scalability; *e*) standardized mechanisms.

This analysis can be of particular interest for both researchers and developers because we are considering issues that must be kept in mind at the time of implement and evaluate e-commerce solutions.

8.2.1 Server Platform

To put into production an e-commerce solution we need to review which is the best solution to deploy the service and making it available for customers. In all schemes, there are entities running as servers (i.e. merchants, bank, issuer, group manager, payment manager and providers) so we must decide where they have to run. In terms of solely evaluate the scheme's functionalities and provide a proof-of-concept, the best option is to use a single local machine where servers and clients run. This way, the deployment becomes fully managed and the costs are cheaper (even free). This was the option that most of proposals choose to obtain performance measures, considering the same machine to run server and client developments [116]. However, running servers in a local machine does not take into account some external effects that must be considered in a real deployment scenario, such as communication lines, network bottlenecks, bounded computing capabilities, limited availability of resources, security levels, etc.

Next, we have to address the three most important issues to consider in order to decide which is the best solution to deploy the server entities: functionalities, scalability and security. First, the solution must provide means to execute multiple request-response exchanges among server entities and customers in order to successfully implement all the protocol flows. Moreover, it must allow the access to disk to store data and logs. Secondly, it must be scalable in such a way that it should provide a reliable service level and it should be able to supply more resources (e.g. compute, storage and network) as needed without it requiring to change the environment. Finally, security is one of the most important concerns in general, and for privacy-protecting schemes in particular, so the solution must provide means to fit some level of security to accomplish the security requirements (we will talk about it in §8.3.4).

There are two main options to run server entities remotely. On one hand, it can be considered the use of a physical, standalone and fully managed server and in the other hand, we have the option to rely on some sort of cloud solution (a cloud provider offers cloud services to cloud clients).

A remote physical server gives the ability to fully manage the deployment scenario, providing a customizable environment from the operating system to the service application in order to provide whatever functionality. It is claimed to be the best solution to fully manage the security since the system administrator can access and configure all the parameters as it was a local machine. However, the scalability could be a problem due to the fact these solutions require to add more servers as they need. This could be a time-consuming and costly task, because the development team must take care of infrastructure management and software development and deployment.

Cloud solutions are an interesting, affordable, distributed and real solution to allocate resources without worrying on the underlying physical infrastructure. Cloud clients use services from the cloud and they are typically charged by the cloud provider based on the amount of resources consumed. It means that most of the cloud solutions are scalable in a way clients can add more resources as they need. Based on both the functionalities offered by cloud solutions and the degree of control given to clients, solutions can be classified in three different types [117, 118]: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS).

Regardless of provided functionalities, privacy is yet another critical concern with regards to cloud computing due to the fact customers' data and business logic reside among servers, which are owned and maintained by the cloud provider. Therefore, there are potential risks that confidential data (e.g., financial data, health records) or personal information (e.g., personal profile) could be disclosed to public or business competitors [117, 118].

Software-as-a-Service (SaaS)

In this top-layer cloud solution, the provider offers software applications that can be used by clients as a service. Typically, clients access these services by using thin clients such as web browsers. There are several commercial examples of SaaS, such as Google Drive (formerly called Google Docs) [119] and Zoho [120], providing specialized applications like office suites, calendars, reminders and so on. The main security issue related to this cloud solution relies in the fact that data generated by clients is stored by the provider in a way they do not control at all. Because of this, most enterprises are uncomfortable with the SaaS model due to the lack of visibility about the way their data is stored and secured. Although this is the most basic cloud solution, since the client only uses an already deployed and configured service by the provider, the client has to depend and trust on the provider for proper security measures, such as storage and data sharing.

Platform-as-a-Service (PaaS)

In order to give clients more control over the cloud stack, the PaaS model emerges as a cloud solution giving a trade-off between flexibility and control. In fact, it is a cloud model allocated in the middle layer, where provider might give some control to clients to build and deploy custom applications and services on top of the platform. Provider gives clients a developing environment together with tools and libraries as a service in such a way that clients build and deploy their applications on top of these services. However, provider continues to have full control on networks, servers, storage and scalability features, making these resources transparent from

Table 8.1: Server models solutions: a comparative analysis.

	Own Server	**SaaS**	**PaaS**	**IaaS**
Deploy client apps	✓	-	✓	✓
Type of applications	any	provider-based	web-based	any
Resources scalability	own	provider	provider	provider
Performance tests	local-global	-	global	global
Security control	own	provider	provider-partially	client-full

✓YES , -NO

the client who can be focused instead to take care on the software development and its configuration. For example, GAE (Google App Engine) [121] and Windows Azure Platform [122] provide PaaS cloud solutions. Regarding to the security of the PaaS, the client has more control on the security of her application even thought any security below the application level (e.g. host and network intrusion prevention) is still in the scope of cloud providers. So, the cloud provider has to offer strong assurances that data remains inaccessible between applications, while clients have to trust on this assurance. As a result, the PaaS solution is more flexible and extensible as regards to both security and functionalities, at the expense the client should make more efforts to put into service an application or service.

Infrastructure-as-a-Service (IaaS)

The last cloud solution is the IaaS, in which provider offers resources in the bottom layer of the stack (e.g. computing and storage) and these resources are aggregated and managed either physically or virtually. It is the most flexible cloud solution because clients can configure their own environment but without worrying about the concrete details of the underlying physical infrastructure. Besides, clients have better control over the security, even they have to configure almost the whole stack (e.g. clients may have to install the operating system together with the required software and they are also in charge of maintaining both updated). Typical examples of IaaS are the Windows Azure Virtual Machines [122], the Google Compute Engine [123] and some of the Amazon Web Services (AWS), such as the Elastic Cloud Computing (EC2) [124] and the Amazon Simple Storage Service (S3) [125]. Regarding to the security of EC2, it includes vendor responsibility for security up to the hypervisor. It is the responsibility of the client to control the security related to the system, including the operating system, software, applications and data [126].

Summarizing, Table 8.1 compares the key feature of all the reviewed options related to the server platform.

Taking into account the requirements needed by an e-commerce service, the SaaS cloud solution should be discarded due to the fact that it does not provide the ability to install customized applications and services. Instead, both PaaS and IaaS solutions could be good choices since both allow deploying applications and services while cloud providers give a development and deployment platform (in case of PaaS) or a complete stack from operating system to the service (in case of IaaS). However, considering the security provided by both cloud solutions, IaaS could be claimed as the more convenient option due to the fact that the system administrator manages the whole system (including all the security) instead of relying to the provider some key tasks, as the way how to store data.

8.2.2 Client Application Platform

Similarly to the considerations about the server platform, some issues must be addressed to choose the way in which the client application must be developed. First, the solution must support not only applications based on web connections (with request-response pattern) but also connections where several messages can be exchanged within the same connection. Secondly, we need to use libraries (not just standard libraries) to provide more functionalities, such as data encoding or credentials management. Finally, minimum response time and high performance provisioning are main concerns regarding to mobile applications that require to do complex operations on the client side. Of course, there are other aspects that affect the response time, such as computational power of mobile devices and the efficiency provided by external cryptographic libraries.

There are many frameworks to develop applications for mobile devices. They can be classified in different ways, but for our purpose, we distinguish them into two main categories: platform dependent and platform independent. Table 8.2 gives a comparative analysis of these categories.

In order to develop applications based on the platform, we must use native frameworks provided by each platform vendor, such as Android (Google), iOS (Apple), Windows Phone 8 (Microsoft), Blackberry OS (Blackberry, formerly RIM), among others. This means that the code reusability is low, so this makes supporting multiple platform costly. However, this alternative has many benefits since these kind of frameworks have been designed to allow a rapid development (including simplicity), a direct access to platform APIs and better performance.

In the second category, we find many solutions under the concept of developing once, deploying everywhere [127, 128] (e.g. PhoneGap, Titanium, Corona, JQuery Mobile, etc.). Unlike native development, these solutions are oriented to develop a multi-platform mobile solution with a high degree of code reuse. In fact, we can differentiate two main types of platform-independent solutions: web-based frameworks and converters.

On one hand, in web-based developments it is the browser who determines the set of functions available to the web application (e.g. JQuery Mobile, Wink, etc.). This means that only technologies provided by all browsers on all devices can be used by web applications. Obviously, these frameworks are HTTP-based, so it restricts the type of application to be developed. Moreover, the developed web application is actually a front-end showing data received from the server side.

On the other hand, converter solutions are a set of programming languages and APIs provided by the framework (e.g. Titanium, Corona, etc.). At the end, there is a conversion step that creates platform specific applications (native code for each vendor platform). However, the use of these frameworks often does not include all functionalities available in native platforms. Moreover, it provides limited support when it is required the client side has to perform complex operations and it has to be linked with other external java libraries. In fact, cross-platform applications presents less performance than that obtained from native code applications [129].

Table 8.2: Frameworks for developing mobile apps: a comparative analysis.

	Platform Dependent	**Platform Independent**	
	Native	**Converters**	**Web-based**
Access to libraries	unlimited	limited (API)	limited (web browser)
Development cost	high	low	low
Client side operations	complex	limited	screen show based
Communication type	not limited	not limited	request-response
Performance	high	medium	low
Security	high	medium	low

8.2.3 Communication Technologies

Another consideration to take into account is the choice on the communication technology provided to the application. This decision should consider which is the best option in terms of coverage, bandwidth, cost, user adoption and availability of devices. As the application has to run on mobile devices, wireless network technologies should be reviewed. We have studied three different options: the Wi-Fi local-area network, the 3G mobile network and the NFC technology.

The Wi-Fi local-area network (IEEE 802.11 related standards) is a widely accepted technology to provide wireless connectivity in homes, offices, and hot-spots. It gives a throughput up to 600 Mbps [130] with a coverage area of about hundreds of meters. However, the throughput depends on many factors, such as the number of connected users, the distance to the access point, etc. Besides, this throughput refers to the internal network capacity, not to the link between the access point and Internet. Wi-Fi is becoming ubiquitous in urban areas, thanks to the growing popu-

larity of community networks such as FON [131] and the rise on the number of cities providing free Wi-Fi access. This makes Wi-Fi communications a realistic scenario in urban areas.

Regarding to 3G, it is based on wireless voice telephony that has evolved to also allow data transmission, providing connectivity with large coverage. Latest versions could provide rates up to 56 Mbps in the downlink and 22 Mbps in the uplink but they are still in the early stages of their deployment. As well as Wi-Fi, the data rate is shared among all users connected to a specific base station and it is also affected by other factors, such as movement, weather, distance to base station, etc. In fact, it is common to achieve data rates of about 2Mbps for stationary users, 384 Kbps for slow-moving users and 128 Kbps for users in fast-moving vehicles [132].

Finally, NFC is also another wireless technology, but NFC differs from Wi-Fi and 3G in some key points: the range of coverage, the amount of bandwidth available and the configuration at the beginning of the transmission [133]. NFC has a radio coverage of few centimeters, it means that NFC can be only used by proximity communications in which devices are physically close each other. Regarding to the offered bandwidth, NFC only allows transferring data at taxes of around few Kbps. Finally, as a difference, NFC does not need any kind of previous configuration, it is only required that devices has to be in the coverage range. Even it is not a new technology, it is not yet widely deployed as well as there are still few mobile devices with NFC support.

Summarizing the level of adoption of the reviewed technologies, both Wi-Fi and 3G are widespread wireless communications [134], they are present in all the current mobile devices and they are well-known technologies by users. Instead, even NFC is not a new technology, it is not yet widely deployed as well as there are still few mobile devices with NFC support.

8.2.4 Messages

Implementation of protocol messages requires to make a decision about how the structure and the syntax of the messages should be. Over the last years, the eXtensible Markup Language (XML) has been the preferred syntax for transferring business information across Internet, receiving a lot of attention. However, before the success of XML, most of security protocols were defined using the Abstract Syntax Notation One (ASN.1). Next, we briefly review some of the most important features of both XML and ASN.1.

Regarding to XML, it was developed in 1996 under the auspices of the World Wide Web Consortium (W3C) [135]. Implementation easiness, interoperability with HTML and human-legible was the main objectives of the XML design. But its main drawback is the achieved performance due to its uncompressed textual syntax [136].

Although message lengths can be reduced using data compression it increases the internal processing cost.

ASN.1 [137, 138] is a standard way to describe a message to be send or received from a network. ASN.1 is an ISO/ITU standard based on the OSI model and it was first defined in 1984. ASN.1 is divided into two parts. On one hand, the rules of syntax for describing the contents of a message in terms of structure and data type. On the other hand, how to encode and decode each data type. The primary performance advantage of ASN.1 over XML is that it was designed to efficiently encode values before transfer them.

Therefore, the syntax used to describe messages have implications on their size and on the time to encode/decode them. As a consequence, it affects the response time perceived on the client side, which is a critical issue because it can cause customer dissatisfaction and it could be a reason to stop using the e-commerce service.

8.3 Implementation Framework

In this Section we describe the common implementation framework we have selected to develop, deploy and afterward evaluate the performance of solutions presented in Part II.

8.3.1 Prototype Environment

Next, we describe the concrete technologies we have selected to develop and deploy solutions from the alternatives discussed in §8.2.

Regarding to the server platform, we have selected different options from those described in §8.2.1. In fact, we have selected own servers and an instance to the EC2 cloud solution.

Concerning to the platform to develop the client application, we have studied each considered mobile platform from §8.2.2 and we have decided to develop mainly on the native Android Platform, instead of other systems, due to the following key reasons:

- Android provides a well structured API to developers to build applications using the widespread and supported Java language.
- It offers a wide market of commercial handset devices to test developments.
- It is supported from a huge community of developers with a large number of resources, examples and so on.
- Android is open source and so anyone can contribute with enhancements.

- Android is supplied free of charge.
- A native framework performs better than cross-platform frameworks.

In addition, developing on the Android Platform covers a large percentage of mobile devices and a large market share. In fact, according to data provided by Google, Android is the leading mobile OS and the platform with a better growth rates, reaching recently up to 750 million of activations [139]. It means that more than a million of new Android devices are activated globally every day.

Besides, we have also developed the User Interface (UI) of the $\mathscr{MC}-2\mathscr{D}$ solution with PhoneGap as an example of platform-independent developing framework to know if it could be useful in our implementation scenario.

As described in §8.2.3, the communication network technology to use is the next choice. Although NFC seems a promising radio technology suitable to be applied to proximity transactions, we have discarded it not only because it is not yet a mainstream technology but also because of the lack of full support of the Android API. In fact, the API does not allow accessing all the features from NFC, such as full peer-to-peer NFC communication to implement complex message exchanges. Instead, the API only exposes the Android Beam feature. It consists on a very limited peer-to-peer NFC communication where an application can only send some data (text, contact card, image, photo, etc.) to another Android with NFC but without waiting or allowing to receive nothing different than a confirmation message reporting whether transmissions were finished properly or not.

As a result, we have selected the Wi-Fi and 3G wireless technologies. In all performance analysis, we have considered the utilization of Wi-Fi, but in the analysis of $\mathscr{MC}-2\mathscr{D}$ we also deal with the 3G network to provide a comparative analysis.

The way to encode protocol messages is an important choice as regards to customer's response time and the amount of data transferred to the network, as stated in §8.2.4. In general, we consider XML, but the implementation of $\mathscr{MC}-2\mathscr{D}$ also implements messages with ASN.1 to provide an extended analysis. This way, we will obtain empiric performance measures to study which is the best message syntax in that concrete scenario and to also know which is the effect on the whole performance of the $\mathscr{MC}-2\mathscr{D}$ solution.

Finally, due to the fact that Java is the language to develop with Android, we have also developed all server entities using the standard Java JDK 6.0 platform. Instead to consider using full-featured Java servers (such as Tomcat, Glassfish, JBoss, etc.) we have developed the server entities on top of the Java socket classes, with a recognized communication port bound to the network listening and waiting for incoming requests.

8.3.2 Secure Storage of Credentials

Both server entities and customer application must store credentials in a secure way. We have used the features provided by the `KeyStore` Java class, which exposes methods to store credentials in a secure repository protected by password. This way, only authorized users and applications can access and extract information from the key store after typing the right secret password.

However, the way Android exposes key stores to users and applications is different from the standard `KeyStore` class. Android provides the *system key store*, which is like a vault where only authorized users and applications can access stored private data. However, for versions previous to 4.0 (Ice Cream Sandwich), system key store was only available for VPN and Wi-Fi authentication procedures, applications were not able to access it. Instead, Android 4.0 brings several enhancements that allow applications to access the system key store to store personal credentials such as private keys and digital certificates, by means of a new API named `KeyChain` [140]. However, it is only available from 4.0 Android version. To preserve compatibility with older devices, we have used the standard method based on the `KeyStore` class. So, the application maintains his own key store saved on the smartphone's storage.

8.3.3 Tools, Libraries and Frameworks

As stated before, we have developed using the Java language. In addition to the Java standard libraries, a number of third-party libraries have been used along the code development. The five most important are the following:

Java Bouncy Castle

The Java Bouncy Castle library [141] is a very complete Java library that implements cryptographic algorithms and serves as a provider for the Java Cryptographic Extension (JCE) framework. The Bouncy Castle library presents a lightweight and comprehensible Application Programming Interface (API) suitable for its use in any Java environment. Moreover, it is continuously maintained and improved by a large group of developers. Bouncy Castle library has been useful to use it as a common framework, methodology and provider to use some cryptographic primitives.

Spongy Castle

Due to class name conflicts, Android does not allow applications to include the official release of Bouncy Castle. It happens mainly in older Android devices because of the differences between the internal version of Bouncy Castle and the last

release of the library. However, there is a project called Spongy Castle [142] distributing a renamed version of the library to work around this issue. It renames all the original packages from `org.bouncycastle.*` to the new one space on `org.spongycastle.*`. As a result, all of the features of the last version of Bouncy Castle can be loaded in any Android version. Bouncy Castle is used along the developed code as a common framework, methodology and provider to use some cryptographic primitives. It is also used to build ASN.1 encoding and decoding procedures where they are needed.

Although Java Bouncy Castle and Spongy Castle are powerful libraries, they do not offer support for pairing-based computations (used by the BBS group signature, see §2.3.3 and specially Chapter 9). Thus, we need specialized libraries to do so:

Pairing-Based Cryptography (PBC)

The PBC [143] is a C library which implements mathematical methods related to the elliptic curve generation, arithmetic and pairing computation. It is designed to be the backbone of other implementations of pairing-based cryptosystems and schemes. The PBC library is developed by Ben Lynn and where Shacham, one of the authors of the BBS group signature, has been contributed to the development. We have used PBC to create elliptic curves suitable for pairing computations. However, the PBC library cannot be linked easily with Java software developments, so a Java port of this library is required.

Java-Pairing Based Cryptography (jPBC)

The jPBC [144] is a Java library that performs the mathematical operations underlying pairing-based cryptosystems. It can work either as a full Java port of PBC or as a wrapper on the PBC library to delegate the pairing computation to PBC. This library has been used to implement all the operations related to the BBS scheme, using as inputs the elliptic curves previously generated by PBC. According both to the source code and the documentation, jPBC uses the Tate pairing [145, 146] to perform pairing computations. Thus, pairing computations rely on the Tate pairing.

Simple XML Framework for Java

The serialization of Java objects into a representation using XML can be a tedious work without using any specialized library. The first considered option was the Java Architecture for XML Binding (JAXB) [147] as it is the Java reference implementation for serializing objects to XML. However, it is not possible to be used on the Android platform due to the fact that Android does not allow calling to classes belonging the package `javax.xml.*`. As an alternative, we have found the Simple

XML Framework for Java (Simple for short) [148] which is a lightweight but powerful library that provides a high performance XML serialization and configuration framework for Java. Simple works without further configuration or problems on Android and it does not depend on any other underlying dependency, resulting in a very lightweight library.

Besides the above libraries, we have used the following three tools to make the developing environment more friendly and professional:

Apache Maven

Apache Maven [149] is a software project management and comprehension tool. Based on the concept of a Project Object Model (POM), Maven can manage the project's life cycle, providing developer many tools in a plugin-based fashion to manage dependencies, build source code, package applications, deploy to server or devices, test software, report problems and document advances during the development, allowing the developer to be focused on coding. With the aim to speed up the developing process and the dependency management we have installed and configured a central Maven repository where all the dependencies and generated applications are stored. In spite it has been a hard work, it improves so much the development life cycle.

Subversion

Subversion [150] is an open source version control system to control the software versions and to maintain the current and historical versions of files, such as the source code and documentation.

Eclipse

Eclipse [151] is the Integrated Development Environment (IDE) used to implement the proposed schemes.

8.3.4 Selection of a Common Security Strength

With the objective to compare and analyze the performance measures, we have to create a common security framework choosing a well-know and industrial accepted security strength. One of the most recognized organizations taking care of the data security is the National Institute of Standards and Technology (NIST) [152]. One of its main fields is to advise about security and cryptography, publishing periodical handbooks and reports with guidelines on standards and recommendations to be followed by the U.S.A. administration and other organizations in their relationships

to exchange and store data in a secure and interoperable way. So, guidelines published by NIST are an excellent framework to know the minimum allowed security strength. According to NIST, the security strength of an algorithm with a particular key length is measured in bits and it is a measure of the difficulty of discovering the key being used ([153], Section §1.2.1). The appropriate security strength to be used depends on the sensitivity of the data being protected, and needs to be determined by the owner of that data.

According to the last report released by the NIST [153], cryptographic algorithms relying their security on integer factorization (like RSA) can continue to use a module of 1024 bits to provide 80 bits of security strength through the end of 2013 for applications and services which do not have to store data in a secure way after that date. Besides, the same report advises that cryptographic algorithms based on elliptic curve and finite fields can also continue to use a subgroup of prime order of 160 bits, equal to 80 bits of security strength, also up to the end of 2013. However, NIST says that, since such keys are more and more likely to be broken as the end of 2013 approaches, the data owner must understand and accept the risk of continuing to use these keys to generate digital signatures. Opposed to that affirmations, other works like [154] claim that this level of security would be enough until the end of 2020, approximately.

In spite of these differences in the time frames, it is clear that it will depend on advances on computing and on the discovery of new methods to break the security of these algorithms. Indeed, at the time of writing this dissertation, the last and biggest known RSA modulus factorized is the RSA 768-bit size. The approximate time to factorize it was close to 200 years of computing on a single-core 2.2GHz AMD Opteron [155]. Authors also stated that unless something dramatic happens in factoring, a factorization of 1024-bit RSA modulus will not be possible at least until 2015. In any case, 80 bits of security (i.e. a 1024 bits RSA module or a subgroup of prime order of 160 bits) could be enough for many applications without strict secure storage requirements for a long time, such as those services built around the schemes presented in Part II of this dissertation.

8.3.5 Testing Devices

We have used some common devices to test every implementation and gather performance measures from all of the privacy-protecting solutions explained in Part II. Table 8.3 shows the most important features of each testing device, such as CPU clock speed, the amount of RAM memory and the operating system. In addition, we set a short name for every device to improve readability in further implementation Chapters.

Regarding to the testing devices where to install and run server entities, we have considered three different options: a conventional laptop, a virtualbox machine on a

server in our corporate network and an EC2 instance from AWS. All servers run the Linux OS with different CPU and RAM features. It is worthy to comment the considered cloud instance from AWS. In fact, EC2 offers their clients different instance types depending on the required resources to meet their concrete requirements. Depending on the amount of required computing time, resources (such as memory, storage, replication, availability, cloud geographical localization, etc.) and the number of running instances, EC2 clients pay different fares. Among available options, we have chosen the free tier micro instance (μ-instance) [156], because it provides, for free, a suitable level of resources to obtain consistent performance measures.

Besides servers, we have three commercial Android smartphones available to test implementations. Ordered from the least featured to the most powerful one, we have: the HTC Wildfire, as a low class smartphone; the HTC Desire, as a medium class Android device; and the Google Galaxy Nexus, which is one of the state-of-the-art Android smartphones in the market equipped with the last version of Android. Evaluating our schemes over different devices with both heterogeneous computing capabilities and different Android versions can provide valuable information about how the schemes perform in different types of smartphones due to the fact that not all customers can access to the best (and so expensive) mobile devices. Moreover, it is also useful from the point of view of predict the performance evolution of these scheme as new smartphones are released to the market.

Table 8.3: Considered testing devices.

