November 2015

Privacy-preserving Payments for Transportation Systems

Gesine Hinterwalder
University of Massachusetts Amherst

PRIVACY-PRESERVING PAYMENTS FOR TRANSPORTATION SYSTEMS

A Dissertation Presented

by

GESINE HINTERWÄLDER

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2015

Electrical and Computer Engineering

PRIVACY-PRESERVING PAYMENTS FOR TRANSPORTATION SYSTEMS

A Dissertation Presented

by

GESINE HINTERWÄLDER

Approved as to style and content by:

Prof. Christof Paar, Chair

Prof. Wayne P. Burleson, Member

Prof. Brian N. Levine, Member

Prof. Christopher V. Hollot, Department Head
Electrical and Computer Engineering

ACKNOWLEDGEMENTS

I am very grateful to have had the chance to pursue the PhD. During the past years I had the chance to develop and grow, not only academically, but also through meeting many interesting people and being involved in countless interesting and inspiring conversations. But, the PhD can also be challenging at times, and I would not have been able to succeed without the support of others.

First of all I would like to thank my adviser Prof. Dr.-Ing. Christof Paar for giving me the opportunity to pursue the PhD in the first place and supporting my wish to study abroad. Thanks also for trusting me to make the right decisions with respect to my research, and giving me the chance to work self-directed. I learned a lot by that. I would also like to thank my second adviser Prof. Wayne P. Burleson. Thanks for supporting me and my work. Thanks also for looking at my work from a different perspective, which always resulted in interesting discussions. Further, I would like to thank Prof. Brian Levine and Prof. Thomas Eisenbarth for serving on my dissertation committee, as well as Prof. Dennis L. Goeckel and Prof. Csaba Andras Moritz for serving on my research qualifying exam's committee.

I would like to thank the many collaborators, namely Foteini Baldimtsi, Björn Haase, Michael Hutter, Anna Lysyanskaya, Amir Moradi, Andy Rupp, Ana Helena Sánches, and Peter Schwabe, whom I had the chance to work with during my PhD. Thanks especially to Andy and Foteini, who engaged in introducing me into e-cash basics. Many thanks to the students Olga Korobova (who is also a great friend), Michael Düll, Felix Riek, and Christian Zenger, whom I supervised during my PhD time.

I would like to express my gratitude towards the members of the VCSG and SPQR groups at University of Massachusetts Amherst, who were always open for discussions and supported me when having to make critical academic decisions. Here to mention in particular Georg Becker, Ibis Benito and Vikram Belur Suresh who were constant companions during my studies at UMass Amherst. Similarly, I was warmly welcomed by the EMSEC and SHA groups in Bochum during several shorter visits and one extended research stay in the last year of my PhD. Here to mention in particular Ingo von Maurich, who struggled with me through the SecMobil project, Irmgard Kühn and Horst Edelmann, who helped with all administrative and technical stuff, and Daehyun Strobel and Ralf Zimmermann, who generously shared their offices with me.

My very special thanks goes to Bahar Haghanipour, who at the right moment jumped in and gave me the support that I needed to pursue the PhD degree.

Yet, the academic support was only one side of the medal. I am grateful to have received tons of support from people surrounding me. With respect to this, I would like to express my deepest gratitude towards my family Anja, Karl, Kilian, Marla, Monika, Sebastian, Theresa, and Ulrike who formed and supported me and taught me to work hard and fulfill my dreams. I am looking back at many, many phone calls in which I knew I was not alone with my challenge. That was a big help, and I am grateful for having an amazing and caring family.

Last but not least, I would like to thank friends, who supported me during the PhD time, in particular Annika & Daehyun, Falk, Georg, Ingo, Jen, Jing, Krzysiek, Leopoldo, Manisha, Mona, Melli, Olga, Olli, Peter, and Steffi. Thanks for being there, cheering me up, making me laugh, and making me enjoy my free time.

Thanks to you all, you rock!!!

ABSTRACT

PRIVACY-PRESERVING PAYMENTS FOR TRANSPORTATION SYSTEMS

SEPTEMBER 2015

GESINE HINTERWÄLDER

Dipl.-Ing., RUHR-UNIVERSITY BOCHUM

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Prof. Christof Paar

The operation of our society heavily relies on high mobility of people. Not only our social life but also our economy and trade are built upon a system where people need to be able to move around easily. The costs for building and maintaining a suitable transportation infrastructure to satisfy those needs are high, and to charge users is thus a central requirement. This calls for well functioning payment systems satisfying the multitude of requirements that transportation systems impose on them.

Electronic payment systems have many benefits over traditional cash payments as they are easy to maintain, can be more secure, reduce revenue collection costs, and can reduce the execution time of a payment. However, as a drawback, currently employed electronic payment systems usually reveal a payer's identity during a payment which greatly infringes customer privacy. In the transportation domain this allows to generate fine grain patterns of customers' locations.

Cryptographic payment protocols called e-cash have been proposed which allow to preserve a customer's privacy. E-cash provides provable guarantees for both security and user privacy, as it allows secure, unlinkable payments which do not reveal the identity of the payer during a payment. From a security and privacy perspective these protocols present a good solution. However, even though e-cash protocols have been proposed three decades ago, there are relatively few actual implementations. One reason for this is their high computational complexity which makes an implementation on potential mobile payment devices rather difficult. While customers usually value their privacy they often do not accept to sacrifice convenience. A fast execution of payments is thus a hard constraint, which conflicts with the computational complexity of e-cash schemes.

This dissertation analyzes how e-cash can be used to solve the issue of privacy in the domain of transportation payments while satisfying the unique requirements of transportation payment systems and achieving high security and ease of use. Highly-efficient implementations of the underlying cryptographic primitives of e-cash schemes on constrained devices as they might be used in the transportation setting are presented. Based on the efficient implementations of these primitives, e-cash schemes are analyzed with regards to speed and hardware requirements. The results show that e-cash presents a good solution for privacy-preserving payments in the domain of public transport, if the number of coins that have to be spent can be limited. It is further practically shown that this limitation can be alleviated relying on the e-cash based privacy-preserving pre-payments with refunds scheme (P4R). Moreover, it is demonstrated that the promising feature of supporting the encoding of user attributes into electronic coins can be implemented at only moderate extra cost. Finally, an e-cash based e-mobility payment scheme is presented which highlights the flexibility and unique advantages of e-cash based transportation payment schemes.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS ..... **iv**

ABSTRACT ..... **vi**

LIST OF TABLES ..... **xii**

LIST OF FIGURES ..... **xv**

CHAPTER

1. INTRODUCTION **1**

1.1 Electronic Payment