Device	**Name**	**CPU**	**Instruction set**	**RAM**	**OS**
Laptop	Laptop	Intel Core2Duo (dual-core 2.4GHz)	x86_64 (64 bits)	4GiB	Debian Linux (Last stable)
Virtualbox machine	VM	1 virtual core (single-core 2.8GHz)	x86_64 (64 bits)	1GiB	Debian Linux (Last stable)
EC2 μ-instance	CS	2 EC2 CU[(1)] ($\approx$ 1.0-1.2GHz)	x86_64 (64 bits)	633MiB	AWS Linux (customized)
HTC Wildfire	Wildfire	Qualcomm MSM7225 (single-core 528MHz)	ARMv6 (32 bits)	512MiB	Android 2.3.5 (API level 10)
HTC Desire	Desire	Qualcomm Snapdragon (single-core 1GHz)	ARMv7 (32 bits)	512MiB	Android 2.2 (API level 8)
Google Galaxy Nexus	Nexus	Texas Instruments OMAP (dual-core 1.2GHz)	ARMv7 (32 bits)	1GiB	Android 4.2.2 (API level 17)

[(1)] One EC2 CU (Compute Unit) provides the equivalent CPU capacity of a 1.0-1.2GHz 2007 Xeon processor [156].

Armed with this list of devices and their technical features, we will describe the concrete testing scenario for each implementation and its corresponding performance evaluation in their respective analysis due to the fact that each proposal has been tested with slightly different scenario configurations. For example, some ex-

periments have been conducted taking into account the side effects caused by the use of network connections (such as in the $\mathscr{MC}-2\mathscr{D}$ performance evaluation) while other ones have been performed locally and only taking into account computation aspects (such as in the group signature performance analysis and comparison).

8.4 Conclusions

In this Chapter we have reviewed some important considerations to take into account when an e-commerce solution has to be implemented, deployed and evaluated. These considerations should contemplate which are the best options for the server and client platforms, the communication technologies and the way to encode messages. Among these general considerations, we have selected some technologies that fit better for our evaluation framework to implement and extract performance measures in the following Chapters.

CHAPTER 9

GROUP SIGNATURE: COMPARING BBS AND ACJT SCHEMES

Group signatures are usually managed as black-boxes by other secure proposals to provide authenticity and anonymity to signers. However, there is a lack of implementations of group signatures to test whether they are really efficient, even in mobile devices. In this Chapter we present a successful implementation of two group signatures with different mathematical backgrounds: the pairing-based BBS group signature and the RSA-based ACJT scheme. Using both implementations, we analyze and compare their performance with mobile devices together with a study to know which are the pairing computations with larger cost on mobile devices.

9.1 Introduction

In the group signature literature, this cryptographic primitive is used very often as an underlying black box by several theoretical proposals of secure applications and services, such as remote attestation [58–60], access to subscription services [61] or e-cash systems [64–66]. Sometimes the computational overload of group signatures has been considered so heavy to be used in portable devices in a real environment so as to ensure users' privacy using those applications. However, there is a lack of implementations of group signature proposals to test their applied efficiency (specially on mobile devices) rather than purely show their mathematical complexity analysis.

In this Chapter we present, to the best of our knowledge, the first successful software implementation of two different group signature schemes by means of the Java language: the pairing-based BBS group signature due to Boneh et al. (refer to §2.3.3) and the state-of-the-art non-pairing RSA-based ACJT group signature by Ateniese et al. (refer to §2.3.3). We have tested both implementations and we have analyzed the performance measures on a conventional laptop and two different Android smartphones, comparing the gathered results in order to provide some interesting insights about their use. To obtain results under the same complexity conditions, we have carefully selected the input parameters in order to use a common security strength of 80-bit, as explained before in §8.3.4.

We do not only provide a simple comparison between both schemes, we also supply a valuable comparative analysis using different types of pairings to instantiate the BBS pairing-based group signature with the same security strength. As a result, this Chapter presents some insights about which pairings are better to use by the BBS scheme, depending on the scenario where they have to be applied. These insights could be even extrapolated to other pairing-based schemes and protocols to obtain an idea about which pairings and operations related to them are more costly in each device. Therefore, the work in this Chapter is intended to clarify the real computing load due to the use of group signatures in secure applications and services, specially on those used on mobile devices, contributing to increase their utilization as a nice cryptographic primitive to give privacy to users. Besides, we have created a complete Java library to use group signature operations in further implementations (in fact, it will be used afterward to implement the multicoupon and the AFC privacy-protecting schemes presented in Chapters 4 and 5, resp.).

This Chapter is organized as follows. Section 9.2 describes how we have selected input parameters to both group signature schemes and how to implement a method based on X.509 to distribute group public keys for the case of the BBS scheme. A study of performance and comparison between both group signatures is provided in Section 9.3. Finally, in Section 9.4 we conclude the Chapter.

9.2 Implementation

In this Section we are going to describe the process we have followed to implement and test both group signature schemes. We describe how we have selected input parameters for both BBS and ACJT schemes in order to obtain signatures with 80-bit of security strength. This has been a tedious process, mainly to chose pairing parameters to be the input to the BBS scheme. Besides, we also describe how to fit the BBS group public key into the X.509 public key infrastructure to distribute keys and certificates.

9.2.1 BBS: Selection of Pairing Parameters

The first step to obtain group signatures is to provide the algorithm with the proper input parameters. Because of BBS group signature scheme is based on pairings, we have to provide pairings and elliptic curves.

Brief Overview on Elliptic Curves

At this point, it may be useful to bring a brief overview on elliptic curves and how they are related to pairings [145, 146]. This overview is not intended to be extensive as we are not experts in this field (neither in the underlying mathematics). Therefore, the purpose of this Section is to provide clarification on some terms, because the understanding of the underlying mathematics is not the objective.

Let q be a prime number, and $\mathbb{F}_q$ a field of integers modulo q. An elliptic curve E over the finite field $\mathbb{F}_q$, denoted as $E(\mathbb{F}_q)$, has its arithmetic in terms of the underlying finite field and is defined by the following equation:

$$y^2 = x^3 + ax + b$$

where a, b are defined in $\mathbb{F}_q$ and satisfy $4a^3 + 27b^2 \not\equiv 0 \mod q$. The set of solutions (x, y) with $x, y \in \mathbb{F}_q$ satisfies the above equation. Additionally, the point-at-infinity, denoted as $\mathcal{O}$ is also on the curve.

Let Q be a point in $E(\mathbb{F}_q)$ with prime order r; then a cyclic subgroup of $E(\mathbb{F}_q)$ generated by Q is denoted as:

$$\langle Q \rangle = \{\mathcal{O}, Q, 2Q, 3Q, \ ..., \ (r-1)\, Q\}$$

where r is the number of elements in the subgroup.

The discriminant $\triangle(E)$ of an elliptic curve E is defined as

$$\triangle(E) = -16\left(4a^3 + 27b^2\right)$$

where a, b are the parameters of the short *Weierstrass formula of E*.

The number of points on $E(\mathbb{F}_q)$ relies in the so called *Hasse interval*:

$$q + 1 - 2\sqrt{q} \leq \#E(\mathbb{F}_q) \leq q + 1 + 2\sqrt{q}$$

The number $t = q + 1 - \#E(\mathbb{F}_q)$ is called the *trace of Frobenius* of E over $\mathbb{F}_q$.

The *embedding degree* of a curve is the smallest integer k where $r \mid (q^k - 1)$ and sometimes it is also referred as the security multiplier of the curve.

Finally, an elliptic curve can be *supersingular* or *ordinary* depending on the following:

- if q divides t (the trace of Frobenius), the elliptic curve is *supersingular*,
- otherwise, the curve is *ordinary*.

Pairing-friendly Elliptic Curves

Pairings have important implications in the Elliptic Curve Cryptography (ECC) as they are the base of the pairing-based cryptography. As a reference, the pairing-based cryptography is the use of a pairing functions to map elements of two input groups $(\mathcal{G}_1, \mathcal{G}_2)$ to a third group $(\mathcal{G}_T)$ to construct cryptographic systems. If $\mathcal{G}_1$ and $\mathcal{G}_2$ are the same group, the pairing is called *symmetric*. Finding groups with a bilinear map, such as the *Weil* pairing and the *Tate* pairing is a challenging task and an important research field in pairing-based cryptography [145].

There are several pairing-friendly elliptic curves with different features to be used to perform pairing operations [145, 157]. Among them, there are the following six elliptic curves that we have selected to analyze the BBS group signature scheme: A, A1, D, E, F and G. The names of these elliptic curves differs from authors to other ones. We have followed the nomenclature from [157]. We provide a brief description about each selected pairing type [157, Ch. 4, pp 62].

Type A. Type A pairing uses supersingular curves built on the curve $y^2 = x^3 + ax$ for any a and for some prime $q = 3 \mod 4$. It is common that $a = \pm 1$. Moreover, it conforms a symmetric pairing. Its order p is some prime factor of $q + 1$. For efficiency, the order p is selected to be a Solinas prime, i.e., p has the form $2^a \pm 2^b \pm 1$ for some integer $0 < b < a$.

Type A1. This pairing type uses the same equation as type A pairing. In this case, a curve of composite order can be used, e.g. $N = pq$ where p and q are large primes so that N is hard to factorize.

The following pairing types are based on the *Complex Multiplication* or *CM* method [158]. It is a method to generate ordinary curves with larger values for the embedding degree (k). The goal is to find an elliptic curve E over $\mathbb{F}_q$ with complex multiplication by D such that there is a large prime p dividing $\#E(\mathbb{F}_q) = q + 1 - t$ and such that p divides $(q^k - 1)$ for a suitable value of k. The method is based on the *CM equation*:

$$DV^2 = 4q - t^2$$

where D, V, q and t are integers satisfying that equation.

Type D. These are ordinary curves [159, 160] with embedding degree $k = 3, 4, 6$, even though the most useful case arises when $k = 6$. The order of these curves is a prime or a prime multiplied by a small constant. It is defined over some field $\mathbb{F}_q$ and has order $h \cdot p$, where p is a prime and h is a small constant. It can be built using the *CM* method and also through the use of the MNT method [159]. The discriminant D must be $D = 0, 3 \mod 4$ and positive. These curves are good for cryptosystems when group elements must be short.

Type E. It is an ordinary curve with embedding degree $k = 1$ and this type of curves can be easily found applying the *CM* method. They only require arithmetic modulo a prime since they can be implemented in a field of prime order.

Type F. It is another ordinary curve with embedding degree $k = 12$ found using a method due Barreto and Naehrig [161]. These curves can be mainly used when the priority is to minimize the bandwidth and the required storage since its high embedding degree allows shorter inputs. If finite field discrete log algorithms improve further, type D pairings will have to use larger fields, but type F can still remain short and secure.

Type G. It is the last considered type of ordinary curve, due to the work presented in [162], with embedding degree $k = 10$, where p and q have similar size. It also uses the *CM* method choosing D such that $D = 43, 67 \mod 120$. Due to its high embedding degree, as type F, type G curves provide shorter inputs and they guard against future improvements of finite field discrete log algorithms.

Table 9.1: Embedding degree and theoretical lower bound size of q for each considered type of pairing (equivalent to 80 bits of security strength).

Type	**Embedding Degree** k	**Minimum Input Input Size** $\lceil log_2 q \rceil$ **(bits)**	**Minimum Output Ouput Size** $\lceil log_2 q^k \rceil$ **(bits)**
A	2	512	1024
A1	2	512	1024
D	6	171	1026
E	1	1024	1024
F	12	160	1920
G	10	160	1600

Summarizing, Table 9.1 shows a comparison among the previously described pairing types [157, Ch. 4, pp 62]. Pairings should consider a trade-off between security and complexity. On one hand, for security reasons, $\mathbb{F}_q$ must be large enough so that $E(\mathbb{F}_q)$ can resist generic discrete log attacks, while $\mathbb{F}_{q^k}$ must be large enough to resist finite field discrete log attacks. On the other hand, regarding to the computation time and the storage requirements, $\mathbb{F}_q$ and $\mathbb{F}_{q^k}$ should be as small as possible. Table 9.1 shows the lower bounds for q, i.e. the minimum length to represent the order of the input $\mathbb{F}_q$ $\left(\lceil log_2 q \rceil\right)$ for each type of curve. Thus, it is commonly accepted (and already pointed out in §8.3.4) that q needs to be at least 160-bit long in the input and more than 1024-bit long for q^k in the output. Moreover, the order of $E(\mathbb{F}_q)$ is currently acceptable to be at least 160-bit, as pointed out in §8.3.4.

Parameter Deployment and Curve Selection

In order to compare the BBS scheme performance using these six types of pairings, we need to define another common security level. Thus, taking as a starting point the theoretical values presented in the Table 9.1 and explained above, we have tried to compute six elliptic curves for each considered pairing type. Curves have been generated by the PBC library using as inputs the order of the subgroup (for A and F pairing types), the number of bits for q (for A and E pairing types) and finally the discriminant D (for D and G pairing types). Then, this curves have been sent to an application made by using the jPBC library to collect the order and the length of a random element (in bits) picked up from each group for each selected pairing type. Table 9.2 shows the results and it also exposes the length achieved for q and q^k.

Table 9.2: Groups order and length of a random element within each group collected using the jPBC library on input of curves generated by PBC.

Type	Order (bits)				Element length (bits)				Minimum Size (bits)	
	$\mathcal{G}_1$	$\mathcal{G}_2$	$\mathbb{Z}_p$	$\mathcal{G}_T$	$\mathcal{G}_1$	$\mathcal{G}_2$	$\mathbb{Z}_p$	$\mathcal{G}_T$	In the Input ($\lceil log_2 q \rceil$)	In the Output ($\lceil log_2 q^k \rceil$)
A	161	161	161	161	1040	1040	168	1040	513	1025
A1	512	512	512	512	1040	1040	512	1040	520	1040
D	161	161	161	161	352	1056	168	1056	175	1048
E	161	161	161	161	2064	2064	168	1032	1025	1025
F	162	162	162	162	336	672	168	2016	162	1935
G	279	279	279	279	608	3040	280	3040	301	3006

The order of each group is around 160 bits, except for type A1 and type G pairings. On one hand, type A1 pairing uses a curve of composite order but PBC library does not allow to set the group order. Without the ability to change values from the composite modulus, the A1 pairing cannot be configured properly to accomplish the required input and output lenghts. However, this type of pairing remains in the com-

parison since it accomplishes the requirements about q and q^k unless it is commonly known that considering composite orders results in significant performance degradations. On the other hand, we could not find any curve for type G pairing with a similar level of security as the other ones. We have tested more than 16.000 discriminants up to $D = 1.000.000$ that accomplish the condition $D = 43,67 \mod 120$. We have found only one curve with q longer than 160 bits. However, this curve presents an order of 279 bits and q with 301 bits, so it cannot be compared with the other curves.

Note that the bit-length of each element randomly picked out from the corresponding group is always multiple of 8 and higher than theoretical values from Table 9.1. It is due to how jPBC works, because the library seems to give results in bytes.

9.2.2 ACJT: Selection of Parameters

As explained in §2.3.3, ACJT group signature requires to choose several security parameters and lengths satisfying some conditions. The first parameter that should be defined is l_p, which determines the composite modulus n. According to §8.3.4, if we want to obtain a security strength of 80 bits, we must compute a RSA modulus with 1024 bits. Because of this, we should fill $l_p = 512$ bits. Moreover, k is selected as to be 160-bit long due to the fact that we want to use the SHA-1 hash algorithm in the implementation, which generates 160 bits outputs. Note that each length specified by $\lambda_1, \lambda_2, \gamma_1$ and γ_2 have been selected ceiling the result to the near byte rather than strictly add one bit according to the *greater than* condition. So, in some parameters, the considered value (third column from Table 9.3) could be greater than the theoretical minimum required by the ACJT specification (second column from Table 9.3).

Table 9.3: ACJT parameter selection.

Parameter	Theoretical minimum value	Considered value	
ϵ	2	2	
l_p	512	512	(bits)
k	160	160	(bits)
λ_1	4419	4440	(bits)
λ_2	2049	2056	(bits)
γ_1	9167	9224	(bits)
γ_2	4422	4448	(bits)
n	1024	1024	(bits)

During the software implementation, we have found some challenging problems such as the search of a value for the second part of the membership certificate (e_i), defined as a random prime number within the range $\Lambda\left(e_i \xleftarrow{R} \Lambda\right)$ (see 2.3.3). Accord-

ing to the lengths defined before, e_i should be a prime number of 9224 bits. We have tried to find a prime number with this length using the OpenSSL library, but unfortunately without results after a very long time computation. As an alternative, we have decided to search for an already computed prime number whose length would be as near as possible to the aforementioned requirements. So, we have selected a primer number of 9212 bits [163], which is very close to the target length of 9224 bits. It is a clear example of a kind of challenge that appears when theoretical algorithms should be implemented and brought to the practice. In this cases, developers should give engineering solutions as close as possible to the theory, as it has been done here.

9.2.3 Group Key Distribution

A group signature scheme requires a group manager can generate and distribute group public keys (along with the corresponding set of user private keys), in a secure way. Often, digital certificates are used for disseminating certified data and it is a very extended instrument to certify data such as public keys, user privileges, etc.

This is a similar concept as in X.509 public key infrastructure [164, 165], where an entity called CA is in charge of managing Public Key Certificates (PKCs). However, the main difference regarding to X.509 is that in a group signature model, a set of users have the same group public key while they have different associated private keys.

We address the three most important requirements that must accomplish the solution for group public keys management. First, the certificate should not contain any information to identify or link the signer. Secondly, data included in the certificate must hold the group identifier and the corresponding group public key. Lastly, the format of the certificate specification must be supported by the most extended user applications, e.g. HTTPS.

```
TBSCertificate  ::=  SEQUENCE  {
     -------
     subject              Name,
     -------
     subjectPublicKeyInfo SubjectPublicKeyInfo,
     extensions      [3]  EXPLICIT Extensions OPTIONAL
     }
```

Figure 9.1: X.509 Certificate Profile.

Indeed, X.509 PKC seems a good candidate to distribute group public keys in a standard and secure way. However, they were originally designed for public key algorithms in which a single public key corresponds with a unique private key. Moreover, the public key is bound to the identity of a unique entity. It is completely

different from group signatures idea, so we have to provide a solution to fit group signatures in the X.509 public key infrastructure. If we take a look at the X.509 v3 certificate format reproduced in Figure 9.1, it can be observed that the definition fields does not restrict its use for other data structures.

```
SubjectPublicKeyInfo  ::=  SEQUENCE  {
     algorithm            AlgorithmIdentifier,
     subjectPublicKey     BIT STRING  }
```

Figure 9.2: Public Key Info.

The `SubjectPublicKeyInfo` (Figure 9.2) field is used to carry out the public key and identify the algorithm which makes use of it (e.g., RSA, DSA, or Diffie-Hellman) [165]. But this field can contain any ASN.1 type coded as an string of bits. For the particular case of the BBS group signature, Figure 9.3 represents the ASN.1 syntax to define the BBS group public key (see also §2.3.3).

```
SGSPublicKey  ::=  SEQUENCE  {
     g1            OCTET STRING,
     g2            OCTET STRING,
     h             OCTET STRING,
     u             OCTET STRING,
     v             OCTET STRING,
     omega         OCTET STRING
     }
```

Figure 9.3: ASN.1 Group Public Key.

The most important point about group public keys is that it can be represented as a complex and structured ASN.1 type composed by ASN.1 simple types. Thus, the output of the `SGSPublicKey` structure can be converted into a bit string accepted as a `subjectPublicKey` field value in the X.509 standard. The `Subject` field defines who is the owner of the public key (who holds the public key), so in our case, it must be the group identifier, not a specific user identity. This way, the `Subject` field does not either identify or link the signer but the group.

To correctly process the certificate, it must include information about the type of public key that it conveys. For this purpose, the field `algorithm` from `SubjectPublicKeyInfo` can be used. However, today there is not a standardized value to identify the group public key, but X.509 allows to define new identifiers as needed. To maintain the interoperability, we propose to use the extensions field to specify that this certificate is in fact a group key certificate. This new extension must be defined as critical, i.e. this extension must be understood by any entity in charge of validating the certificate.

Moreover, using PKCs X.509 allows taking advantage of already available tools and the infrastructure to manage X.509 certificates, from the use of secure containers to store certificates and public keys (e.g. PKCS12, PKCS8, etc.) to the PKC management messages (e.g. CMP) [166, 167].

Therefore, using the X.509 PKCs we accomplish the three requirements for using group public keys: unlinkability to the private key holder, feasibility of group public keys representation and the use of standardized management mechanisms.

9.3 Performance Discussion

Now we are going to analyze the results retrieved from the implementation of both schemes. Remember that the input security parameters have been carefully selected to compare both schemes with a security strength equivalent to 80 bits. This Section is divided in four parts: in §9.3.1 the considered test scenario is described; in §9.3.2 we offer an analysis about the computation of the group manager to generate group key pairs; in §9.3.3 we analyze the computation required by a group member to sign and verify a message; finally, in §9.3.4 we provide an analysis about the storage requirements of both group manager and group members.

9.3.1 Test Scenario

Since we only consider the performance related to the computing overhead of those procedures belonging group signature schemes, the test scenario does not take into account communications. We have used Desire and Wildfire Android smartphones and the Laptop, without any kind of network connection among them. All features of these testing devices have been summarized in Table 8.3 in §8.3.5. In order to obtain average measures, we have repeated each test up to 100 times by each device.

9.3.2 Group Manager Computation Analysis

Figure 9.4 shows the measured time to compute on the Laptop a number of key pairs during the $KeyGen^G$ algorithm considering only the BBS scheme. Note that, since ACJT defines an interactive $Join^G$ algorithm between the group manager and the candidate to become a group member, it does not compute all the user secret signing keys during the $KeyGen^G$ phase. Instead, ACJT creates a single secret signing key every time a user requests the $Join^G$. Analyzing Figure 9.4, type D and type F pairings are the fastest pairings to complete the process. Instead, type E and A1 pairings are the slowest because the former only uses modular arithmetic and the latter uses elements that are larger than those provided by the other curves. It is also interesting to observe that as the size of the group increases (i.e. the number of user private keys to be built grows), the computational cost needed for type D

and type F pairings to finish the algorithm is increasing slowly. Instead, for the other types of pairings, the cost grows quickly. So, it is clear that from the point of view of computational cost in the server side (group manager entity), type D and type F pairings are the best choice in terms of scalability and performance, even this algorithm can be performed offline.

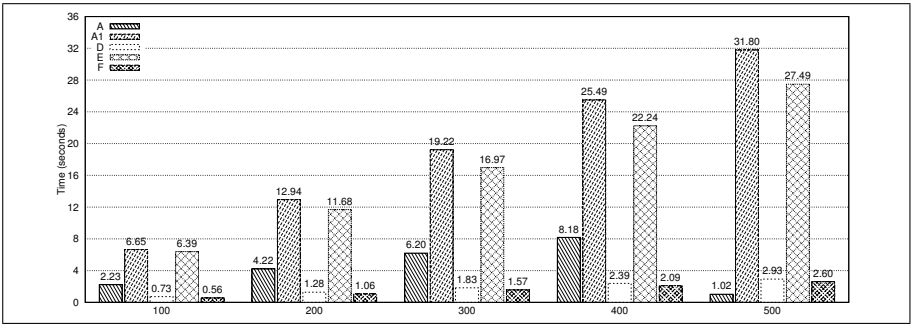


Figure 9.4: Performance results related to the BBS group manager who performs the $KeyGen^G$ algorithm on the laptop.

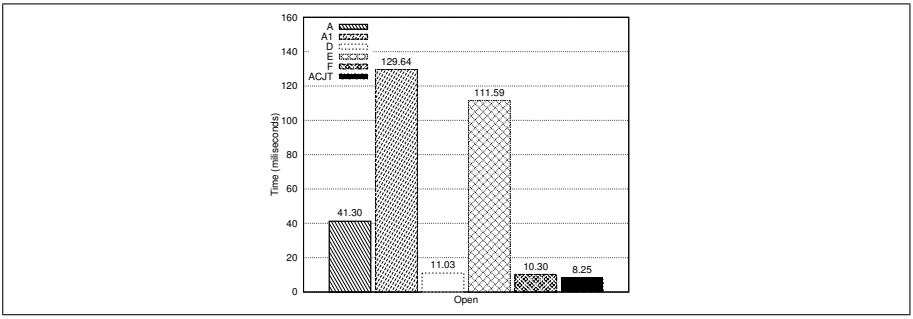
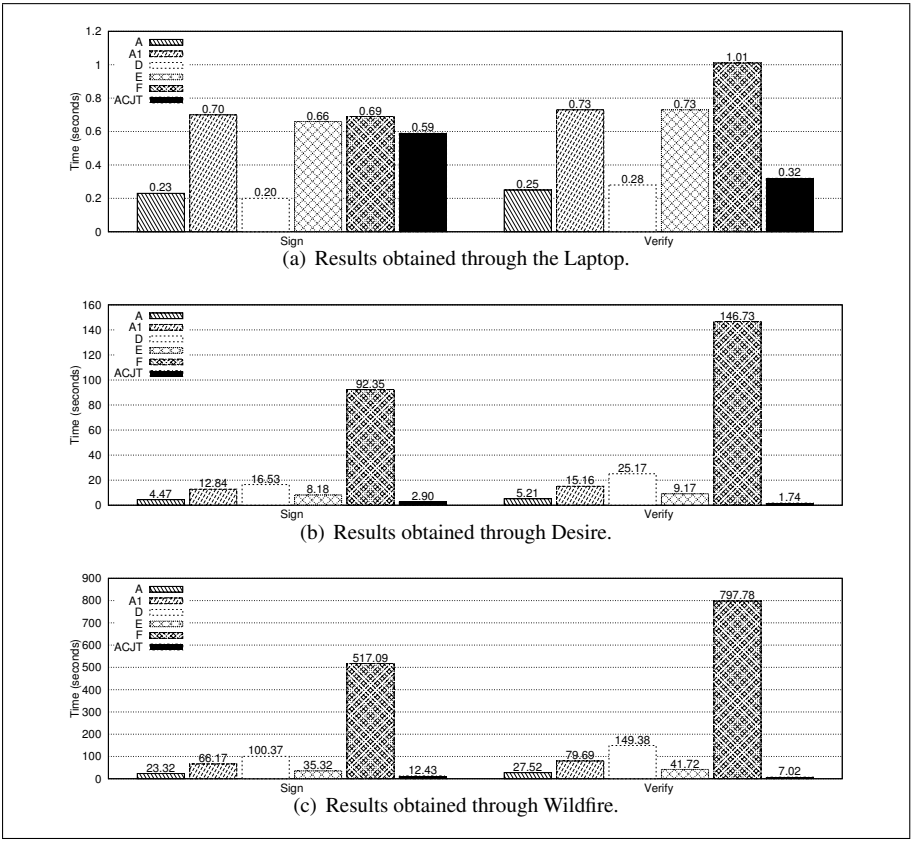


Figure 9.5: Time to trace a signature to the corresponding signer.

Concerning the $Open^G$ algorithm, Figure 9.5 reflects the time needed to trace a signature to the corresponding signer considering both group signature schemes. Note that this time does not include the signature verification process in any case. Like in signing and verification processes, type D is the best pairing obtaining roughly the same performance than type F pairing. However, ACJT seems to be a little bit more efficient than BBS, but only around 2 ms faster on average.



(a) Results obtained through the Laptop.

(b) Results obtained through Desire.

(c) Results obtained through Wildfire.

Figure 9.6: Performance results related to a user who performs the $Sign^G$ and the $Verify^G$ algorithms from BBS (using types A, A1, D, E and F pairings) and ACJT schemes.

9.3.3 Group Member Computation Analysis

Regarding to the $Sign^G$ and the $Verify^G$ algorithms performed by a group member, we have analyzed the performance of both considered schemes on the Laptop and on two different smartphones, to see how the behavior of the implementation in different hardware and operating systems is. Results are depicted in Figure 9.6.

Figure 9.6(a) exposes the results obtained using the testing Laptop. Regarding to the BBS scheme, type A and type D pairings are the best ones, beating the other types of pairings. Type D is slightly better for signing while type A is narrowly faster for verification. The other types of pairings have similar performance, excepting the

type F for the verification process, which is slightly slower. Curiously, if we analyze results obtained by the Desire (Figure 9.6(b)) and the Wildfire (Figure 9.6(c)), we can see that results follow a different trend than those produced by tests performed on the Laptop. So much so that, type A pairing remains the best pairing type both for signing and verifying. Instead, type D pairing is affected by a fall in performance becoming one of the worst pairings, just ahead of type F pairing, which is proven not usable at all in a current smartphone device.

If we compare the BBS scheme with the ACJT scheme, we see two types of results depending whether the schemes are tested on the Laptop or on the smartphones. On one hand, BBS performance on the Laptop is very similar to the performance obtained using the ACJT group signature. Actually, types A and D pairings seem to be faster than ACJT for signing and verifying. On the other hand, when tests are performed on the smartphones, it is clear that the BBS pairing-based group signature tends to be slower than the ACJT. This way, carefully analyzing results presented in Figure 9.6, we can prove that there are some operations that could be more difficult to compute by a smartphone, probably due to its architecture, i.e. CPU, memory access, operating system, etc. It implies that the performance of the BBS scheme falls down whether it is used by a smartphone.

We want to know whether it is possible to improve the performance, specially on the smartphones, applying some kind of precomputation. Table 9.4 shows the number of operations that should be performed by the signing and the verifying algorithms, and how many of them can be precomputed by the signer and the verifier, respectively. In fact, in the case of the BBS scheme, during the $Sign^G$ algorithm, most of the operations can be precomputed, allowing the signer to compute only 5

Table 9.4: Number of precomputable operations along the signature and verification algorithms.

		Signature			Verification		
Scheme	**Operation**	**Number**	**Precomp.**	**%**	**Number**	**Precomp.**	**%**
BBS	Random ($\mathbb{Z}_p$)	4	4	100	-	-	-
	Multiplication ($\mathbb{Z}_p$)	7	2	28.6	-	-	-
	Multiplication ($\mathcal{G}_1$)	3	3	100	4	0	0
	Exponentiation ($\mathcal{G}_1$)	9	9	100	8	0	0
	Inverse ($\mathcal{G}_1$)	-	-	-	4	0	0
	Multiplication ($\mathcal{G}_T$)	2	2	100	4	0	0
	Exponentiation ($\mathcal{G}_T$)	3	3	100	4	0	0
	Inverse ($\mathcal{G}_T$)	-	-	-	1	1	100
	Pairing	3	3	100	5	3	60
ACJT	Random	5	5	100	-	-	-
	Modular multiplication	6	6	100	10	0	0
	Modular exponentiation	12	12	100	13	4	30.8
	Modular inverse	2	2	100	2	0	0
	Modular additions	-	-	-	4	0	0

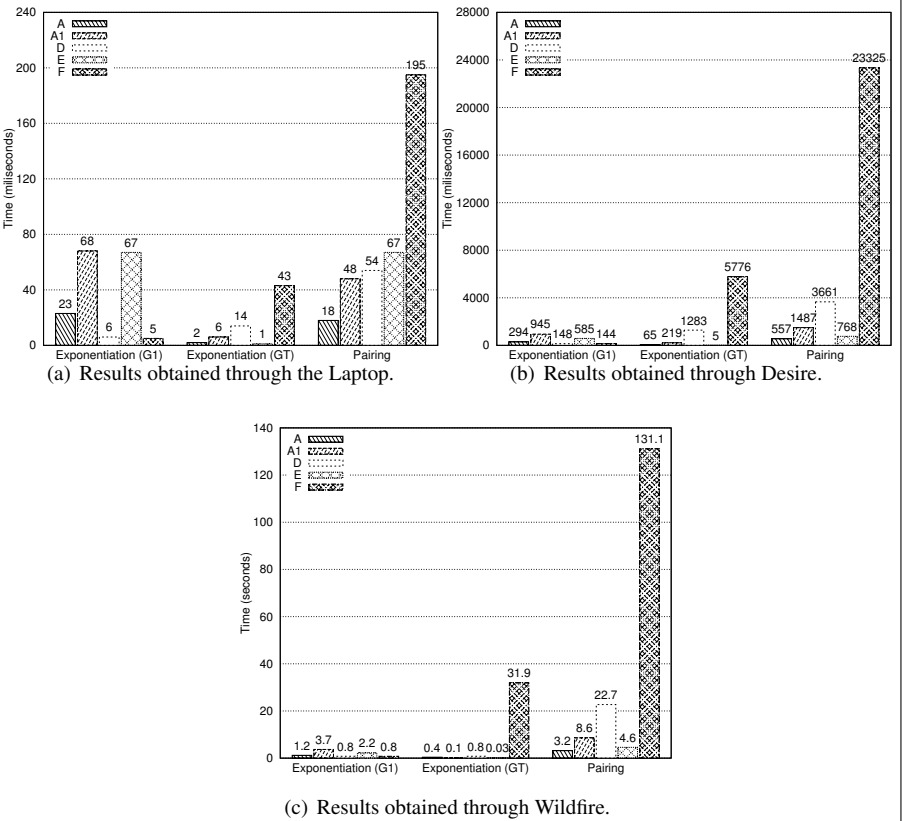
of 7 multiplications in $\mathbb{Z}_p$ and one hash computation. Regarding to ACJT, the signer has not to compute any modular operation until he decides the message to group sign if precomputation is enabled. Instead, during the *Verify*^{G} algorithm using the BBS scheme, only 3 of 5 pairing evaluations and one inverse in $\mathcal{G}_T$ can be precomputed, and only 4 of 13 modular exponentiations in the case of the ACJT scheme. So the improvement seems to be greater during the signature generation rather than in the verification.

Table 9.5: Percentage of improvement if precomputation is enabled during the signature generation and verification on both Laptop and Desire.

Type	**Computing time on Laptop**			
	Signing		Verification	
	(miliseconds)	(% improved)	(miliseconds)	(% improved)
A	0.84	99.64	198.27	19.17
A1	0.62	99.91	609.43	16.52
D	0.29	99.85	166.43	43.50
E	1.31	99.80	547.74	24.52
F	0.42	99.94	551.55	45.10
ACJT	1.3	99.78	320	3.03

Type	**Computing time on Desire**			
	Signing		Verification	
	(seconds)	(% improved)	(seconds)	(% improved)
A	0.14	96.96	3.76	58.08
A1	0.13	98.99	10.97	58.02
D	0.05	99.70	14.14	64.03
E	0.39	95.45	6.85	57.24
F	0.15	99.84	73.92	66.50
ACJT	0.01	99.63	1.40	19.54

Table 9.5 resumes the improvement that reports precomputation if it is enabled using both the Laptop and the Desire (the same pattern applies to the Wildfire). The percentage of time that can be precomputed during *Sign*^{G} algorithm is almost the same in both devices, reaching up to the 99%. Surprisingly, precomputable time during the *Verify*^{G} algorithm is higher on the smartphone than on the Laptop (in general, it is around of 20% of improvement on the Laptop and up to 60% on the smartphone). So, the improvement on the performance due to the precomputation during the *Verify*^{G} algorithm on the smarphones could be better than the benefit obtained on the Laptop. If we take into account that only 3 of 5 pairings and one inverse in BBS is precomputed before to receive and verify the group signature, we can point out that one of these operations tends to be more difficult to execute by a mobile device. In fact, Figure 9.7 reflects a selection of costlier operations used by the BBS scheme in each testing device: the exponentiation in $\mathcal{G}_1$, the exponentiation in $\mathcal{G}_T$ and the pairing evaluation. Then, pairing evaluation (especially for type F



(a) Results obtained through the Laptop.

(b) Results obtained through Desire.

(c) Results obtained through Wildfire.

Figure 9.7: Elapsed time to perform costlier pairing operations on testing devices.

pairing) tends to be the most expensive operation on smartphones. Therefore, this is the reason because type F pairing is the worst pairing if it is tested on a smartphone. Since all pairing operations during the BBS signature generation can be precomputed and due to the fact that they are the most costly operations on smartphones, the percentage of improvement stands about of 99%.

9.3.4 Storage Requirements Analysis

Related to the storage requirements, Table 9.6 summarizes the length of the elements generated by the BBS and the ACJT schemes, and the corresponding storage to be reserved for group members and group manager. Regarding to the BBS scheme,

Table 9.6: Storage requirements for users and group manager.

Type	σ (bits)	*gpk* (bits)	*gsk[i]* (bits)	*gmsk* (bits)	**User storage** (bits)	**Group manager storage** (bits)
A	4128	6240	1208	336	7448	$1208 \cdot n_{users} + 6576$
A1	6192	6240	1552	1024	7792	$1552 \cdot n_{users} + 7264$
D	2064	3520	520	336	4040	$520 \cdot n_{users} + 3856$
E	7200	12384	2232	336	14616	$2232 \cdot n_{users} + 12720$
F	2016	2688	504	336	3192	$504 \cdot n_{users} + 3024$
ACJT	40400	4104	10240	2048	14344	$2048 \cdot n_{users} + 6152$

using type F pairing, shorter elements are generated, so it could be the pairing type suitable for applications where the storage and the amount of data transferred are critical. Elements generated using type D pairings are also short and they are only little longer than type F elements. Instead, type E pairing generates longer signatures and keys because elements from group $\mathscr{G}_1$ are six times longer than the same elements produced by type D pairing. It is due to the very low embedding degree ($k = 1$) of type E curves. As pointed out starting §2.3.3, ACJT scheme, due to the fact that it relies on the Strong-RSA assumption, outputs longer signatures and keys than the BBS scheme. Considering the same security strength, a signature made by ACJT could be up to 20 times longer than a signature generated by BBS using a type D pairing. This result is the evidence that a pairing-based group signature such as the BBS scheme could be better than a Strong-RSA group signature like ACJT in terms of storage and bandwidth requirements.

9.3.5 Summarizing Performance Results

Summarizing all the performance results, we can conclude that from the point of view of the group manager scalability, it would be preferable the use of type F pairings. It is because type F curves generate the shortest elements and it is also the fastest pairing producing all the required private keys for group members no matter the size of the group. However, since this process could be performed offline by the group manager (typically a powerful server), we should prioritize performance for users. Thus, for group members, it is better to use the type D pairing if members use only devices like laptops and type A pairing if members use also smartphones. It is important to note that precomputation is mandatory to use efficiently the BBS group signature on mobile devices. If storage requirements are a bottleneck, type D pairing is the most suitable pairing since it generates only slightly larger elements than type F curves, so users are not affected so much. But the use of the type F pairing could be better in order to protect the system from improvements of discrete log attacks to the base fields, as pointed out in §9.2.1, due to its larger embedding degree ($k = 12$). However, currently it cannot be used by smartphones as performance

results prove. Finally, even though ACJT computes and verifies signatures as fast as the BBS scheme, ACJT presents the problem, as already proved, that outputs longer elements than BBS.

9.4 Concluding Remarks

We have built, to the best of our knowledge, the first successful implementation of the pairing-based BBS group signature scheme. The implementation allows us to provide many performance benchmarks obtaining a series of interesting results about the scheme. These results have been analyzed and compared with the ACJT group signature, the state-of-the-art non-pairing group signature, which we have also implemented. The comparison has been conducted using a common security level of 80 bits. First, pairings A, D and F seems to be the most interesting options, but the choice among them depends on the requirements of the application and the existence of group members using the algorithm by means of mobile devices, such as smartphones. In addition, a good choice on the type of pairing and its underlying curve parameters could be fundamental to obtain a good performance, making the use of group signatures feasible in real scenarios and applications. Moreover, we have proved that the BBS scheme generates shorter elements than ACJT, as expected. Besides, we have detected some interesting facts about which are the pairing operations with larger computational cost on mobile devices.

Finally, we hope that our effort implementing and giving performance results about these group signatures will be very useful for further works for all kind of applications that use group signatures and, in general, to the cryptographic use of pairings on both computers and mobile devices.

CHAPTER 10

μEasyPay: Implementation and Performance Analysis

In this Chapter we describe the prototype design, how the implementation has been conducted and the performance analysis of our *μEasyPay* micropayment scheme proposed in Chapter 3.

10.1 Introduction

In Chapter 3 we already pointed out that the *μEasyPay* solution is a general, secure and efficient micropayment proposal to pay for low value purchases. Now, in this Chapter we prove that it is actually efficient by means of a real implementation and performance analysis. We describe the implementation of *μEasyPay* as a concrete application to pay for the access to a LBS. Then, we provide a discussion about the performance obtained using real mobile devices. We follow the guidelines and considerations stated in Chapter 8 about the common implementation scenario.

This Chapter is organized as follows. In Section 10.2 we describe how the user experience for customers is. To do that, we create a concrete service in which customers can search for goods or services in her surroundings in a location-based way. Later, in Section 10.3 we discuss the performance results obtained by our implementation, proving that our scheme is suitable to be used in a real mobile scenario with current smartphones. Finally, Section 10.4 concludes the Chapter.

10.2 Implementation

In order to not only present a performance analysis, we also provide an example of a service that could be paid with our micropayment scheme. As pointed out in §3.6, the proposed micropayment scheme fits both security and legal requirements to be applied to LBS. Therefore, we have applied our proposal to the case of access to LBS subject to payment. In this service example, a customer acquires a coin with a number of microcoupons to pay a location-based provider who acts as a merchant. Provider offers location-based responses providing customer with information related to her surroundings, such as the localization of Points Of Interests (POIs) (restaurants, hotels, cinemas close to her current position, etc.). Then, the provider sends to the customer application the response with the localization of the requested places and these are placed on a geolocalized map, using the Google Maps API, together with her current location, so customer can easily find the desired places. Thus, it is a clear example of an application to buy small pieces of information (location data) using an efficient micropayment scheme to pay small amounts of money in each spending transaction.

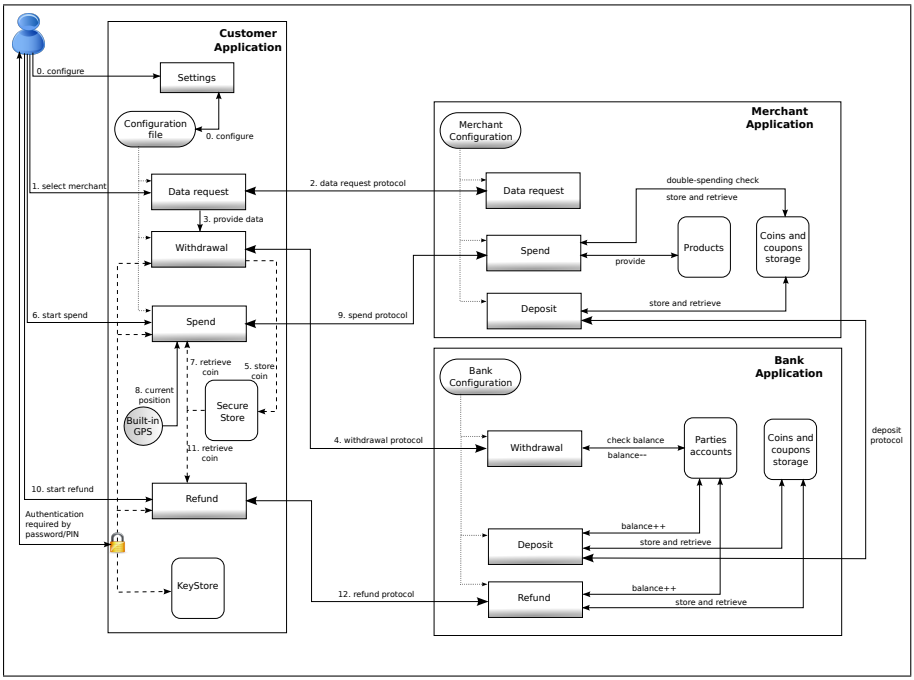


Figure 10.1: System workflow: interactions between customer application, bank and merchant servers.

In order to implement the described service together with the micropayment scheme, we identify the system workflow represented in Figure 10.1. The workflow represents all the interactions between customer application and servers, i.e. merchant and bank. So, we can distinguish three key modules in the service design and implementation: the client application and the provided experience to its users; the server platform; and finally the messages exchanged during protocol executions. However, there are another module that it is fundamental to put the service into production: the partially blind signature implementation. Below we describe further each module.

10.2.1 Partially Blind Signature Implementation

As described in Chapter 3, the process of withdrawal of a coin has been designed using a partially blind signature scheme [55] to provide anonymity and untraceability to customers when they spend microcoupons with merchants. So, the first implementation task is the development of the partially blind signature scheme. It has been a difficult task, among other considerations, because modular operations are mixed with hash operations and they use different Java types to represent data. So, we have developed some methods to transform data from one primitive type (like byte streams) to another type to apply mathematical operations (such as BigInteger type). Moreover, we have also developed methods to encode and serialize values from byte streams into strings ready to be set in XML messages. Of course, the inverse process is also necessary. Mathematical operations have been computed using the Java BigInteger class and the Bouncy Castle cryptographic library (see §8.3 and especially §8.3.3). The most complex operations, due to the number of modular operations combined with hash computations are $\gamma = \mathscr{H}(\Gamma)^d \left(\alpha\left(\lambda^2+1\right)\beta^{-2}\right)^{2d} \mod n$ (during the step 4th of the `Withdrawal` protocol) and the equation to verify $\mathbb{C}_{\mathscr{PBS}}$ (see §3.5.3) . Figure 10.2(a) shows a piece of Java code developed to compute the partially blind signature and Figure 10.2(b) the corresponding code to verify it.

10.2.2 Customer Application and User Experience

Although the micropayment scheme has been designed to be as simple as possible in order to increase the efficiency and decrease computational, storage and network costs, the implementation needs to be even simpler keeping in mind that maybe customers do not have previous skills in similar payment methods. So, our implementation minimizes the effort that customers have to do to use the scheme. So much so that we have made transparent for the customer the withdrawal process since once customer chooses the target merchant, the application collects the required information to withdraw the coin. It means that with a single action (browse

```
BigInteger FI = beta.modInverse(bank.rsaNBank);
BigInteger GAMMA1 = Utils.getHash(commonInfo).modPow(bank.rsaDBank,
    bank.rsaNBank);
BigInteger GAMMA21 = lambda.modPow(new BigInteger("2"),
    bank.rsaNBank).add(BigInteger.ONE).mod(bank.rsaNBank);
BigInteger GAMMA22 = beta.modInverse(bank.rsaNBank).modPow(new
    BigInteger("2"), bank.rsaNBank);
BigInteger GAMMA2 = alpha.multiply(GAMMA21).mod(bank.rsaNBank);
GAMMA2 = GAMMA2.multiply(GAMMA22).mod(bank.rsaNBank);
GAMMA2 = GAMMA2.modPow(bank.rsaDBank.multiply(new BigInteger("2")),
    bank.rsaNBank);
BigInteger GAMMA = GAMMA1.multiply(GAMMA2).mod(bank.rsaNBank);
```

(a) Piece of code referred to the $\mathscr{PBS}$ computing process.

```
String M = coin.getW0User().toString(16)+coin.getW0Provider().toString(16);
BigInteger checkCoin1 = coin.getOmega().modPow(bank.rsaEBank, bank.rsaNBank);
BigInteger checkCoin2 = Utils.getHash(commonInfo).mod(bank.rsaNBank);
checkCoin2 = checkCoin2.multiply(Utils.getHash(M).modPow(new BigInteger("2"),
    bank.rsaNBank)).mod(bank.rsaNBank);
checkCoin2 = checkCoin2.multiply(coin.getDelta().modPow(new BigInteger("2"),
    bank.rsaNBank).add(BigInteger.ONE).
  mod(bank.rsaNBank).modPow(new BigInteger("2"),
    bank.rsaNBank)).mod(bank.rsaNBank);
if (!checkCoin1.equals(checkCoin2))
  throw new Exception("Verification of the partially blind signature does not
    hold");
```

(b) Piece of code refered to the $\mathscr{PBS}$ verification process.

Figure 10.2: Code snippets of the implemented $\mathscr{PBS}$ scheme.

the list of merchants and chose one of them), customer application executes the `InitialReq` protocol and immediately the process starts the `Withdrawal` protocol without more interactions from customer. It makes the user experience more comfortable and it also reduces possible mistakes from the customer input.

Since the application is required to use some private information stored in the customer smartphone (RSA keys and data related to the coin and microcoupons), we follow the recommendations stated in §8.3.2 to protect the access to that information in a secure way.

Customer application is defined by means of five instances to the main class of the Android UI (`android.app.Activity` class), as already represented by the scheme workflow in Figure 10.1:

- *DataReqWithdrawalActivity.* This is the first view of the customer application (Figure 10.3(a)). It presents to the customer the list of available services offered by registered merchants. Every row describes the number of available requests and the price for each of them. When the customer selects one of the merchants, a request of a new and specific coin for the desired merchant is sent

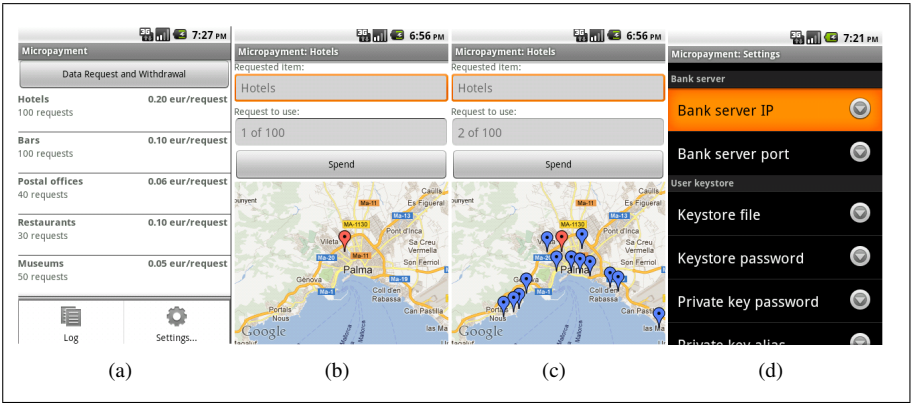


Figure 10.3: Screenshots from the developed Android customer application. (a) DataReqWithdrawalActivity view before to select a targe merchant. (b) SpendActivity view before execute the `Spend` protocol, showing the current customer's localization. (c) SpendActivity view after an execution of the `Spend` protocol, with results localized on map. (d) SettingsActivity view to configure some parameters from the application.

to the bank. So, this view covers operations described by the `InitialReq` and the `Withdrawal` protocols. If the customer is already registered by the bank and in case she has enough funds in her account, the application address customer automatically to the SpendActivity view (Figure 10.3(b)). Otherwise, an error message is sent to customer.

- *SpendActivity.* In this view, the customer is geolocalized on the map (Figure 10.3(b)) and when she clicks on the *Spend* button, the `Spend` protocol starts. So, the application sends the number of microcoupons required to fulfill the value of that request. Then, the application shows on the map the list of received results from the merchant (Figure 10.3(c)).

- *RefundActivity.* This activity starts the `Refund` protocol requesting to the bank a reimbursement of her unused coupons. After run this activity, application presents a confirmation message reporting whether it has been successful or not.

- *LogActivity.* All actions and messages exchanged during an application session are collected and logged by a background process. Then, this activity shows the log of the application.

- *SettingsActivity.* It allows the customer to modify various settings related to

the application configuration, such as network addresses, paths and credentials (Figure 10.3(d)).

10.2.3 Server Entities

Entities running as servers, such as merchant ($\mathscr{M}$) and bank ($\mathscr{B}$), have been developed as an independent and standalone server instances. Both servers have been developed using the Java JDK 6.0 platform. In the same way the customer application does, servers have to store some private data (public and private keys, digital certificates, etc.). To do so, servers use the same method as customer application: the Java KeyStore.

In the presented prototype, the bank server stores in a local file the list of accounts from parties (both customers and merchants) and their corresponding funds. Besides, merchant server stores received coins, microcoupons, indexes (as described in the scheme definition as the triple $\left(\mathbb{C}, \omega_{j\mathscr{C}}, j\right)$) and the current state of each step of the protocol in a temporal in-memory data store. Because it is the first prototype and we want to prioritize the performance testing, we do not consider the use of a full-featured database (such as MySQL or similar one).

10.2.4 Messages

Messages exchanged among parties have been encoded using XML and messages are digitally signed by means of XML Signature standard [168].

Besides, RSA keys have been carefully computed to achieve the proposed security strength of 80 bits, as stated in §8.3.4.

10.3 Performance Evaluation

After we have described the implementation of our *μEasyPay* micropayment scheme, in this Section we give a performance analysis in order to prove its efficiency and usability on a real mobile testing scenario. We describe the test scenario in §10.3.1 and we analyze and discuss the performance results in §10.3.2.

10.3.1 Test Scenario

Figure 10.4 represents the considered test scenario. It is composed by the testing Laptop, where servers run, and the Desire and Wildfire Android smartphones, where the customer application runs. Table 8.3 from Chapter 8 lists all technical features of these devices. Connectivity between the client side and servers is provided through the use of a 802.11g wireless network, while interactions between servers are done

through laptop's localhost network interface. Finally, each test has been performed at least 20 times to obtain average measures.

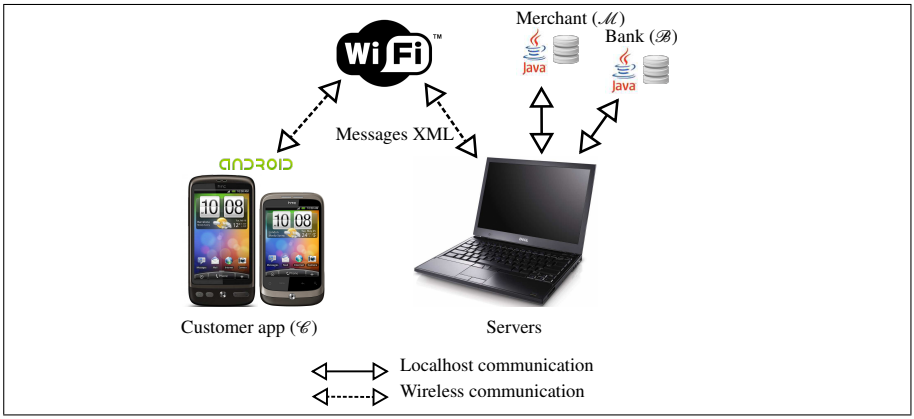


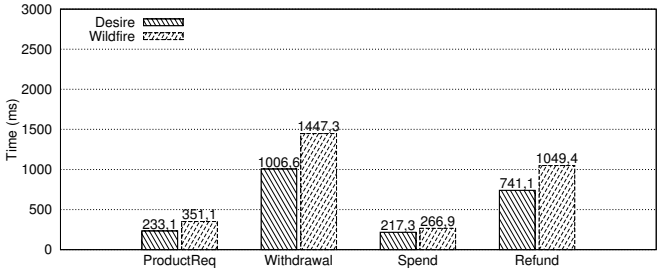
Figure 10.4: Test scenario.

10.3.2 Discussion of Results

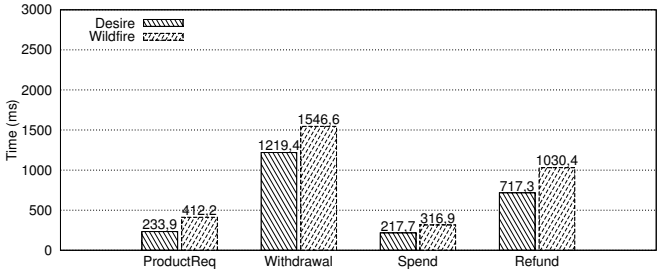
We are going to discuss the performance of protocols running on customer application because we want to focus the analysis for the client side as it usually disposes of less computing and bandwidth resources than servers. Therefore, we analyze `InitialReq`, `Withdrawal`, `Spend` and `Refund` protocols. `Deposit` protocol is performed between two entities running on servers, so it is difficult to become a system bottleneck. Besides, customer is not interested on the performance of `Deposit` protocol due to the fact that it does not perceives its execution.

Despite the fact *μEasyPay* has been designed to avoid as much as possible the use of asymmetric cryptographic operations (especially during the `Spend` protocol) and we have also set a common and constant security strength of 80 bits, it is also useful to prove what happens in case the scheme needs to use longer RSA keys. So, Figure 10.5 represents how RSA key length affects the performance of each protocol. Note that we have supposed the use of coins with two microcoupons.

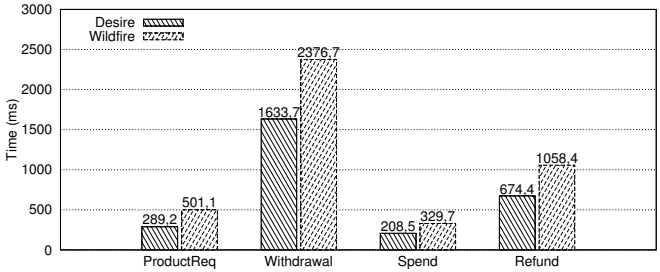
First, only the `Withdrawal` protocol reduces its performance due to the growth of the RSA key length. However, the difference is only notable whether key length is increased from 2048 bits to 4096 bits. So, it is not a problem to use a higher level of security through the use of a 2048-bit RSA key, because the protocol takes only 1.2 seconds in the Desire smartphone and around 1.5 seconds in the Wildfire device. However, the time spent for all protocols is almost the same whether we use 1024-



(a) `Withdrawal` protocol with 1024 bits RSA modulus.



(b) `Withdrawal` protocol with 2048 bits RSA modulus.



(c) `Withdrawal` protocol with 4096 bits RSA modulus.

Figure 10.5: Time required to comple the `Withdrawal` protocol depending on the RSA modulus length.

bit or 2048-bit keys. Indeed, it proves as weightless is our *μEasyPay* micropayment scheme referred to the use of asymmetric cryptography.

Secondly, this analysis also proves by performance that the `Spend` protocol does

not use asymmetric cryptography in any step. `Spend` protocol remains constant (about 200 ms in Desire and 400 ms in Wildfire) to complete the execution without depending in any case of the RSA key length. Besides, it is a very fast process even it is executed in low-class smartphones, such as Wildfire, so we have proved that `Spend` protocol is actually an efficient protocol.

Finally, as opposed to the `Spend` protocol, the `Refund` protocol uses asymmetric cryptography in order to digitally sign the initial request sent to the bank. However, it only takes around 700 ms on Desire and around 1 second on Wildfire. It is worthwhile to give an explanation to justify why `Withdrawal` protocol reduces its performance as RSA key length increases while `Refund` protocol not. A reason could be related to the partially blind signature computation bythe customer. In fact, RSA public key is involved up to four complex operations during the `Withdrawal` protocol. Instead, the RSA signature algorithm used during the `Refund` protocol is only used once and in addition, the regular RSA signature algorithm is more efficient than the partially blind signature.

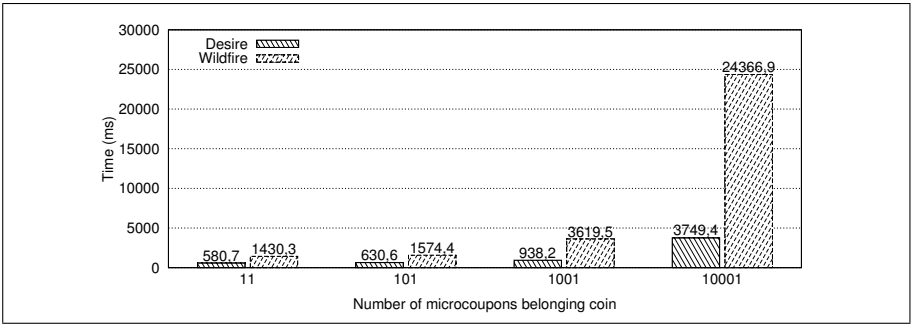


Figure 10.6: `Withdrawal` protocol performance depending on the chain length for each smartphone (considering RSA modulus of 1024 bits).

Figure 10.6 shows the relation between the required time to run the `Withdrawal` protocol depending on the number of microcoupons belonging the coin. Note that it is related to the length of the hash chain, i.e. the number of hash identifiers belonging $\mathbb{C}_\omega$. As explained in §3.4, $\mathbb{C}_\omega$ has up to $2N+1$ hash identifiers (the $+1$ represents the chain identifier), i.e. N microcoupons. Analyzing Figure 10.6, we detect that the cost to withdrawn a coin remains very low up to 1000 microcoupons. In this particular case, Desire requires approximately 1 second while Wildfire takes about 4 seconds. However, the gap between both smartphones increases as length of the $\mathbb{C}_\omega$ element growths. In fact, Wildfire becomes impractical to withdraw a coin with a number of microcupons from 10000, because it takes more than 24 seconds to execute the `Withdrawal` protocol. Instead, Desire allows requesting a coin with this

number of microcoupons and probability it will continue to be practical to request coins with longer $\mathbb{C}_\omega$. Therefore, the bottleneck parameter in the `Withdrawal` protocol is the hash chain procedure to generate the desired number of hash identifiers belonging $\mathbb{C}_\omega$, in particular for smartphones with limited resources. In case it could be precomputed in advance by the customer application, it will allow to request coins with longer number of microcoupons. However, practical size of $\mathbb{C}_\omega$ will be never as long because we are considering coins with low value, suitable for micropayments, instead of coins to pay for large amounts of money. For example, let us suppose a practical example in which the worth value of a single coin is 20€ (in general, it is the maximum value carried by coins to consider its use in the micropayment scenario, because from 20€, payment methods are usually classified as full payments [13]), and considering a worth value for each microcoupon of about 5 cents, then it means that the number of microcupons belonging $\mathbb{C}_\omega$ should be 400. As already proved, for this number of microcoupons, the `Withdrawal` protocol is really efficient to withdrawn coins to be used in micropayments.

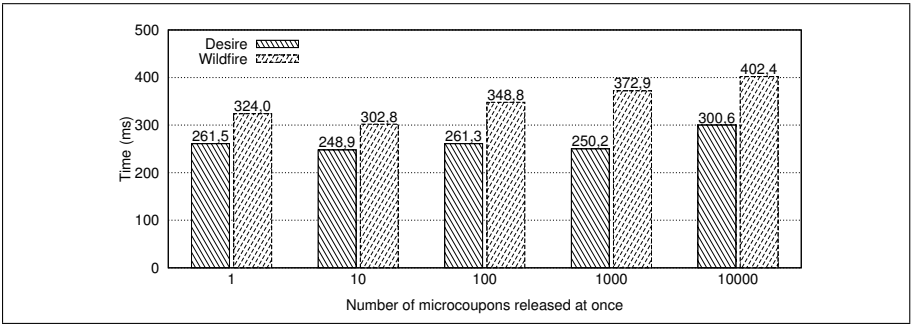


Figure 10.7: `Spend` protocol performance depending on the number of coupons released at once for each smartphone.

Figure 10.7 shows a comparison of required time for the `Spend` protocol depending on the number of microcoupons released at once to pay for the desire item. Figure 10.7 shows that performance remains constant as well as the required time is very low regardless the number of microcoupons released within a single `Spend` run. In fact, the scheme allows customers to pay for an item releasing up to 10000 microcoupons, requiring it no more than 300 ms on Desire and about 400 ms on Wildfire. Thus, we can pay for an item that costs 10000 microcoupons (even this number of microcoupons will never be used following the reasoning provided above for the `Withdrawal` protocol), using more or less the same time as paying for an item of only 10 microcoupons. It is due to the fact that the `Spend` protocol only transfers a single microcoupon regardless the total number of microcupons required to pay. As a difference from the `Withdrawal` protocol, during the `Spend` protocol

is the merchant who is in charge to apply hash chain procedure on the list of received microcoupons. As merchant runs on a powerful server, he can execute this process faster, so the global protocol performance only decreases linearly and slowly. So, the `Spend` protocol is scalable in relation to the growing number of coupons needed to pay for the item and it remains efficient regardless whether the customer device has low resources.

Regarding the data transferred and stored during the `Spend` protocol, it depends mainly on the size of the coin. Its size depends on the used RSA modulus and the hashing algorithm, since the size of the partially blind signature also depends on them. For example, in case we use a RSA modulus of 1024 bits and SHA-1 as the hash algorithm, the coin size is around 300 bytes plus some common information of variable size. Thus, the cost for the bank and the merchant to store a coin and the last microcoupon spent by customers is very low: they should store only 300 bytes due to the coin, 20 bytes due to the last received microcoupon and some bytes to store the chain index.

10.4 Conclusion

We have proved that our scheme is efficient, scalable and ready to use on customer devices such as current smartphones no matter they have low computing features. Regarding the `Withdrawal` protocol, we have demonstrated that it is really efficient to withdraw coins to be used in a micropayment scenario regardless it is used by either Desire or Wildfire smartphones. Since the `Spend` protocol does not use expensive asymmetric cryptography, the process can be very fast. Moreover, the protocol allows a scalable and efficient spending process regardless the number of microcoupons released at once to pay merchants. It grants customers to pay for more expensive items with the same computational and communication cost as for cheaper items.

Concerning storage requirements by server entities (merchants and bank), our scheme also optimizes the required storage to reduce costs, since both bank and merchants only need to store less than 350 bytes for every not yet expired coin. In addition, note that this data can be erased when the coin are no longer valid, as stated by the scheme's description.

So, the *μEasyPay* scheme has been proved to fit all functional requirements stated by the functional model described in §3.2.2.

CHAPTER 11

$\mathscr{MC}-2\mathscr{D}$: IMPLEMENTATION AND PERFORMANCE ANALYSIS

This Chapter shows the implementation and the performance analysis of the $\mathscr{MC}-2\mathscr{D}$ solution presented in Chapter 4. Moreover, we compare our solution with the previous scheme that deals with the same multi-merchant scenario to prove that $\mathscr{MC}-2\mathscr{D}$ outperforms that previous multicoupon solution.

11.1 Introduction

In this Chapter we describe the implementation and the performance analysis of the $\mathscr{MC}-2\mathscr{D}$ multicoupon scheme proposed in Chapter 4. As already pointed out in the theoretical proposal, our multicoupon scheme for a multi-merchant scenario improves a previous proposal by Armknecht [25] both in security and performance (see the complexity comparison in §4.6). In order to prove also by implementation that our $\mathscr{MC}-2\mathscr{D}$ scheme outperforms the previous proposal, we provide a practical computing performance comparison between both schemes in similar conditions. Besides, armed with a successful $\mathscr{MC}-2\mathscr{D}$ implementation on the Android Platform, we also provide an extensive performance analysis taking into account both computation and communications in a real deployment scenario. We compare the use of different networking technologies (Wi-Fi and 3G) and the implications on the resulting messages size due to how data is actually encoded (XML and ASN.1).

The organization of this Chapter is the following. In Section 11.2 we describe the implementation of $\mathscr{MC}-2\mathscr{D}$, focusing on the customer application. Section 11.3 provides a computing comparison between the previous multicoupon solution and our proposal. Next, Section 11.4 discusses performance measures achieved by using a real deployment scenario. Finally, Section 11.5 concludes this Chapter.

11.2 Implementation

In this Section we describe how we have implemented the $\mathscr{MC}-2\mathscr{D}$ scheme. Server entities have been developed in the same way as in the implementation of *μEasyPay*. So, we address the reader to §10.2.3 for this particular piece of development. We focus this Section on describing how the customer application has been developed. Note that since $\mathscr{MC}-2\mathscr{D}$ uses group signatures, the library and the knowledge acquired in Chapter 9 are as well applied to this implementation.

11.2.1 Customer Application and User Experience

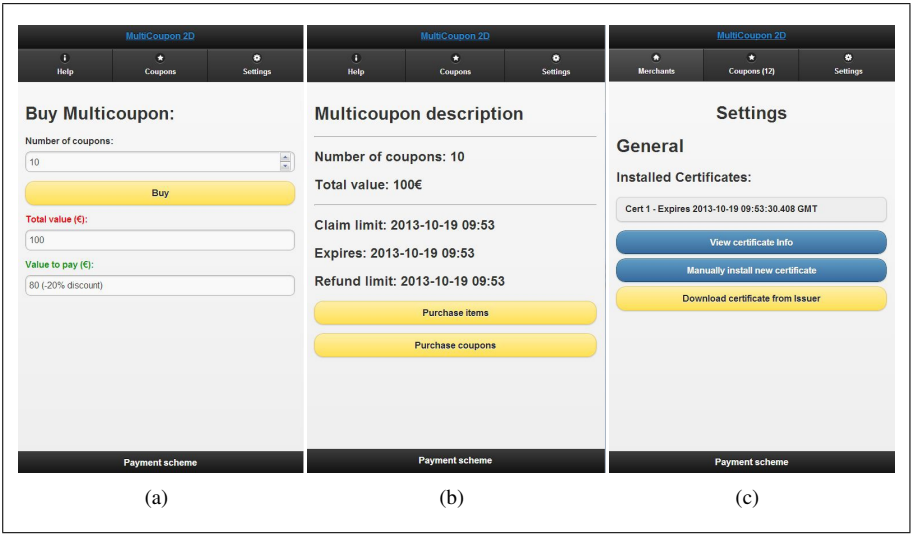
To develop the customer application, we have to take into consideration the same arguments stated in §10.2.2 for the *μEasyPay* scheme: simplicity and secure storage of sensitive data. On one hand, our implementation tries to minimize the customer effort at the time of using the scheme to improve its simplicity. On the other hand, customer application follows the guidelines to store private data described in §8.3.2, so sensitive data is protected.

Regarding to the application's UI, it has been organized with at least seven windows to manage all processes related to each protocol defined by the multicoupon scheme together with some views to manage multicoupons such as remaining coupons, parameters from certificates, etc. As a difference from the *μEasyPay* scheme, we have implemented the UI with PhoneGap although internal processes have been developed with Java using the Android Platform. Figure 11.1 shows three views from the customer application.

The following list summarizes the functionalities of each application view:

- *RegisterToGMView*. This view allows the customer to request a registration to the group manager. It shows customer terms and conditions of the service and it allows to start the registration protocol. At the end of the protocol, this view stores in a secure way (as stated before) the received data from the group manager: the group credentials.

- *IssueNewMCView*. If customer already has group signature credentials, she can execute the `Issue` protocol (see Figure 11.1(a)). In order to do that, customer chooses how many coupons should compose the multicoupon they want



(a) (b) (c)

Figure 11.1: Screenshots from the developed Android customer application. (a) *IssueNewMC* view. (b) *ManageMC* view. (c) *Settings* view.

and their unit-value. After that, the `Issue` protocol starts, and without any more interaction from the customer, the multicoupon is received and stored.

- *MultiRedeemView*. This view allows the customer (in case she already has previously issued a multicoupon) looking for merchants already affiliated to the issuer. Every merchant could be classified by its type of activity such as hotel accommodation, car rental and so on. When customer is interested in something offered by a particular merchant, she starts the `Multiredeem` protocol only clicking a button. In case customer has more than a single multicoupon stored, the application requests customer to choose which one to redeem. The application returns the result of the multiredeem execution and the customer receives the desired good or service: the service itself in case it is digital or a receipt in case it is a physical good.

- *RefundView*. This view allows the customer to request issuer a reimbursement of her unused coupons executing the `Refund` protocol. The view is so easy because it only has a simple button to start the refund procedure. In case there are more than one multicoupon stored in the secure storage, the application notifies customer and allows her to choose which multicoupon has to be used.

- *ManageMCView*. This view provides customer with an interface to manage her multicoupons (see Figure 11.1(b)). Several data can be showed about each

multicupon: how many coupons are remaining, the expiration date, the history of purchases, etc.

- *SettingsView*. It allows customer to configure some application settings, such as install new personal certificates among others (see Figure 11.1(c)).
- *LogView*. Since application logs all activities performed by customer, this view is useful to check them. It can be configured to show either informative messages (such as protocol beginning and finishing events) or more complex information (such as full message contents).

11.3 Performance Comparision with the Armknecht's Scheme

In addition to the analytic complexity analysis presented in §4.6, based on the number of long and short exponentiations, we provide a real performance comparison based on the performance evaluation achieved by the author in [116] of the unique previous solution designed for a multi-merchant scenario presented in [25]. We have reproduced the same tests in order to compare both proposals in the same conditions. Our performance evaluation measures, as the author of [116] does, the time required to issue and to redeem a group of five coupons ($k = 5$), and the overhead produced by every new coupon issued or redeemed ($k + 1$). Due to the fact that he only take into account the required computational time without compromising network communications among entities, we also consider a scenario with a single laptop. His testing machine was a laptop equipped with a single-core CPU running on 1.7GHz using a Linux distribution. In our case, in order to approximate to the features of their machine, our laptop (see Table 8.3 in §8.3.5) has been underclocked to a single-core CPU running on 1.6GHz. Table 11.1 sets, side by side, the performance results claimed in [116] compared with the results obtained by our proposal implementation.

Table 11.1: Multi-merchant solutions - An implementation comparison (in seconds).

	Issuing				**Redeem**			
	$k = 5$ coupons			$k+1$	$k = 5$ coupons			$k+1$
	$\mathscr{C}$	$\mathscr{I}$	Total		$\mathscr{C}$	$\mathscr{M}$	Total	
[25]	-	-	4.280	0.811	-	-	33.01	6.476
Our proposal	0.023	1.182	1.205	< 0.005	0.877	1.204	2.082	< 0.02
Our proposal*	n/a	n/a	n/a	n/a	0.093	1.204	1.297	< 0.02

* - applying group signature precomputation in the customer side during the multi-redeem protocol
n/a - not applicable

Regarding to the issuing process, our protocol is around 3.5 times faster than the issuing protocol proposed in [25]. Moreover, the customer needs only 23 ms of computation to obtain the $\mathbb{MC}^{2D}$, so our `Issue` protocol is very lightweight for the customer. It is very important because we have to think that the customer application will be executed in a device with limited resources (e.g. a smartphone or a PDA), so the computational requirements should be minimized.

Concerning the redeem procedure performance, the results are also better than [25]. In fact, the required time to redeem 5 coupons with our `Multiredeem` protocol is around 15 times lower than the time needed by the redeem protocol of [25]. Note that our protocol performance can be further improved if we apply precomputation techniques in the customer side during the `Multiredeem` procedure when she is required to generate a group signature (as analyzed in Chapter 9). The precomputation allows the customer to generate and store values before starting the `Multiredeem` protocol, obtaining an improvement better than of 90% over the required time to do the same operations without enabling the precomputation, and it can be 25 times faster compared to [25].

The required time to issue or redeem an additional coupon is negligible in our proposal, because the overhead introduced is only due to the generation and the verification of two additional hash operations (to build the additional coupon composed by the payment and proof information), so this value is very low. Hence, we achieve an efficient and scalable solution since our proposal is independent of the number of coupons in a multicoupon unlike the proposal in [25].

11.4 Performance Discussion

In this Section we present the performance measures obtained from a realistic and practical implementation of our multicoupon scheme with mobile devices and taking into account the considerations stated in §8.2. First, we describe in §11.4.1 the test scenario we have used and then we analyze the results. We analyze the two main factors not considered until now by other multicoupon solutions. In §11.4.2 we analyze the length of messages depending on the method to encode them and the implications on the response time perceived by the customer while in §11.4.3 we evaluate in more detail the results, distinguishing processing, encoding/decoding and networking activities.

11.4.1 Test Scenario

In line with the considerations we have stated in §8.2 and the reasoned choices we have made in §8.3, Figure 11.2 represents the test scenario we have used to obtain performance measures of the $\mathscr{MC}-2\mathscr{D}$ solution. Remember that Table 8.3 describes

the list of main technical features of selected testing services and devices together with a short name to identify them easily during the analysis.

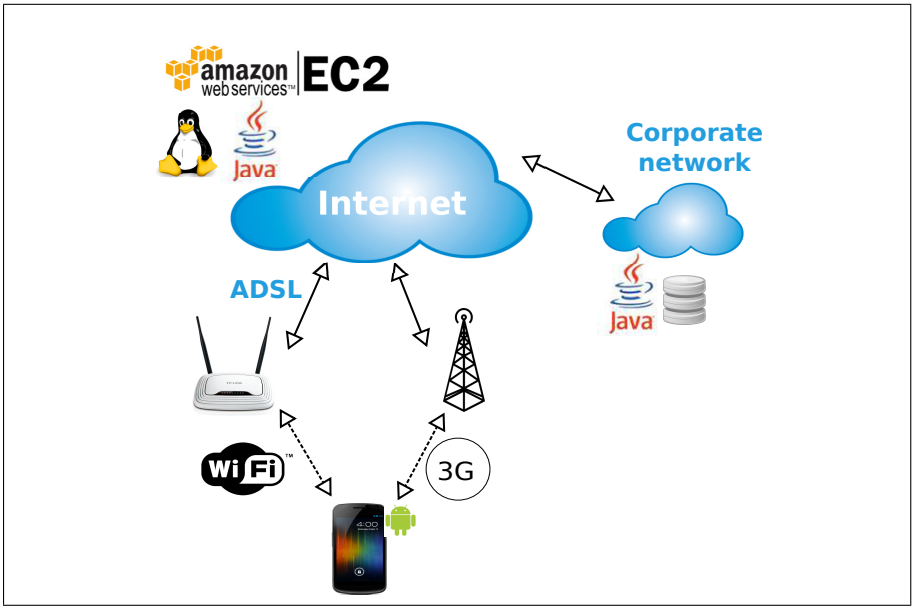


Figure 11.2: Considered performance testing scenario.

Regarding to the server platform, as already stated in §8.3.1, we have selected the EC2 cloud solution (we refer it as `CS` during the analysis). Besides, we have also considered the use of a virtual server (called `VM`) to test the delay perceived by customers when a merchant performs an online claim with the issuer during a `Multiredeem` protocol.

As stated in §8.3.1, we have used two different wireless network technologies: a 802.11g local wireless network connected to Internet via a commercial Asymmetric Digital Subscriber Line (ADSL) and a 3G mobile network. Table 11.2 summarizes features (average bandwidth and latency) from network connections among servers and customer.

Table 11.2: Network features of each communication path (estimated average).

Communication Path		Bandwidth (est. avg.)		Latency (est. avg)
Origin	**Target**	**Downstream**	**Upstream**	
`CS`	`VM`	>25Mbps	>25Mbps	<100 ms
Customer app (ADSL)	`CS`	<3Mbps	<0.3Mbps	>100 ms
Customer app (3G)	`CS`	2-8Mbps	2-4Mbps	>200ms

Thus, the considered testing scenario replicates very well what happens in a real production scenario. On one hand, mobile customers are often connected to wireless networks where transmission rates are limited and depend on several factors. On the other hand servers have a lot of available resources.

We have executed performance tests for multicoupon protocols involving customer application, such as the `GMRegistration`, `Issue`, `Multiredeem` without online `Claim` and `Multiredeem` with online `Claim`. Furthermore, we have also analyzed the `Claim` protocol when it is executed offline by the merchant. Note that each test has been called at least 20 times to obtain average measures.

11.4.2 Time response and Network Overhead

Taking into account the testing scenario described above, we present and discuss the performance results obtained by running the multicoupon implementation.

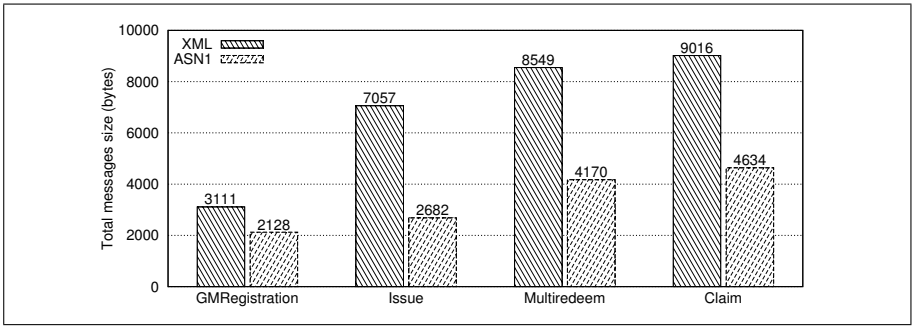


Figure 11.3: Total message lenght for each protocol.

Figure 11.3 shows the results about the length of the messages generated for each protocol considering the influence of the message syntax. Clearly, the use of ASN.1 syntax reduces nicely the network overhead, arriving up to 62% for the `Issue` protocol.

Regarding to the total response time for each protocol, Figure 11.4 shows the results obtained on the customer application installed on Desire and considering the Wi-Fi connection. The most costly protocol is the `Multiredeem` due to the fact that it compromises the execution of cryptographic operations, such as group signatures in first and third steps of the protocol, as described in §4.4. Moreover, the response time is very different whether it is considered XML or ASN.1 formats to encode messages. In fact, for each protocol, ASN.1 contributes to improve the performance perceived by the customer. It is due to both messages length (Figure 11.3) and its encoding efficiency (see §11.4.3).

Also, note the difference between the `Multiredeem` protocol performance results depending on whether the merchant executes an online `Claim` or not. Indeed, the overhead added by the online `Claim` protocol is about 680 ms in case of using XML and about 465 ms if ASN.1 is utilized. Therefore, the use of the ASN.1 encoding format also improves the response time even if the protocol has been executed between two powerful server instances. In fact, the total response time increases due to the `Claim` online execution, but it only adds around 0.5 seconds to the `Multiredeem` without online `Claim`. Thus, these results prove that it is viable for all parties the use of online claims because it does not imply a large overhead.

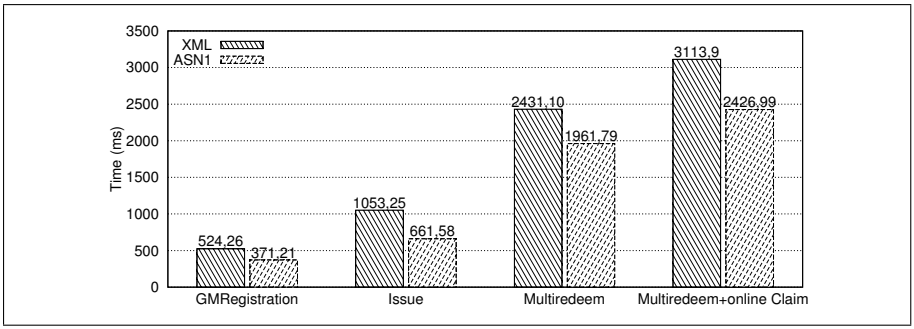


Figure 11.4: Total response time for each protocol (Desire with Wi-Fi).

11.4.3 Analyzing Influence Factors on Computational Performance

Lets us analyze in more detail the obtained results for each protocol and concrete activity for each customer device. The total response time covers three groups of activities:

- The time spent on processing data, building requests and computing received responses. Here, the values depends on the processing power of each device, mainly CPU and memory.

- The time spent on encoding and decoding data to be sent and just being received, respectivelly. We have divided this activity depending on whether the encoding/decoding process has been performed by means of either XML or ASN.1 encoding formats.

- The time spent on network activities, i.e. sending and receiving data through the network. We evaluate the effect of the network access technology used by the customer: Wi-Fi vs. 3G.

First, let us focus now on the processing and the encoding/decoding times for each protocol and customer device. Figures 11.5-11.7 show the time spent by the customer application to execute each protocol.

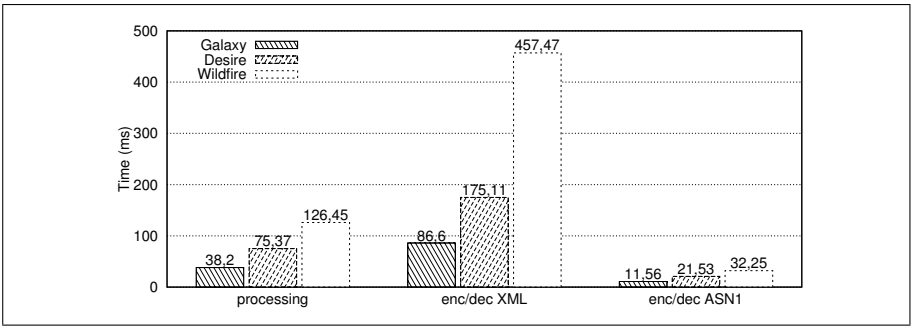


Figure 11.5: `GMRegistration` protocol: processing and encoding/decoding in function of smartphone.

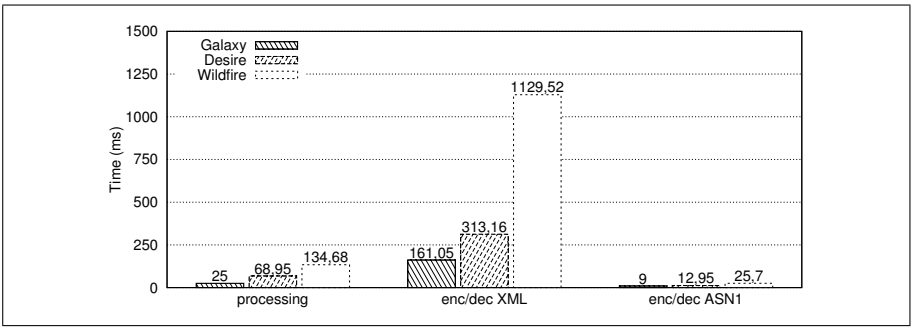


Figure 11.6: `Issue` protocol: processing and encoding/decoding in function of smartphone.

Aside the differences due to computing power of each mobile device, we detect some interesting results. In Figure 11.5 and 11.7 it is notable the gap between encoding/decoding with either XML or ASN.1. In fact, for the better case (considering the most powerful mobile device), encoding with ASN.1 is about 7 times faster than encoding with XML. It becomes even higher as the features of mobile devices are worse. For example, if we observe Figure 11.5 and we evaluate the encode/decode time using the Wildfire smartphone, we see that ASN.1 is more than 14 times faster than using XML. The same trend is observed in the `Issue` protocol, but in this case encoding/decoding with ASN.1 is even better (about 43 times faster). When XML is used, the encoding/decoding activities are the main concern and as computing power

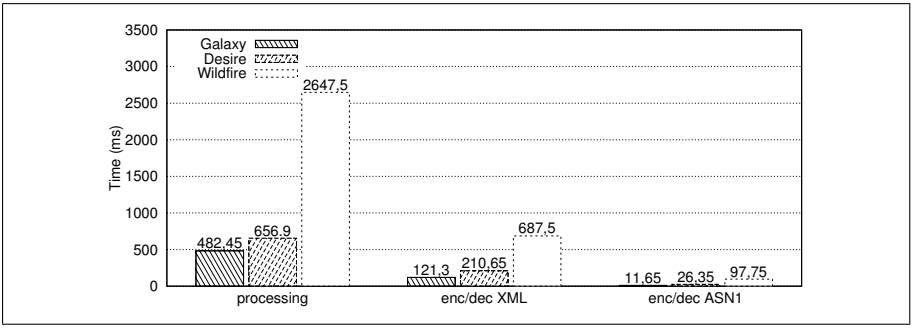


Figure 11.7: `Multiredeem` protocol: processing and encoding/decoding in function of smartphone.

of devices decreases, this time is even more important. Instead, with the ASN.1 format, the processing time is the main concern due to the fact that encoding/decoding with ASN.1 seems to be very optimized.

Next, we study the influence of the network access technology. To do this, the customer application runs on Desire and we choose ASN.1 as the message syntax since this is the better option for efficiency as we have shown above.

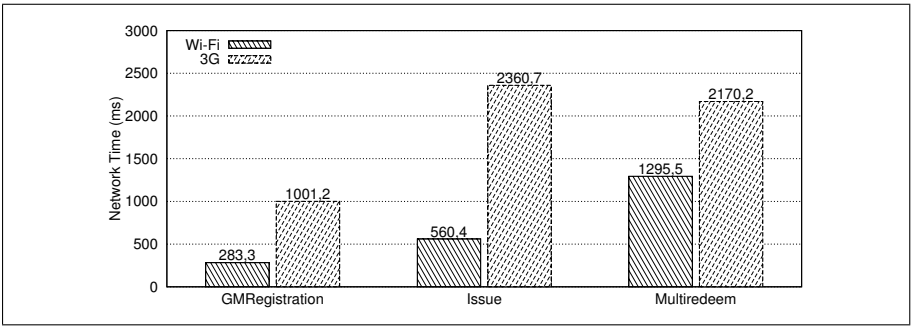


Figure 11.8: Networking response time for each protocol.

Figure 11.8 shows that executing protocols using the Wi-Fi network is a better option than using 3G. It is due to the fact that Wi-Fi is attached to an ADSL Internet connection, which is often more stable and offers better throughput than 3G. In addition, network activities consumes a large amount of time if we compare Figure 11.8 with Figure 11.4 in which the total response time is shown. Therefore, besides the encoding/decoding format, the network throughput (and stability) is a big concern.

11.5 Conclusions

We have demonstrated that $\mathscr{MC}-2\mathscr{D}$ scheme for the multi-merchant scenario is efficient, scalable and ready to be used on mobile devices. In addition, the comparison with the previous proposal dealing with the same scenario proves that our multicoupon scheme outperforms that solution considering a similar test scenario. It is due to some factors such as the use of lightweight cryptography and the ability to issue and redeem multiple coupons within the same transaction.

Besides, we have presented a functional implementation on the Android platform using also another external framework to build the UI: PhoneGap. This framework has been proven really useful to implement UIs even core computations have been developed in Java using libraries from Android and other external libraries. Using this implementation, we have also provided a complete performance evaluation considering a real scenario with remote servers (even using a cloud solution as EC2), taking into account all factors that contribute to the total response time of a protocol. In any case, $\mathscr{MC}-2\mathscr{D}$ has been proved efficient and perfectly usable on all considered smartphones.

CHAPTER 12

AUTOMATIC FARE COLLECTION IMPLEMENTATION AND PERFORMANCE ANALYSIS

This Chapter describes the implementation and the performance evaluation of time-based and distance-based AFC solutions presented in Chapter 5.

12.1 Introduction

In this Chapter we describe the implementation and performance analysis of both time-based and distance-based (the enhanced scheme not vulnerable to the colluding attack) AFC schemes presented in Chapter 5. This analysis will convince that both schemes are suitable to be used by current mobile devices. Besides, it contribute to show the performance differences between time-based and distance-based proposals.

This Chapter is organized as follows. In Section 12.2 we explain how we have implemented both AFC solutions. Later, in Section 12.3 we discuss the performance results obtained by means of the described implementation. Finally, in Section 12.4 we conclude the Chapter with some remarks.

12.2 Implementation

Armed with the development libraries and the methodology described in Chapter 8, together with the implementation and knowledge of the BBS group signature scheme presented in Chapter 9, we have conducted the implementation of the time-based and distance-based AFC schemes.

We have focused our implementation on developing the scheme's core protocols, i.e. `URegistrationMG`, `URegistrationMP`, `SysEntrance` and `SysExit` protocols for both AFC schemes. These protocols are critical, so their performance should be analyzed, especially `SysEntrance` and `SysExit` protocols, due to the fact that they must present a fast response time. In fact, tickets must be issued and validated as fast as possible, even faster than traditional paper-based ticketing systems.

12.2.1 Client Side

For this implementation we have not developed a full UI for the user application, as in previous payment proposals in this dissertation. It is due to the fact that UI is not as important in this kind of schemes than previous ones. In fact, as stated by the service name (**Automatic** Fare Collection), the automatic word implies that the process should be as automatic as possible. However, a minimal interface for users should be described. Therefore, a user application for the AFC service should compromise the following main views:

- Registration. Through this view, the user starts the process of registration with $\mathscr{T}_G$ and $\mathscr{T}_P$ by means of `URegistrationMG` and `URegistrationMP` protocols, resp. It should be as simple as a window with a button to start these processes and a dialog to offer the user the ability to choose her own digital certificate.

- Entrance to AFC service. It only should allow the user to start the `SysEntrance` protocol pushing a simple button.

- Leaving the AFC service. In the same way as the previous view, it only should present user with a button to start the `SysExit` protocol.

Even if the NFC technology could be used without any restriction in the Android platform (see §8.3.1), the user application could also omit entrance and exit views, because the process will become fully automatic when the mobile device is under the coverage of entrance and exit stations. Besides, thinking about an *in-car* equipment to deal with these systems, UI will be integrated in the car's dashboard, and may be without much more than visualizing some confirmation messages.

12.2.2 Server Entities

We have implemented the behavior of every entity to follow the protocol flows. Therefore, the server side compromises the following protocol parties: the group manager ($\mathscr{T}_G$), the payment manager ($\mathscr{T}_P$), the source provider ($\mathscr{P}_{\mathscr{S}}$) and the destination provider ($\mathscr{P}_{\mathscr{D}}$).

12.2.3 Messages

Messages exchanged by involved entities and the result of their processing (such as entrance and exit tickets), has been serialized and encoded to fit the XML format.

As stated in §8.3.4, all involved cryptographic algorithms (such as BBS group signature and digital public key signature) use as input a security strength of 80 bits, enough for this service. Moreover, regarding to group signature, we have selected to deal with type D pairing with a corresponding elliptic curve to fulfill security strength requirements. As proved in Chapter 9, type D pairing offers a good trade-off between computing time and size of output signatures, and it could be also used by mobile devices.

12.2.4 Tickets

As said before, tickets have been encoded with XML. Due to the fact that XML is a textual and human-readable syntax, some parameters obtained by performing either cryptographic or mathematical operations (such as results from exponentiations or results from digital signatures) should be transformed from their original representation (usually byte streams) to textual representation (as well the reverse process). So, all parameters have been previously encoded in a string with Base64 representation. To make an idea about the look of entrance and exit tickets and to see also how parameters described in the theoretical scheme compose real tickets, we represent them in Figure 12.1.

12.3 Performance Evaluation

In this Section we present the performance analysis obtained by the AFC implementation. We describe the test scenario and we discuss later the results obtained taking into account both computing and storage requirements.

12.3.1 Test Scenario

We consider mainly the effects of computation, leaving the networking effects in a second plan. Testing devices have been the Laptop in which all server entities

```
<ticket-in-signed>
   <signature>12ccfd7c8d7547493c65956742e614fc7e12 ...</signature>
   <ticket-in>
      <serial-number>1003</serial-number>
      <sigma-signed>
         <sigma>
            <deltau>44e16ffddeb07d708a8ae63b99f ...</deltau>
            <hk>68274ced5421544110682e963f069262bc13f886</hk>
            <s1>3b0999caaa0643b75ea14c32b2495474708e ...</s1>
         </sigma>
         <signature>
            <t1>38e69f067a998996b48d92fdcf53ab5 ...</t1>
            <t2>1e8146c1bd7bad8d19dc9946e1f531f ...</t2>
            <t3>58596a87276a55231a87c2ab286f892 ...</t3>
            <c>00f2ff7e2a9260dd85f057aaa8111efe739cd9fc3c</c>
            <salpha>00a643919d47a5f8353126ea440dbbd870fd10280a</salpha>
            <sbeta>000921471fa9fb24063401d929d273cacd9719ebcc</sbeta>
            <sx>00525072dc167e12530d8d8f23f22bbbb731f4f925</sx>
            <sdelta1>010ce0e33d714978312b055ffdc563c24a14049b40</sdelta1>
            <sdelta2>0080e4105306edae1f2a13065570e7075f5030cd04</sdelta2>
         </signature>
         <group-public-key>
            <g1>1607b5f1057cebf1b29a99ed149a123 ...</g1>
            <g2>3ada8ae31f589abeeafee19de80606f ...</g2>
            <h>264bf9d4fcf3cf35242fc09e70966522 ...</h>
            <u>0e52bdc10023f4b37b5deb214993d57a ...</u>
            <v>0e52bdc10023f4b37b5deb214993d57a ...</v>
            <omega>032509d31186818eefca97483bb5 ...</omega>
         </group-public-key>
      </sigma-signed>
      <source-provider>1</source-provider>
      <tau1>1308235786900</tau1>
      <validity>1308239386900</validity>
   </ticket-in>
</ticket-in-signed>
```

(a) Entrance ticket with XML encoding.

```
<ticket-out-signed>
   <signature>4b786f526179493445716b4f704a49562b77 ...</signature>
   <ticket-out>
      <destination-provider>5</destination-provider>
      <fare>3.6</fare>
      <serial-number>1001</serial-number>
      <str>leave taking at Thu Jun 16 16:49:26 CEST 2011</str>
   </ticket-out>
</ticket-out-signed>
```

(b) Exit ticket with XML encoding.

Figure 12.1: Entrance and exit tickets with XML syntax.

runs and two smartphones (Desire and Wildfire) to execute the user application. Table 8.3 in §8.3.5 lists the technical description of each device. In addition, we have considered that the user application and server entities are connected through

the same Wi-Fi network. Therefore, the test scenario is equivalent to that already considered by the performance evaluation of the *μEasyPay* scheme (see §10.3.1). Finally, as usual, each considered test has been repeated at least 20 times to obtain average measures.

12.3.2 Discussion of Results

After the protocol implementation we have to analyze and discuss the achieved performance results, from less to more detail. Figure 12.2 shows the total time spent to execute both time-based and distance-based AFC schemes on Desire and Wildfire smartphones. As expected, the complete scheme execution on Desire is faster than on Wildfire (compromising all protocols). On one hand, time-based AFC spends about 113 seconds on Wildfire to complete, while Desire spends only 20 seconds, i.e. the latter presents a better performance of about 5.6 times compared to the former. On the other hand, distance-based AFC presents a similar gap between both smartphones. This way, distance-based AFC needs 134 seconds to complete while Desire only needs 26 seconds, i.e. Desire is about 5.2 times faster than Wildfire. So, performance is decreased more or less in a similar way by both schemes and smartphones. Initially, differences between both AFC schemes are not as large as expected because Desire only expends 6 seconds more in the distance-based scheme to finish a complete execution while Wildfire needs 21 more seconds.

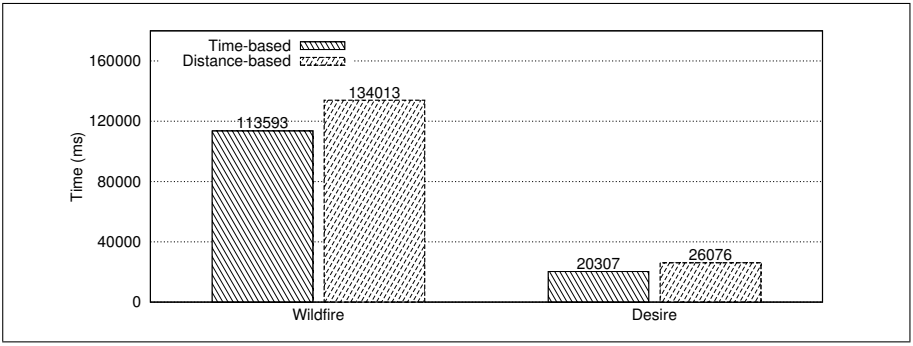
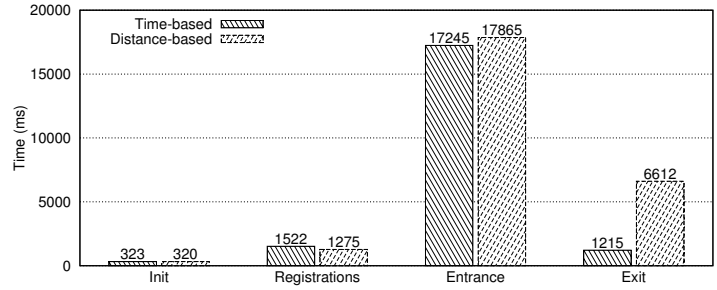


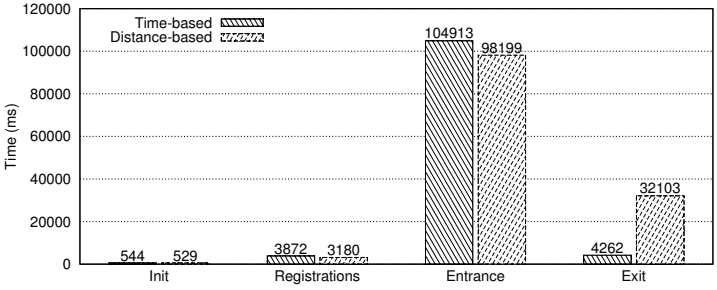
Figure 12.2: Total time expended for each protocol version over both smartphones.

This first analysis is useful to corroborate that the distance-based AFC solution requires more time to execute than time-based AFC. However, we need to extract more information from performance measures to distinguish exactly which are the most expensive processes.

Figures 12.3 and 12.4 expose the execution times for each scheme phase on both smartphones, depending on whether precomputation is enabled or not. To simplify



(a) Time elapsed running user application on Desire without precomputation.



(b) Time elapsed running user application on Wildfire without precomputation.

Figure 12.3: Time to run each scheme phase considering precomputation disabled.

the analysis, we have grouped similar processes related to implementation requirements (not directly related to the scheme definition but required by the implementation, such as starting the user application, loading parameters from storage, etc.) and protocols in the x-axis. The explanation of each phase is as follows:

- **Init**. This phase belongs the client application deployment time and also the time needed to request the public parameters α, p and q to the group manager ($\mathcal{T}_G$).

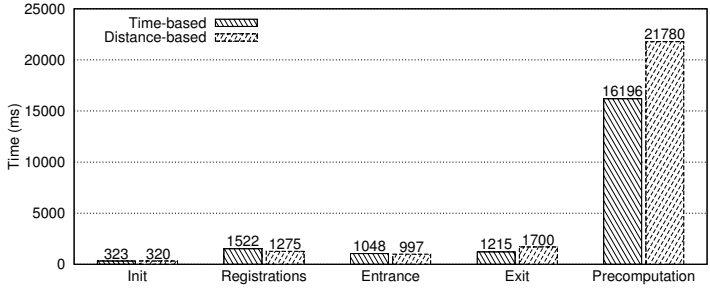
- **Registrations**. It covers both `URegistrationMG` and `URegistrationMP` protocols in which user registers herself to $\mathcal{T}_G$ and $\mathcal{T}_P$, resp. Therefore, it is the required time by a user to complete the system registration in order to be allowed to use services AFC system manages.

- **Entrance**. It covers the time required by the user to execute the `SysEntrace`

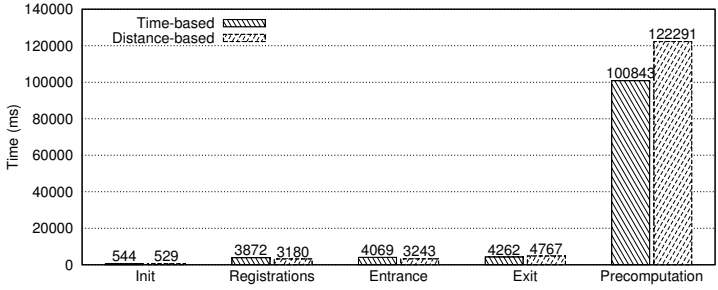
protocol to receive the entrance ticket. It includes all computing and networking load, also computation of group signature parameters.

- **Exit**. It is the time needed by the user to run the `SysExit` protocol that allow her to leave the system after the receipt of the exit ticket. As before, it covers all computing and network load.

- **Precomputation**. It represents the accumulated precomputation time along all protocols. In time-based AFC, precomputation is only performed before the `SysEntrance` protocol, while in distance-based AFC, precomputation is applied before both `SysEntrance` and `SysExit` protocols. Note that this phase only appears in Figure 12.4.



(a) Time elapsed running user application on Desire with precomputation.



(b) Time elapsed running user application on Wildifre with precomputation.

Figure 12.4: Time to run each scheme phases considering precomputation enabled.

On one hand, Figure 12.3 shows that in time-based AFC, the entrance phase is the bottleneck of the system. In distance-based scheme, the entrance phase consumes

less time than in time-based scheme because an encryption has been erased from the specification. Regarding to the exit phase, as a result of the security improvement (adding another group signature), Figure 12.3 proves that in fact this phase requires more time to complete.

On the other hand, Figure 12.4 represents the same classification of phases as before. However, the last bar has been added to Figure 12.4 to represent precomputable time before to execute `SysEntrance` and `SysExit` protocols. So, it proves whether precomputation is enabled, it saves a lot of time when users run protocols at checking points. Comparing Figures 12.4-12.3, we can see how precomputation consumes the most of the time. For example, for Wildfire, the entrance consumes about 100 seconds in case precomputation is not applied. Instead, if precomputation is enabled, the time is reduced significantly to about only 4.0 seconds. The same happens whether the exit phase is analyzed using Wildfire and distance-based AFC, because this step spends about 32 seconds but with considering precomputations, the time is reduced to only 4.7 seconds.

Next, we show a detailed study of consumed time and how we can improve the application performance for both smartphones. So, we have analyzed in detail the required time to run each phase to analyze where the application requires more computation power. In Figures 12.5(a) and 12.5(b), we have organized again the protocol phases in the x-axis as following:

- **Load Pairing**. It represents time needed by the user application to load all data required to perform pairing operations, e.g. loading elliptic curve parameters from a file, preparing Java objects, etc.

- **Prepare RSA**. It depicts time to load the RSA key from the secure key store from the storage.

- **Req. Public Params.**. It covers the time required by the user application to request to the group manager ($\mathcal{T}_G$) the public parameters α, p, q that she needs in the next step.

- **URegistrationMG**. It is the time required to complete the `URegistrationMG` protocol with the group manager ($\mathcal{T}_G$).

- **URegistrationMP**. This time covers the `URegistrationMP` protocol in which the user registers herself with the payment manager ($\mathcal{T}_P$).

- **Prepare SysEntrance**. It refers to the time to precompute some elements (group signature parameters, as stated in Chapter 9) before to execute the `SysEntrance` protocol. It applies to both time-based and distance-based AFC schemes.

- **SysEntrance**. It is the time spent to run the `SysEntrance` protocol.
- **Prepare SysExit**. It is the time needed to perform some precomputations related to the group signature scheme before to execute the `SysExit` protocol. It applies only to the distance-based AFC because the time-based AFC scheme does not requires a second and linkable group signature.
- **SysExit**. Finally, it is the required time to execute the `SysExit` protocol.

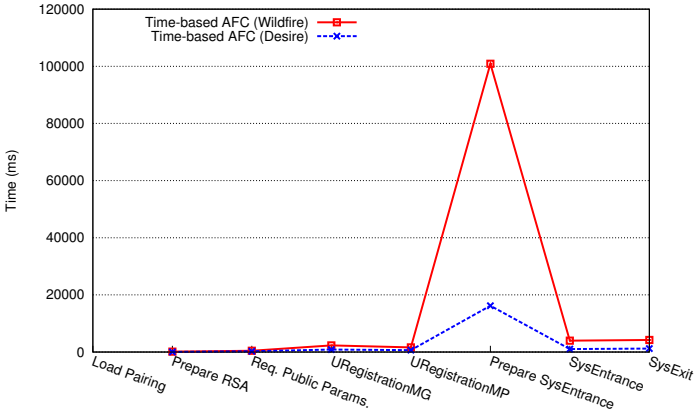
Figure 12.5(a) shows the time line flow of each phase step by step for the time-based AFC scheme. As already state before, Figure 12.5(a) proves that time is mainly consumed to compute the `SysEntrance` protocol. Despite differences about computing features between both smartphones, Wildfire is only clearly slower than Desire in the task of precomputing parameters from the group signature before the `SysEntrance` protocol. So, if we use precomputation before to execute the `SysEntrance` protocol, the performance of the AFC scheme is similar in both devices and it only opens an important gap in phases where it needs more computation power.

Figure 12.5(b) gives similar information as previous one about the case of distance-based AFC. The difference appears with the additional complexity due to the computation of a second group signature. As before, in case the precomputation is enabled also in distance-based AFC, performance obtained by both devices is similar. Thus, the important result is that the `SysExit` protocol is able to finish after 1.7 seconds on Desire and it only needs about 3 more seconds in case Wildfire is used (about 4.7 seconds).

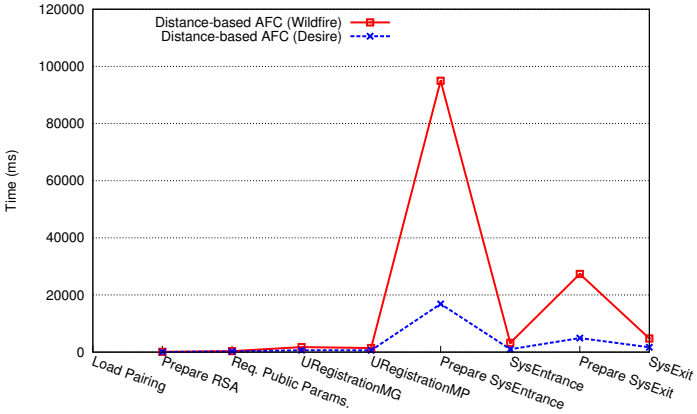
12.3.3 Size of Tickets and Exchanged Messages

Aside from computing performance, the size of tickets and messages exchanged is another important concern to take into account for a ticketing system. One of the functional requirements of tickets is that they must minimize as much as possible the data to transfer and store. Therefore, Figure 12.6 represents the length of all messages exchanged during `SysEntrance` and `SysExit` protocols, including entrance ($t_{out}{}^{*}$) and exit ($t_{out}{}^{*}$) tickets. Moreover, Figure 12.6 also shows differences between time-based and distance-based AFC schemes.

In reference to the size of tickets, the entrance ticket is larger than the exit ticket due to the fact that the former includes more information than the latter. Thus, the entrance ticket conveys a group signature and the group public key which is used by the entrance provider to check the group signature made by $\mathscr{U}$. Instead, the exit ticket only contains its serial number, the exit station, the corresponding fare and a short informative string. Note that entrance ticket is slightly shorter for the distance-based AFC than for the time-based AFC, because the former does not includes a



(a) Time flow for time-based AFC and smartphone.



(b) Time flow for distance-based AFC and smartphone.

Figure 12.5: Time flow for both AFC schemes and smartphones.

hash (hash of k) but adds a validity time and the direction as a simple 0/1 integer depending on the direction (it can be also dropped from the exit ticket in case stations are not separated by direction, as stated in §5.7). However, the exit ticket remains without any changes. Therefore, tickets in any AFC version are short (bellow 3Kb for $t_{in}{}^{*}$ and about 0.6Kb for $t_{out}{}^{*}$), so users with limited storage can continue to store them without problems and providers do not need huge amounts of storage to store issued tickets during some required time range.

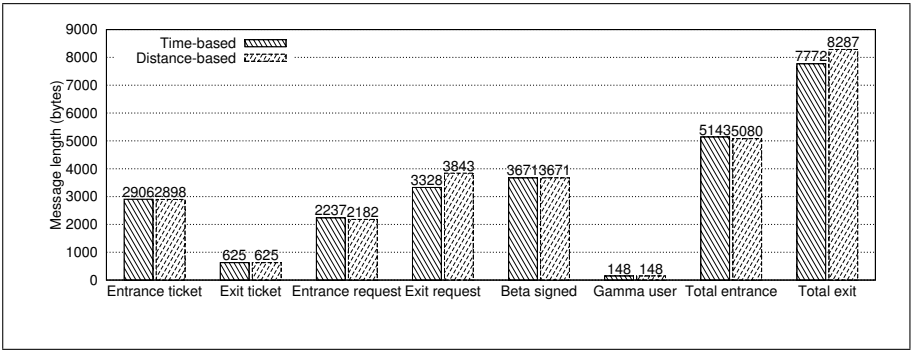


Figure 12.6: Size of tickets and exchanged messages during `SysEntrance` and `SysExit` protocols (bytes).

Regarding to the messages exchanged during protocols, the entrance requests are similar in both AFC schemes (the difference is that in distance-based AFC it is not included the aforementioned hash of k). Instead, the exit request sent by $\mathscr{U}$ during the `SysExit` protocol increases its size in case of the distance-based scheme due to the fact that it compromises an additional group signature. Note that the first message of `SysExit` protocol does not need to include the corresponding group public key, as it is already included in the entrance ticket (also included in message). The remaining two messages maintain their size with independence of the protocol.

Summarizing, regardless whether time-based or distance-based AFC are used to deploy them in a real scenario, the size of messages exchanged during entrance and exit procedures are so short. Besides, the security enhancement to avoid the colluding attack described in §5.6.1 for the distance-based AFC does not impact so much on the amount of exchanged data during `SysEntrace` and `SysExit` protocols. So, it follows that the enhanced distance-based scheme is not penalized in performance, so it could be used in all scenarios if it is necessary.

12.4 Conclusions

After we have proved in §7.3 both time-based and distance-based AFC schemes described in Chapter 5 are secure by means of a security analysis, the final step is to prove that these schemes are also suitable to be used on current mobile devices. Thus, in this Chapter we have successfully developed both AFC schemes using the Java language. Because described AFC schemes make use of group signatures (in particular the BBS group signature scheme), we also have used the knowledge acquired in Chapter 9 and the resulting Java library to deal with the BBS group signature. It has been also required to develop code modification to the BBS Java library to

achieve the objective of linkability of two signatures. As a result, we have conducted an extensive performance analysis of both AFC schemes. This time, the main concern has been the computing time for users and storage requirements for both users and providers. Because of this, we have used the type D pairing to compute operations related to group signature. As stated in Chapter 9, this pairing type presents a good trade-off between storage and computing requirements as well as provides it provides a good performance in both mobile devices and servers.

Regarding to the performance results, we can conclude that the supposed gap between the time-based and distance-based AFC schemes due to the additional computing requirements of the latter scheme are not as large as we have expected in the beginning. Besides, we have proved that the achieved performance by both schemes on Desire and Wildfire smartphones is also manageable on condition that always it has to be considered mandatory the process of precomputing parameters related to the group signature scheme before to run `SysEntrance` and `SysExit` protocols. This way, the principal bottleneck of this implementation has been successfully resolved, making these AFC schemes suitable to be used by users even they do not dispose of powerful smartphones. As a result, the process of issue and validate tickets at checking points (entrance and exit stations) is very fast, even better than traditional paper-based tickets. This suggest that in the near future, when even more powerful mobile devices appear in the market, the protocol performance will be even better. Finally, due to the little difference between time-based and distance-based AFC schemes, we can be confided to say that the latter scheme could be used in all scenarios with only a little performance penalty compared to the time-based AFC proposal.

Part V

Concluding Remarks

CHAPTER 13

CONCLUSIONS AND FUTURE WORK

Along the development of this dissertation, the main objective stated in the Introduction has been achieved: to propose new privacy-protecting schemes and to prove their practicability for e-commerce by means of using mobile devices.

Thus, we have studied previous proposals detecting some unresolved drawbacks. Then, using it as the starting point, we have designed new privacy-protecting schemes that protect the privacy of customers and achieve high levels of security for merchants and providers to detect and even avoid malicious customers committing illicit actions to obtain unfair benefits.

All schemes have been analyzed in order to demonstrate that they accomplish the security requirements stated by their corresponding scenarios and security models. Besides, a particularly interesting task from our point of view has been the implementation and performance evaluation of all proposed schemes. It was indeed a time-consuming task because it is often not easy to implement cryptographic schemes, particularly when they are related to e-commerce. By means of testing them, we have also obtained interesting results to prove that our proposals are actually efficient, lightweight, scalable and suitable to be used in a real scenario by current mobile devices.

Let us review in more detail every single contribution belonging to this dissertation to emphasize the fact that all initial objectives have been successfully achieved.

13.1 *μEasyPay*: Contribution to Micropayments

In this dissertation *μEasyPay* has been proposed, a micropayment scheme suitable to be used to pay merchants for low value items. We have described the security model and the general scenario for micropayments, clarifying and formalizing some security requirements. Then, we have presented the specifications of *μEasyPay* which is based on hash chains and the use of a partially blind signature. Customers withdraw a coin composed by a configurable and pre-agreed number of microcoupons. Then, customers can use these microcoupons to pay merchants without having to reveal any information about their real identity. Depending on the concrete service, we propose two slightly different versions of *μEasyPay*. The first one can be used in scenarios where the limited and controlled risk of losing a single microcoupon is assumed in exchange for a better performance (applicable to services such as video streaming or other real-time services). Instead, in case parties do not agree to this minimal risk, the second version can be used because the risk of losing a microcoupon is even discarded. As proved by formal security analysis, *μEasyPay* protects the anonymity of customers, as well as they cannot be traced by merchants. Regarding to merchants, they are also protected from malicious behavior from customers, as the scheme detects and avoids the microcoupon reuse and forging attempts. In addition, we have conducted a formal analysis of fairness using CPN methodology, verifying both *μEasyPay* versions.

Aside from the security, we have also presented a functional implementation of *μEasyPay* on the Android Platform. We have conducted a performance analysis of protocols run by customers by means of their mobile devices. As a result, we have proved that *μEasyPay* fits all efficiency requirements stated by the functional model, because processes of both withdrawal and spending microcoupons are actually very fast.

Future Work

Future work will be focused on trying to search some method to substitute the use of hash chains to also consider the unlinkability of microcoupons within the same micropayment coin. However, it is probable that transactional costs may suffer a growth, so it could be an interesting research try to mitigate them.

13.2 $\mathscr{MC}-2\mathscr{D}$: Contribution to Electronic Multicoupons

A contribution of this dissertation to the electronic multicoupons solutions has been described. $\mathscr{MC}-2\mathscr{D}$ is a scheme for multicoupons suitable for multi-merchant scenarios. We prove that it enhances security and efficiency as regards the previous

solution that deals with this scenario. This solution provides generic security requirements as well as new mechanisms to increase the trust of customers and merchants attracting them to the multicoupon system. On the one hand, customers achieve a high degree of privacy (anonymity and unlinkability) and they are also protected from misbehaving merchants who can be forced to revoke their affiliation to the multicoupon system. This disaffiliation, whether it is forced due to a merchant misbehavior or by his own decision, does not cause security problems since the system does not share any kind of sensitive data with merchants. In fact, our proposal is the first one to achieve this property. On the other hand, merchants are protected from malicious customers trying either to reuse or to forge coupons, because the solution provides mechanisms to detect and avoid those misbehaviors. In case of misbehavior, $\mathcal{MC}-2\mathcal{D}$ can revoke the anonymity of malicious customers to provide their identities to proper authorities. This way, as we provide measures to protect both customers and merchants from malicious behavior, this proposal could improve the trust in this kind of systems, attracting more attention to them.

A complexity analysis has been conducted, demonstrating that $\mathcal{MC}-2\mathcal{D}$ outperforms the previous proposal, because it is based on lightweight cryptographic operations and it also allows redeeming and claiming multiple coupons during the same protocol run. Moreover, a customer can use different coupons of the same multicoupon at different merchants without requiring any agreement and any shared data among the different merchants. It enhances the flexibility of our proposal together with the fact that the customer and the issuer can agree on the value and the number of coupons of each multicoupon.

Besides the complexity analysis, we have also presented a complete implementation of $\mathcal{MC}-2\mathcal{D}$ multicoupon scheme. Using this implementation, we have conducted a performance comparison with a previous multicoupon proposal considering similar test conditions using a conventional laptop. It proves that $\mathcal{MC}-2\mathcal{D}$ is clearly faster than previous solution both issuing and redeeming coupons. Moreover, increasing the number of coupons issued or redeemed does not penalize the performance of both procedures, as opposed to the previous proposal.

In addition, in the same way as the previous Chapter, we have also presented a full and practical implementation on real Android smartphones, taking into account not only computing performance but also distinguishing the time to encode and decode messages together with time due to communications. Thus, entities running on servers (merchant, issuer and group manager) have been deployed in a real cloud service. This way, we have obtained a complete and functional implementation and performance results have proved how efficient and scalable $\mathcal{MC}-2\mathcal{D}$ is.

Future Work

Further work will be focused on trying to design a new multicoupon solution to allow customers to split and share coupons in a untrusted scenario without involving the issuer in this process. It should consider additional security measures to avoid false accusations and further means to control the reuse of coupons and who actually commits fraud. It will be also interesting to implement the scheme to use NFC in case Android allows using it without restrictions.

13.3 AFC: Contribution to Automatic Fare Collection Systems

AFC contribution has been focused on improving a previous solution in which a distance-based AFC scheme was presented. First, we have provided a consistent and extended security model with a description of the scenario together with a formal description of required security properties. Then, we have generalized the previous proposal to be applied to a general AFC service and we have described the minor differences to fit in a time or distance-based scenario. Previous solution applied to distance-based fare scenario presented a vulnerability allowing colluding users (in some circumstances) to achieve a fare less than the corresponding value according to the given utilization of the service. To avoid this security flaw, we have enhanced the original scheme in a collaboration with researchers from *Universitat Rovira i Virgili (URV)*. The solution meets all security requirements as has been proved by a security analysis.

In the same way as in previous Chapters, we have conducted a practical implementation using Android smartphones. Besides, we have evaluated the performance of both versions (time-based and improved distance-based AFC) in order to know how the security improvement has affected the final performance. As a result, we can state that both time-based and distance-based AFC schemes are actually suitable to be used by current smartphones, even in-car systems in case distance-based solution is used to manage highways tolls. Both schemes have been proved fast (even faster than traditional paper-based ticketing systems) in the task of issuing and validating tickets, which is the main functional requirement. Moreover, we have analyzed storage and communication costs of protocols, and we conclude that schemes also preserve storage of providers.

Future Work

As a future work, if the Android platform allows full access to all NFC features, it will be very interesting to modify the implementation to use the NFC technology to provide the communication channel between user application and providers. This

way, entrance and exit procedures could be started at the moment they are close to entrance and exit stations without further interactions.

It could also be of particular interest to study how to propose a similar solution to automatically charge users in open parking lots, such as cars parked in streets.

13.4 Automatic Formal Analysis of Secure Protocols with Colored Petri Nets

Formal verification of protocols and schemes is often a tedious task. To facilitate and automatize this process, some methodologies and languages have been proposed in the literature. In this dissertation, some of them have been analyzed but only one seems to be able to provide a strong verification framework to analyze security properties from protocols and schemes: the Colored Petri Net (CPN) language. It integrates the mathematical background and formalism from Petri nets, adding some interesting features. Moreover, CPN is supported by a graphical application called CPNTools which covers all tasks related to CPN: modeling protocols, checking correctness and automatic verification.

Based on CPN and some previous works, we have analyzed the applicability of CPNs to automatically provide a formal analysis of some security properties, such as fairness and NR. We have described a CPN methodology to model secure protocols and to analyze them. As an example, we have chosen the FPH contract signing scheme. We have modeled with CPN the aforementioned scheme and we have analyzed its fairness property. As a result, and particularly satisfactory, the proposed model has allowed some undiscovered situations in which parties may acquire a unfair advantage during the contract signing to be addressed. Then, after a careful analysis, we have found why these situations arise and we have proposed guidelines to improve the FPH scheme.

Although CPN method has been proved useful to automatically analyze some security properties, the method applicability will depend on the complexity of the scheme to be analyzed. Therefore, we suggest that before starting to model a protocol, it is be essential to conduct an initial analysis to determine whether it actually can be modeled.

Besides, knowledge acquired with this analysis has been useful to apply the same CPN methodology to formally analyze the fairness property of *μEasyPay* micropayment presented in Chapter 3.

Future Work

Future work in this research line could be the applicability analysis of the CPN method to automatically verify other secure properties, such as TTP verificability

(to prove whether a TTP behaves correctly).

13.5 Implementation and Performance Evaluation of Group Signatures

Both $\mathscr{MC}-2\mathscr{D}$ and AFC schemes consider the utilization of group signatures to provide revocable anonymity for customers. One of the most referred group signatures in the literature is the BBS scheme, so we have selected this one as the cryptographic tool to design our proposals. So, it has been necessary to implement the BBS scheme to be used afterward by the other implementations. This contribution is the first successful and complete implementation (to the best of our knowledge) with Java language of the pairing-based BBS group signature scheme.

Besides, we have also contributed with an extensive and complete performance analysis of BBS group signature in order to know what its behavior is when executed on mobile devices. Besides, we have also tested performance with different types of pairings and elliptic curves considering a common security strength. In addition, with the purpose of providing a comparison between a pairing-based group signature and a non-pairing based group signature, we have also implemented another group signature which relies on different mathematical assumptions. As a result, we have presented a complete comparison between both schemes, proving some facts about computing and storage efficiency. Moreover, we have also detected some interesting facts about different performance measures of some pairing operations. Thus, both implementation and performance analysis provides valuable information for further works with applications using group signatures and in general, for cryptographic tools which take advantage of pairings features.

Future Work

It could be interesting to publish and distribute the developed library to allow other researchers to use it as a developing box for their projects.

13.6 Implementation and Performance Evaluation of Privacy-protecting Schemes

Although in the above conclusions related to each contribution we have already highlighted the fact of achieving successful, practical and efficient implementations, we want to emphasize it once again because it was described as one of the main objectives of this dissertation.

Summarizing, we have proposed new privacy-protecting schemes for the electronic commerce field. Along this document, we have covered all the steps required to design and prove the benefits of our proposals compared to the previous related works. In fact, we have not only presented the solutions and proved their security, but we have also demonstrated how our solutions improve those previous works in terms of security, practicability and performance, as well as how our solutions can be deployed in a real scenario with current mobile devices. Thus, all the objectives of this dissertation have been successfully achieved.

CHAPTER 14

LIST OF PUBLICATIONS SUPPORTING THE DISSERTATION

The following list gathers all publications that support this dissertation.

International Conferences

M. Payeras-Capellà, M. Mut-Puigserver, A.P. Isern-Deyà, J.L. Ferrer-Gomila and L. Huguet-Rotger. *"Formal Analysis of FPH Contract Signing Protocol Using Colored Petri Nets"*. Proceedings of 5th International ICST Security and Privacy in Communications (SecureComm'09), 2009, Springer-Verlag, Athens, Greece. *(Core A)*. It appears as *"Formal Analysis of FPH Contract Signing Protocol Using Colored Petri Nets"* in Security and Privacy in Communication Networks, 5th International ICST Conference, SecureComm 2009, Revised Selected Papers. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 19, 2009, p.p. 101-120. DOI: 10.1007/978-3-642-05284-2_7.

A.P. Isern-Deyà, M. Payeras-Capellà, M. Mut-Puigserver, J.L. Ferrer-Gomila. *"Anonymous, secure and fair protocol to access location-based services subject to payment*". Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia (MoMM'10), 2010, p.p. 436-441, ACM, Paris, France. DOI: 10.1145/1971519.1971594. *(Core B)*.

A. Paniza-Fullana, M. Payeras-Capellà, M. Mut-Puigserver, A.P. Isern-Deyà. *Reflections on privacy in new location based services in social networks*. Proceedings of the International IADIS e-commerce Conference 2011, 2011. Rome, Italy.

A.P. Isern-Deyà, M. Payeras-Capellà, M. Mut-Puigserver, J.L. Ferrer-Gomila. "*Untraceable, anonymous and fair micropayment scheme*". Proceedings of the 9th International Conference on Advances in Mobile

Computing and Multimedia (MoMM'11), 2011, p.p. 42-49, ACM. Ho Chi Minh City, Vietnam. DOI: 10.1145/2095697.2095707. *(Core B)*.

A.P. Isern-Deyà, M.F. Hinerajos, J.L. Ferrer-Gomila and M. Payeras-Capellà. *"Secure Multicoupon Solution for Multi-Merchant Scenarios"*. In proceedings of IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom'11), 2011, p.p. 655-663. Changsha, China. DOI: 10.1109/TrustCom.2011.84. *(Core A)*.

A. P. Isern-Deyà, L. Huguet-Rotger, M. Payeras-Capellà and M. Mut-Puigserver. *"Esquema de Micropago Anónimo, Equitativo y no Rastreable: Aplicación a los Servicios LBS"*. VI Congreso Iberoamericano de Seguridad Informática (CIBSI'11). 2011. Bucaramanga, Colombia. *Selected paper to publish in IEEE Transactions Latin America journal*.

M. Mut-Puigserver, A.P. Isern-Deyà, A. Paniza-Fullana, M. Payeras-Capellà, M. *Privacy in New Touristic Location-Based Services*. Proceedings of the International IADIS E-society Conference, 2012. Berlin, Germany.

A.P. Isern-Deyà, M. Payeras-Capellà, M. Mut-Puigserver, J.L. Ferrer-Gomila and L. Huguet-Rotger. *"Micropayment scheme implementation on the Android platform with performance evaluation"*. In proceedings of the 10th International Conference on Advances in Mobile Computing and Multimedia (MoMM'12), 2012, p.p. 64-73. Bali, Indonesia. *(Core B)* DOI: 10.1145/2428955.2428974. *It was selected to receive the best paper award*.

International Journals

A.P. Isern-Deyà, A. Vives-Guasch, M. Mut-Puigserver, M. Payeras-Capellà and J. Castellà-Roca. "A Secure Automatic Fare Collection System for Time-Based or Distance-Based Services with Revocable Anonymity for Users". *The Computer Journal*, 2012. DOI: 10.1093/comjnl/bxs033. *(JCR Impact Factor: 0.755; SJR Impact Factor: 0.84)*.

A.P. Isern-Deyà, L. Huguet-Rotger, M. Payeras-Capellà and M. Mut-Puigserver. *"Anonymous, Fair and Untraceable Micropayment Scheme: Application to LBS*". IEEE Transactions Latin America, Vol. 10, Num. 3, p.p. 1774-1784. 2012. DOI: 10.1109/TLA.2012.6222584. *(JCR Impact Factor: 0.218)*.
A.P. Isern-Deyà, M. Payeras-Capellà, M. Mut-Puigserver, J.L. Ferrer-Gomila. *"Anonymous and Fair Micropayment Scheme with Protection against Coupon Theft"*. In International Journal of Adaptative, Resilient and Autonomic Systems (IJARAS), vol 4, num. 2, 2013. DOI: 10.4018/jaras.2013040103.

A.P. Isern-Deyà, M. Payeras-Capellà, M. Mut-Puigserver, J.L. Ferrer-Gomila and L. Huguet-Rotger. *Micropayment proposal with formal verification using coloured Petri nets and performance analysis on the Android Platform*. International Journal on Business Intelligence and Data Mining (IJBIDM), vol. 8, num. 1, 2013. *Selected paper from MoMM'12 to be expanded. (SJR Impact Factor: 0.23)*

M.F. Hinarejos, A.P. Isern-Deyà, J.L. Ferrer-Gomila and M. Payeras-Capellà. *"MC-2D: An Efficient and Scalable Multicoupon Scheme"*. The Computer Journal. *Accepted, September 2013. (JCR Impact Factor: 0.755; SJR Impact Factor: 0.84)*.

A.P. Isern-Deyà, M.F. Hinarejos, J.L Ferrer-Gomila. *"Considerations on the Deployment and Performance Evaluation of Electronic Multicoupon System"*. Electronic Commerce Research. *Submitted and currently in revision. (JCR Impact Factor: 1.553; SJR Impact Factor: 0.64)*.

A.P. Isern-Deyà, L. Huguet-Rotger, M. Payeras-Capellà and M. Mut-Puigserver. *"On the Practicability of Using Group Signatures on Mobile Devices: Implementation and Performance Analysis on the Android*

Platform". Security and Communication Networks. *Submitted and currently in revision. (JCR Impact Factor: 0.311; SJR Impact Factor: 0.24).*

Book Chapter

A.P. Isern-Deyà, M. Payeras-Capellà, M. Mut-Puigserver, J.L. Ferrer-Gomila. *"Anonymous, Secure and Fair Micropayment System to Access Location-Based Services"*. Trustworthy Ubiquitous Computing. Atlantis Ambient and Pervasive Intelligence vol. 6, p.p. 227-247, 2012. Atlantis press with Springer-Verlag. DOI: 10.2991/978-94-91216-71-8_11. *Selected paper from MoMM'10 to appear in this book chapter.*

National Conferences

A.P. Isern-Deyà, M. Payeras-Capellà, M. Mut-Puigserver, J.L. Ferrer Gomila, L. Huguet-Rotger. *Verificación Formal Automatizada del Protocolo de Firma de Contratos FPH Usando Colored Petri Nets*. Actas de la VIII Jornadas de Ingeniería Telemática (JITEL'09), 2009, pp.54-61. Cartagena, Spain.

A.P. Isern-Deyà, M. Payeras-Capellà, M. Mut-Puigserver, J.L. Ferrer Gomila. *Protocolo Anónimo y Equitativo de Acceso a Servicios de Pago Basados en Localización*. Actas de la IX Jornadas de Ingenieria Telemática (JITEL'10), 2010. Valladolid, Spain.

A.P. Isern-Deyà, A. Vives-Guasch, M. Payeras-Capellà, M. Mut-Puigserver, J. Castellà-Roca. *Sistema de Tarificación Automático con anonimato revocable: evaluación del rendimiento*. Actas de las X Jornadas de ingeniería Telemática (JITEL'11), 2011. Santander, Spain.

A.P. Isern-Deyà, M. Francisca Hinarejos, J.L. Ferrer-Gomila, M. Payeras-Capellà, Carlos Gañán, José Luis Muñoz, Jordi Forné, Oscar Esparza *Un Esquema de Pago Seguro mediante Multicupones para Escenarios Multi-Comerciante*. XII Reunión Española sobre Criptología y Seguridad de la Información (RECSI'12), 2012. San Sebastián, Spain.

A.P. Isern-Deyà. *Implementación de las firmas de grupo basadas en las 'pairing functions'*. Congreso de la Real Sociedad Matemática Española (RSME), 2013. Santiago de Compostela, Spain.

BIBLIOGRAPHY

[1] International Telecommunication Union, "ICT Facts and Figures, The World in 2013," PDF Report, downloaded on July, 2013, March 2013, http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2013.pdf. 1

[2] Mobithinking.com, "Global mobile statistics 2013," Online report, last checked in July, 2013, May 2013, http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats. 1

[3] Forrester, "US Online Retail Forecast, 2012 To 2017," Online forecast, last checked in July, 2013, March 2013, http://www.forrester.com/US+Online+Retail+Forecast+2012+To+2017/fulltext/-/E-RES93281. 1

[4] ——, "European Online Retail Forecast, 2012 To 2017," Online forecast, last checked in July, 2013, March 2013, http://www.forrester.com/European+Online+Retail+Forecast+2012+To+2017/fulltext/-/E-RES93341. 1

[5] Techcrunch, "Forrester 2012-2017 e-commerce forecast analysis," Website, last checked in July, 2013, March 2013, http://techcrunch.com/2013/03/13/forrester-2012-2017-ecommerce-forecast/. 1

[6] B. Gellman and L. Poitras, "U.S., British intelligence mining data from nine U.S. Internet companies in broad secret program," Website, last checked in July, 2013, June 2013, goo.gl/wRXMN (Washington Post website). 1

[7] K. Johnson, S. Martin, J. O'Donnell, and M. Winter, "NSA taps data from 9 major Net firms," Website, last checked in July, 2013, June 2013, http://www.usatoday.com/story/news/2013/06/06/nsa-surveillance-internet-companies/2398345/. 1

[8] P. McCole, E. Ramsey, and J. Williams, "Trust considerations on attitudes towards online purchasing: The moderating effect of privacy and security concerns," *Journal of Business Research*, vol. 63, no. 9-10, pp. 1018 – 1024,

2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0148296309001933 1

[9] G. J. Udo, "Privacy and security concerns as major barriers for e-commerce: a survey study," *Information Management & Computer Security*, vol. 9, no. 4, pp. 165 – 174, 2001. 1

[10] B. J. Corbitt, T. Thanasankit, and H. Yi, "Trust and e-commerce: a study of consumer perceptions," *Electronic Commerce Research and Applications*, vol. 2, no. 3, pp. 203 – 215, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1567422303000243 1

[11] C. Jutla and M. Yung, "Paytree: amortized signature for flexible micropayments," in *2nd USENIX Workshop on electronic commerce*, 1996, pp. 213–221. 1.2.1, 1.3, 3.1, 3.3, 3.3, 3.1

[12] Q. Nguyen, "Multi-Dimensional Hash Chains and Application to Micropayment Schemes," in *Coding and Cryptography*, ser. Lecture Notes in Computer Science, 2006, vol. 3969, pp. 218–228. 1.2.1, 1.3, 3.1, 3.3, 3.3, 3.1

[13] R. Rivest and A. Shamir, "PayWord and MicroMint: Two simple micropayment schemes," in *Security Protocols*, ser. Lecture Notes in Computer Science, 1997, vol. 1189, pp. 69–87. 1.2.1, 1.3, 3.1, 3.2.2, 3.3, 3.3, 3.1, 10.3.2

[14] F. Wang, W. Dong, and Y. Ji, "A new credit based micropayment scheme," in *IEEE International Conference on e-Business Engineering, ICEBE'08.*, 2008, pp. 596 –601. 1.2.1, 3.1, 3.3, 3.3, 3.1

[15] X. Zhao, Y. Lv, and W. He, "A novel micropayment scheme with complete anonymity," in *Fifth International Conference on Information Assurance and Security, IAS'09*, 2009, pp. 638–642. 1.2.1, 3.1, 3.3, 3.3, 3.1

[16] Edenred, "An international company for booklet coupons of restaurants," Website, last checked in July, 2013, available: http://www.edenred.com. 1.2.2, 4.1

[17] Gourmet, "An international company for booklet coupons of restaurants," Website, last checked in July, 2013, available: http://www.cheque-dejeuner.com/. 1.2.2, 4.1

[18] Bancotel, "An international company for booklet coupons of hotel rooms," Website, last checked in July, 2013, available: http://www.bancotel.es/. 1.2.2, 4.1

[19] C. Blundo, S. Cimato, and A. De Bonis, "Secure e-coupons," *Electronic Commerce Research*, vol. 5, pp. 117–139, January 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1029323.1029330 1.2.2, 1.3, 4.1

[20] L. Chen, M. Enzmann, A.-R. Sadeghi, M. Schneider, and M. Steiner, "A Privacy-Protecting Coupon System," in *Proceedings of the 9th International Financial Cryptography and Data Security Conference (FC2005)*, ser. Lecture Notes in Computer Science, vol. 3570. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 578–578. 1.2.2, 1.3, 4.1, 4.2.1, 4.2.9, 4.3, 4.3, 4.1

[21] L. Nguyen, "Privacy-protecting coupon system revisited," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 2006, vol. 4107, pp. 266–280. 1.2.2, 1.3, 4.1, 4.2.1, 4.3, 4.3, 4.1

[22] S. Canard, A. Gouget, and E. Hufschmitt, "A handy multi-coupon system," in *Applied Cryptography and Network Security*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 2006, vol. 3989, pp. 66–81. 1.2.2, 1.3, 4.1, 4.2.1, 4.2.2, 4.3, 4.3, 4.1

[23] A. N. Escalante, H. Löhr, and A.-R. Sadeghi, "A non-sequential unsplittable privacy-protecting multi-coupon scheme," *GI Jahrestagung*, vol. 2, pp. 184–188, 2007. 1.2.2, 1.3, 4.1, 4.2.1, 4.2.9, 4.3, 4.1, 4.3

[24] L. Chen, B. A. N. Escalante, H. Löhr, M. Manulis, and A.-R. Sadeghi, "A privacy-protecting multi-coupon scheme with stronger protection against splitting," in *Proceedings of the 11th International Conference on Financial Cryptography and 1st International Conference on Usable Security*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 2007, vol. 4886, pp. 29–44. 1.2.2, 1.3, 4.1, 4.2.1, 4.2.2, 4.2.4, 4.3, 4.1, 4.3

[25] F. Armknecht, B. A. N. Escalante, H. Löhr, M. Manulis, and A.-R. Sadeghi, "Secure multi-coupons for federated environments: privacy-preserving and customer-friendly," in *Proceedings of the 4th International Conference on Information Security Practice and Experience*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer-Verlag, 2008, vol. 4991, pp. 29–44. [Online]. Available: http://portal.acm.org/citation.cfm?id=1788494.1788497 1.2.2, 1.3, 4.1, 4.2.1, 4.2.2, 4.2.9, 4.3, 4.1, 4.3, 4.4, 4.6, 4.3, 4.6, 11.1, 11.3, 11.1, 11.3

[26] S.-C. Hsueh and J.-M. Chen, "Sharing secure m-coupons for peer-generated targeting via eWOM communications," *Electronic Commerce Research and Applications*, vol. 9, pp. 283–293, July 2010. [Online]. Available: http://dx.doi.org/10.1016/j.elerap.2010.01.002 1.2.2, 4.1

[27] M. Mut-Puigserver, M. M. Payeras-Capellà, J.-L. Ferrer-Gomila, A. Vives-Guasch, and J. Castellà-Roca, "A survey of electronic ticketing applied to transport," *Computers & Security*, vol. 31, no. 8, pp. 925 – 939, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404812001058 1.2.3, 1.3, 5.2.1, 5.2.2, 5.2.8, 5.3

[28] H. Wang, J. Cao, and Y. Zhang, "Ticket-based service access scheme for mobile users," *Aust. Comput. Sci. Commun.*, vol. 24, no. 1, pp. 285–292, 2002. 1.2.3, 5.2.1, 5.3, 5.1

[29] L. Buttyán, T. Holczer, and I. Vajda, "Providing location privacy in automated fare collection systems," in *In Proceedings of the 15th IST Mobile and Wireless Communication Summit, Mykonos, Greece*, June 2006. 1.2.3, 5.2.1, 5.3, 5.1

[30] T. S. Heydt-Benjamin, H.-J. Chae, B. Defend, and K. Fu, "Privacy for public transportation," in *6th Workshop on Privacy Enhancing Technologies (PET 2006)*, 2006, pp. 1–19, lNCS 4258. 1.2.3, 5.2.1, 5.3, 5.1

[31] S.-P. Hong and S. Kang, "Ensuring privacy in smartcard-based payment systems: A case study of public metro transit systems," in *Communications and Multimedia Security*, 2006, pp. 206–215. 1.2.3, 5.2.1, 5.3, 5.1

[32] O. Jorns, O. Jung, and G. Quirchmayr, "A privacy enhancing service architecture for ticket-based mobile applications," in *2nd International Conference on Availability, Reliability and Security*. Vienna, Austria: ARES 2007 - The International Dependability Conference, Apr 2007, pp. 374–383, vol. 24. 1.2.3, 5.2.1, 5.3, 5.1

[33] G. Madlmayr, P. Kleebauer, J. Langer, and J. Scharinger, "Secure communication between web browsers and nfc targets by the example of an e-ticketing system," in *EC-Web '08: Proceedings of the 9th international conference on E-Commerce and Web Technologies*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 1–10. 1.2.3, 5.2.1, 5.3, 5.1

[34] A. Vives-Guasch, J. Castellà-Roca, M. M. Payeras-Capella, and M. Mut-Puigserver, "An electronic and secure automatic fare collection system with revocable anonymity for users," in *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia*, ser. MoMM '10. New York, NY, USA: ACM, 2010, pp. 387–392. [Online]. Available: http://doi.acm.org/10.1145/1971519.1971585 1.2.3, 5.1, 5.4, 5.8

[35] K. Jensen, *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use*, ser. Monographs in Theoretical Computer Science. Springer-Verlag, 1992 - 1997, vol. 1-3. 1.2.4, 6.1, 6.4, 6.4

[36] AVISPA, Website, last checked in July, 2013, http://www.avispa-project.org/. 1.2.4, 6.1, 6.2

[37] SPIN, Website, last checked in July, 2013, http://spinroot.com/spin/whatispin.html. 1.2.4, 6.1, 6.2

[38] P. Katsaros, "A roadmap to electronic payment transaction guarantees and a Colored Petri Net model checking approach," *Information and Software Technology*, vol. 51, no. 2, pp. 235 – 257, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584908000219 1.2.4, 6.1, 6.2, 6.10

[39] P. Sornkhom and Y. Permpoontanalarp, "Security analysis of micali's fair contract signing protocol by using coloured petri nets," *9th ACIS Int. Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp. 329–334, 2008. 1.2.4, 6.1, 6.7, 6.8, 6.8.4, 6.8.5, 6.10

[40] J. Ferrer-Gomila, M. Payeras-Capellà, and L. Huguet-Rotger, "Efficient Optimistic N-Party Contract Signing Protocol," *Information Security Conference. 4th International Conference, ISC'01, LNCS 2200, Springer Verlag*, pp. 394–407, 2001. 1.2.4, 6.1, 6.6.3, 6.8.6, 6.9, 6.10

[41] D. Chaum and E. Van Heyst, "Group signatures," in *Proceedings of the 10th annual international conference on Theory and application of cryptographic techniques*, ser. EUROCRYPT'91. Berlin, Heidelberg: Springer-Verlag, 1991, pp. 257–265. 1.2.5, 2.3

[42] M. Bellare, D. Micciancio, and B. Warinschi, "Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions," in *Advances in Cryptology — EUROCRYPT 2003*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2003, vol. 2656, pp. 644–644. 1.2.5, 2.3.1

[43] M. Bellare, H. Shi, and C. Zhang, "Foundations of group signatures: The case of dynamic groups," in *Topics in Cryptology – CT-RSA 2005*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, vol. 3376, pp. 136–153. 1.2.5, 2.3.1

[44] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A Practical and Provably Secure Coalition-Resistant Group Signature Scheme," in *Advances in Cryptology — CRYPTO 2000*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2000, vol. 1880, pp. 255–270. [Online]. Available: http://dx.doi.org/10.1007/3-540-44598-6_16 1.2.5, 2.3.1, 2.3.3

[45] J. Camenisch and J. Groth, "Group Signatures: Better Efficiency and New Theoretical Aspects," in *Security in Communication Networks*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005, vol. 3352, pp. 120–133. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30598-9_9 1.2.5, 2.3.1

[46] D. Boneh, X. Boyen, and H. Shacham, "Short Group Signatures," in *Advances in Cryptology – CRYPTO 2004*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004, vol. 3152, pp. 227–242. 1.2.5, 2.3.1, 2.3.3, 6, 4.6, 7.2.1

[47] J. Groth, "Fully Anonymous Group Signatures Without Random Oracles," in *Advances in Cryptology – ASIACRYPT 2007*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2007, vol. 4833, pp. 164–180. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-76900-2_10 1.2.5, 2.3.1

[48] B. Libert, T. Peters, and M. Yung, "Scalable Group Signatures with Revocation," in *Advances in Cryptology – EUROCRYPT 2012*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2012, vol. 7237, pp. 609–627. 1.2.5, 2.3.1

[49] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology: Proceedings of Crypto 83*, vol. 82, 1983, pp. 199–203. 2.2.1

[50] D. Pointcheval and J. Stern, "Provably secure blind signature schemes," in *Asiacrypt'96*, ser. Lecture Notes in Computer Science. Springer Verlag Berlin / Heidelberg, 1996, vol. 1163, pp. 152–165. 2.2.1

[51] A. Juels, M. Luby, and R. Ostrovsky, "Security of blind digital signatures," in *Crypto'97*, ser. Lecture Notes in Computer Science. Springer Verlag Berlin / Heidelberg, 1997, vol. 1294, pp. 150–164. 2.2.1

[52] M. Abe and E. Fujisaki, "How to date blind signatures," in *Asiacrypt'96*, ser. Lecture Notes in Computer Science. Springer Verlag Berlin / Heidelberg, 1996, vol. 1666, pp. 244–251. 2.2.2

[53] M. Abe and T. Okamoto, "Provably Secure Partially Blind Signatures," in *CRYPTO 2000*, ser. Lecture Notes in Computer Science. Springer Verlag Berlin / Heidelberg, 2000, vol. 1880, pp. 271–286. 2.2.2

[54] T. Okamoto, "Efficient Blind and Partially Blind Signatures Without Random Oracles," in *TCC 2006*, ser. Lecture Notes in Computer Science. Springer Verlag Berlin / Heidelberg, 2006, vol. 3876, pp. 80–99. 2.2.2, 2.2.3

[55] H.-Y. Chien, J.-K. Jan, and Y.-M. Tseng, "RSA-based Partially Blind Signature with Low Computation," *International Conference on Parallel and Distributed Systems, ICPADS 2001*, pp. 385–389, 2001. 2.2.2, 2.2.3, 4.6, 7.1.1, 7.1.3, 7.2.1, 7.2.2, 10.2.1

[56] F. Zhang, R. Safavi-Naini, and W. Susilo, "Efficient Verifiably Encrypted Signature and Partially Blind Signature form Bilinear Pairings," in *Indocrypt 2003*, ser. Lecture Notes in Computer Science. Springer Verlag Berlin / Heidelberg, 2003, vol. 2904, pp. 191–204. 2.2.2, 2.2.3

[57] D. Boneh and X. Boyen, "Short signatures without random oracles," in *Advances in Cryptology - EUROCRYPT 2004*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3027, pp. 56–73. 2.3.1, 2.3.3

[58] Y. Rong-wei, W. Li-na, M. Xiao-yan, and K. Bo, "A Direct Anonymous Attestation Protocol Based on Hierarchical Group Signature," in *Computational Science and Engineering, 2009. CSE '09. International Conference on*, vol. 2, aug. 2009, pp. 721 –726. 2.3.2, 9.1

[59] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra: a virtual machine-based platform for trusted computing," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 193–206, Oct. 2003. 2.3.2, 9.1

[60] C.-H. Wang and W.-Y. Tsai, "An Anonymous Roaming Protocol Based on Group Signature without Communication with Home Server," in *Proceedings of the Joint Workshop on Information Security*, 2009. 2.3.2, 9.1

[61] A. Fujii, G. Ohtake, G. Hanaoka, and K. Ogawa, "Anonymous Authentication Scheme for Subscription Services," in *Knowledge-Based Intelligent Information and Engineering Systems*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2007, vol. 4694, pp. 975–983. 2.3.2, 9.1

[62] X. Liu, Q.-L. Xu, and J.-Q. Shang, "A public auction scheme based on group signature," in *Proceedings of the 3rd international conference on Information security*, ser. InfoSecu '04. ACM, 2004, pp. 136–142. 2.3.2

[63] C.-C. Lee, P.-F. Ho, and M.-S. Hwang, "A secure e-auction scheme based on group signatures," *Information Systems Frontiers*, vol. 11, pp. 335–343, 2009. 2.3.2

[64] G. Maitland and C. Boyd, "Fair electronic cash based on a group signature scheme," in *Information and Communications Security*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2001, vol. 2229, pp. 461–465. 2.3.2, 9.1

[65] S. Canard and J. Traoré, "On fair e-cash systems based on group signature schemes," in *Information Security and Privacy*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, vol. 2727, pp. 237–248. 2.3.2, 9.1

[66] G. Fuchsbauer, D. Pointcheval, and D. Vergnaud, "Transferable constant-size fair e-cash," in *Cryptology and Network Security*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, vol. 5888, pp. 226–247. 2.3.2, 9.1

[67] N. Asokan, V. Shoup, and M. Waidner, "Asynchronous Protocols for Optimistic Fair Exchange," *IEEE Symposium on Research in Security and Privacy*, pp. 86–99, 1998. 3.1, 3.2.10, 3.2.10, 6.6.2

[68] J. Zhou, R. Deng, and F. Bao, "Some Remarks on a Fair Exchange Protocol," in *Public Key Cryptography*, ser. Lecture Notes on Computer Science, 2000, vol. 1751, pp. 46–57. 3.1, 3.2.10, 3.2.10

[69] I. Papaefstathiou and C. Manifavas, "Evaluation of micropayment transaction costs," *Journal of Electronic Commerce Research*, vol. 5, no. 2, pp. 99–113, 2004. 3.2, 3.2.2

[70] T. Poutanen, H. Hinton, and M. Stumm, "NetCents: A lightweight protocol for secure micropayments," in *USENIX Workshop on Electronic Commerce*, 1998, pp. 25–36. 3.2

[71] C. Schmidt and R. Müller, "A framework for micropayment evaluation," *NETNOMICS*, pp. 187–200, 1999. 3.2

[72] R. Lipton and R. Ostrovsky, "Micro-payments via efficient coin-flipping," in *Financial Cryptography*, ser. Lecture Notes on Computer Science, 1998, vol. 1465, pp. 1–15. 3.2, 3.2.2

[73] S. Kremer, O. Markowitch, and J. Zhou, "An Intensive Survey of Fair Non-Repudiation Protocols," *Computer Communications*, vol. 25, pp. 1606–1621, 2002. 3.2.10, 6.8.6

[74] N. Asokan, M. Shunter, and M. Waidner, "Optimistic Protocols for Fair Exchange," *4th ACM Conference on Computer and Communications Security*, pp. 7–17, 1997. 3.2.10, 6.6.1, 6.6.2

[75] L. Buttyan, "Removing the financial incentive to cheat in micropayment schemes," *Electronics Letters*, vol. 36, no. 2, pp. 132 –133, 2000. 3.3

[76] J. Nan, L. Xiang-dong, Z. Jing-ying, and Y. De-li, "A Mobile Micropayment Protocol Based on Chaos," in *Eighth International Conference on Mobile Business*, 2009, pp. 284–289. 3.3

[77] Y. Zongkai, L. Weimin, and T. Yunmeng, "A new fair micropayment system based on hash chain," in *IEEE International Conference on e-Technology, e-Commerce and e-Service, 2004. EEE'04*, 2004, pp. 139 – 145. 3.3

[78] M. Payeras-Capellà, J. Ferrer-Gomila, and L. Huguet-Rotger, "An efficient anonymous scheme for secure micropayments," in *Web Engineering*, ser. Lecture Notes in Computer Science, 2003, vol. 2722, pp. 227–246. 3.3

[79] A. Esmaeeli and M. Shajari, "MVPayword: Secure and efficient Payword-based micropayment scheme," in *Second International Conference on the Applications of Digital Information and Web Technologies. ICADIWT '09*, 2009, pp. 609 –614. 3.3

[80] Y.-S. Jeong, N. Sun, and S.-H. Lee, "IPTV Micropayment System Based on Hash Chain Using RFID-USB Module," in *IEEE 34th Annual Computer Software and Applications Conference (COMPSAC)*, july 2010, pp. 155 –160. 3.3

[81] J. Hao, J. Zou, and Y. Dai, "A Real-Time Payment Scheme for SIP Service Based on Hash Chain," in *IEEE International Conference on e-Business Engineering, 2008. ICEBE '08*, 2008, pp. 279 –286. 3.3

[82] N. Jiang, X.-d. Liu, J.-y. Zhao, and D.-l. Yang, "A Mobile Micropayment Protocol Based on Chaos," in *Proceedings of the Eighth International Conference on Mobile Business. ICMB 2009*, 2009, pp. 284–289. 3.3

[83] K. Chaudhary, X. Dai, and J. Grundy, "Experiences in Developing a Micropayment System for Peer-to-Peer Networks," *International Journal of Information Technology and Web Engineering*, vol. 5, no. 1, pp. 23–42, 2010. 3.3

[84] K. S. Kadambi, J. Li, and A. H. Karp, "Near-field communication-based secure mobile payment service," in *Proceedings of the 11th International Conference on Electronic Commerce*, ser. ICEC '09. ACM, 2009, pp. 142–151. 3.3

[85] L. Liu, "Privacy and location anonymization in location-based services," *SIGSPATIAL Special, Volume 1, Issue 2 (July 2009), pp. 15-22*, 2009. 3.6.1

[86] B. Gedik and L. Liu, "A Customizable k-Anonymity Model for Protecting Location Privacy," Georgia Institute of Technology, Tech. Rep., 2004. 3.6.1

[87] M. Gruteser and D. Grunwald, "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking," in *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*. New York, NY, USA: ACM, 2003, pp. 31–42. 3.6.1

[88] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan, "Private queries in location based services: anonymizers are not necessary," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 121–132. 3.6.1

[89] "Directive 95/46/EC," http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML. 3.6.3

[90] "Directive 2002/58/EC," 2002, http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32002L0058:en:HTML. 3.6.3

[91] "Directive 2009/136/EC," http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32009L0136:en:NOT. 3.6.3

[92] "Article 29 Working Party (art. 29 WP)," http://ec.europa.eu/justice/data-protection/article-29/. 3.6.3

[93] "Working Party 29 Opinion 5/2005 on the use of location data with a view to providing value-added services," http://ec.europa.eu/justice/policies/privacy/docs/wpdocs/2005/wp115_en.pdf. 3.6.3

[94] "Spanish Telecommunications Act 32/2003," http://www.boe.es/buscar/doc.php?id=BOE-A-2003-20253. 3.6.3

[95] L. Xin and Q. liang Xu, "Practical compact multi-coupon systems," in *IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS)*, vol. 3, 2009, pp. 211 –216. 4.3

[96] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *Proceedings of the 3rd International Conference on Security in Communication Networks*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2003, vol. 2576, pp. 268–289. 4.6, 4.6

[97] K. Fujimura and Y. Nakajima, "General-purpose digital ticket framework," in *Proceedings of the 3rd USENIX Workshop on Electronic Commerce*, 1998, pp. 177–187. 5.2.1

[98] C.-P. Schnorr, "Efficient signature generation by smart cards," *Journal of cryptology*, vol. 4, no. 3, pp. 161–174, 1991. 5.5.2

[99] V. Wei, "Tracing-by-linking group signatures," in *Information Security*, ser. Lecture Notes in Computer Science, J. Zhou, J. Lopez, R. Deng, and F. Bao, Eds. Springer Berlin Heidelberg, 2005, vol. 3650, pp. 149–163. [Online]. Available: http://dx.doi.org/10.1007/11556992_11 5.7.1

[100] V. Benjumea, J. Lopez, and J. M. Troya, "Anonymous attribute certificates based on traceable signatures," *Internet Research*, vol. 16, no. 2, pp. 120–139, 2006. 5.7.1

[101] J. Liu, V. Wei, and D. Wong, "Linkable spontaneous anonymous group signature for ad hoc groups," in *Information Security and Privacy*, ser. Lecture Notes in Computer Science, H. Wang, J. Pieprzyk, and V. Varadharajan, Eds. Springer Berlin Heidelberg, 2004, vol. 3108, pp. 325–335. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-27800-9_28 5.7.1

[102] A. P. Isern-Deyà, A. Vives-Guasch, M. Mut-Puigserver, M. Payeras-Capellà, and J. Castellà-Roca, "A Secure Automatic Fare Collection System for Time-Based or Distance-Based Services with Revocable Anonymity for Users," *The Computer Journal*, 2012. 5.7.1

[103] S. Micali, "Simple and Fast Optimistic Protocols for Fair Electronic Exchange," *Proceedings of 21st Symposium on Principles of Distributed Computing*, pp. 12–19, 2003. 6.1, 6.6.1, 6.10

[104] F. Bao, G. Wang, J. Zhou, and Z. Zhu, "Analysis and Improvement of Micali's Fair Contract Signing Protocol," *Procedings of The 9th Australasian Conference on Information Security and Privacy*, pp. 176–187, 2004. 6.1, 6.8

[105] International Organization for Standardization, "ISO/IEC 19501:2005 - Unified Modeling Language (UML) Specification," Website, last checked in July, 2013, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=32620. 6.2

[106] ProVerif, Website, last checked in July, 2013, http://prosecco.gforge.inria.fr/personal/bblanche/proverif/. 6.2

[107] Cryptyc, Website, last checked in July, 2013, http://cryptyc.cs.depaul.edu/. 6.2

[108] PROMELA, Website, last checked in July, 2013, http://spinroot.com/spin/Man/promela.html. 6.2

[109] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1981. 6.3, 6.4

[110] CPNTools, Website, last checked in July, 2013, http://cpntools.org/. 6.4

[111] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems," *Int. J. Softw. Tools Technol. Transf.*, vol. 9, no. 3, pp. 213–254, May 2007. [Online]. Available: http://dx.doi.org/10.1007/s10009-007-0038-x 6.4

[112] J. Ferrer-Gomila, M. Payeras-Capellà, and L. Huguet-Rotger, "Optimallity in asynchronous contract signing protocols," *Trust and Privacy in Digital Bussines, TrustBus'04. LNCS 3184*, pp. 200–208, 2004. 6.6.2

[113] J. Garay, M. Jackobson, and P. MacKenzie, "Abuse-Free Optimistic Contract Signing," *CRYPTO'99, LNCS 1666, Springer Verlag*, 1999. 6.6.2

[114] J. Zhou, R. Deng, and F. Bao, "Some remarks on a fair exchange protocol," *PKC 2000, LNCS 1751, Springer Verlag*, pp. 46–57, 2000. 6.6.2

[115] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," Cryptology ePrint Archive, Report 2004/174, 2004, http://eprint.iacr.org/. 7.2.1, 7.2.2, 1, 7.2.3, 7.2.5

[116] A. N. Escalante, "Privacy-protecting multi-coupon schemes with stronger protection against splitting," in *Master's Thesis. Department of Computer Science. Saarland University*, 2008. 8.2.1, 11.3

[117] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1 – 11, 2011. 8.2.1

[118] Z. Xiao and Y. Xiao, "Security and privacy in cloud computing," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 2, pp. 843–859, 2013. 8.2.1

[119] "Google Drive," Website, last checked in July, 2013, 2013, https://drive.google.com. 8.2.1

[120] "Zoho," Website, last checked in July, 2013, 2013, http://www.zoho.com/. 8.2.1

[121] "Google App Engine (GAE)," Website, last checked in July, 2013, 2013, https://appengine.google.com/. 8.2.1

[122] "Windows Azure," Website, last checked in July, 2013, 2013, http://www.windowsazure.com/. 8.2.1, 8.2.1

[123] "Google Compute Engine," Website, last checked in July, 2013, 2013, https://cloud.google.com/products/compute-engine. 8.2.1

[124] "Elastic Cloud Computing (EC2)," Website, last checked in July, 2013, 2013, http://aws.amazon.com/ec2/. 8.2.1

[125] "Simple Storage Service (S3)," Website, last checked in July, 2013, 2013, http://aws.amazon.com/s3/. 8.2.1

[126] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing v3.0," 2011. [Online]. Available: https://cloudsecurityalliance.org/research/security-guidance/ 8.2.1

[127] S. Allen, V. Graupera, and L. Lundrigan, *Pro smartphone cross-platform development: iPhone, blackberry, windows mobile and android development and distribution*. Apress, 2010. 8.2.2

[128] Mobile Frameworks Comparison Chart Project, Website, last checked in July, 2013, 2013, http://www.markus-falk.com/mobile-frameworks-comparison-chart/. 8.2.2

[129] Y. Han, Y. Choi, and J.-K. Hong, "Experience on the development of a comsoc application for smart phones," *Communications Magazine, IEEE*, vol. 50, no. 4, pp. 106–112, 2012. 8.2.2

[130] IEEE 802.11™-2012, "Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications." 8.2.3

[131] FON Project: Your Global WiFi Network, Website, last checked in July, 2013, 2013, http://corp.fon.com/. 8.2.3

[132] International Telecommunication Union, "About mobile technology and IMT-2000," Website, last checked in July, 2013, 2005, http://www.itu.int/osg/spu/imt-2000/technology.html. 8.2.3

[133] NFC Forum, "NFC Forum Technical Specifications," NFC Forum website, last checked in 2013, July, http://www.nfc-forum.org/specs/spec_list/. 8.2.3

[134] R. Gass and C. Diot, "An experimental performance comparison of 3g and wi-fi," in *Passive and Active Measurement*, ser. Lecture Notes in Computer Science, A. Krishnamurthy and B. Plattner, Eds. Springer Berlin Heidelberg, 2010, vol. 6032, pp. 71–80. 8.2.3

[135] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (xml) 1.0 (fifth edition)," World Wide Web Consortium, Recommendation REC-xml-20081126, November 2008. 8.2.4

[136] D. Mundy and D. W. Chadwick, "An xml alternative for performance and security: Asn.1," *IT Professional*, vol. 6, no. 1, pp. 30–36, 2004. 8.2.4

[137] International Telecommunication Union, "Information Technology - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation," ITU-T Recommendation X.680, July 2002. 8.2.4

[138] ——, "Information Technology - ASN.1 Encoding Rules - Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER), and Distinguished Encoding Rules (DER)," ITU-T Recommendation X.690, July 2002. 8.2.4

[139] Larry Page, Google's CEO, "Android activations (2012, March)," Google+ profile, posted on March, 13th, 2013, http://googleblog.blogspot.com.es/2013/03/update-from-ceo.html. 8.3.1

[140] T. Chan, F. Chung, B. Carlstrom, and K. Root, "Unifying key store access in ice cream sandwich," Android Developers Blog, last checked in July, 2013, March 2012, http://android-developers.blogspot.com.es/2012/03/unifying-key-store-access-in-ics.html. 8.3.2

[141] "Bouncy Castle Library," Website, last checked in July, 2013, http://www.bouncycastle.org/java.html. 8.3.3

[142] "Spongy Castle Library," Website, last checked in July, 2013, http://rtyley.github.io/spongycastle/. 8.3.3

[143] B. Lynn, "PBC Library," 2012, http://crypto.stanford.edu/pbc/l. 8.3.3

[144] A. de Caro, "jPBC Library," 2012, http://gas.dia.unisa.it/projects/jpbc/index.html. 8.3.3

[145] L. J. Dominguez Perez, "Developing an automatic generation tool for cryptographic pairing functions," Ph.D. dissertation, Dublin City University, 2011. 8.3.3, 9.2.1, 9.2.1

[146] H. Cohen and G. Frey, *Hanbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman & Hall/CRC, 2006. 8.3.3, 9.2.1

[147] "Java Architecture for XML Binding (JAXB)," Website, last checked in July, 2013, http://jaxb.java.net. 8.3.3

[148] "Simple XML Framework," Website, last checked in July, 2013, http://simple.sourceforge.com. 8.3.3

[149] "Apache Maven," Website, last checked in July, 2013, http://maven.apache.org. 8.3.3

[150] "Subversion," Website, last checked in July, 2013, http://subversion.apache.org. 8.3.3

[151] "Eclipse," Website, last checked in July, 2013, http://www.eclipse.org. 8.3.3

[152] NIST, Website, last checked in July, 2013, http://www.nist.gov/. 8.3.4

[153] E. Barker and A. Roginsky, "NIST Special Publication 800-131A. Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths," U.S. Department of Commerce and National Institute of Standards and Technology (NIST), Tech. Rep., January 2011. 8.3.4

[154] J. W. Bos, M. E. Kaihara, T. Kleinjung, A. K. Lenstra, and P. L. Montgomery, "On the security of 1024-bit rsa and 160-bit elliptic curve cryptography," Cryptology ePrint Archive, Report 2009/389, 2009, http://eprint.iacr.org/. 8.3.4

[155] T. Kleinjung, K. Aoki, J. Franke, A. Lenstra, E. Thomé, J. Bos, P. Gaudry, A. Kruppa, P. Montgomery, D. A. Osvik, H. te Riele, A. Timofeev, and P. Zimmermann, "Factorization of a 768-bit rsa modulus," Cryptology ePrint Archive, Report 2010/006, 2010, http://eprint.iacr.org/. 8.3.4

[156] Amazon Web Services, "Amazon Elastic Compute Cloud: Instance Types and Compute Resources Measurement," Website, last checked in July, 2013, 2013, http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html#MeasuringComputeResources. 8.3.5, 8.3

[157] B. Lynn, "On the implementation of pairing-based cryptosystems," Ph.D. dissertation, Stanford University, 2007. 9.2.1, 9.2.1

[158] I. Blake, G. Seroussi, and N. Smart, *Elliptic curves in cryptography*, ser. London Mathematical Society Lectures Note Series. Cambridge Univ Pr, 1999, vol. 265. 9.2.1

[159] A. Miyaji, M. Nakabayashi, and S. Takano, "New explicit conditions of elliptic curve traces for FR-reduction," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 2001. 9.2.1

[160] M. Scott and P. Barreto, "Generating More MNT Elliptic Curves," *Designs, Codes and Cryptography*, vol. 38, pp. 209–217, 2006. 9.2.1

[161] P. Barreto and M. Naehrig, "Pairing-Friendly Elliptic Curves of Prime Order," in *Selected Areas in Cryptography*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, vol. 3897, pp. 319–331. 9.2.1

[162] D. Freeman, "Constructing Pairing-Friendly Elliptic Curves with Embedding Degree 10," in *Algorithmic Number Theory*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, vol. 4076, pp. 452–465. 9.2.1

[163] PrimB(49785), "Prime number of 2774 decimal numbers," 2003, downloaded from http://primes.utm.edu/primes/page.php?id=65151. 9.2.2

[164] International Telecommunication Union – ITU, "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks Technical Corrigendum 2," Series X: Data Networks, Open System Communications and Security Directory, nov 2008, iTU-T Recommendation X.509. 9.2.3

[165] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "RFC 5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," Tech. Rep., May 2008. [Online]. Available: http://tools.ietf.org/html/rfc5280 9.2.3, 9.2.3

[166] C. Adams, S. Farrell, T. Kause, and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)," RFC 4210 (Proposed Standard), Internet Engineering Task Force, Sep. 2005, updated by RFC 6712. [Online]. Available: http://www.ietf.org/rfc/rfc4210.txt 9.2.3

[167] T. Kause and M. Peylo, "Internet X.509 Public Key Infrastructure – HTTP Transfer for the Certificate Management Protocol (CMP)," RFC 6712 (Proposed Standard), Internet Engineering Task Force, Sep. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6712.txt 9.2.3

[168] W3C, "XML Signature Syntax and Processing (Second Edition)," Website, last checked in July, 2013, 2008, http://www.w3.org/TR/xmldsig-core/. 10.2.4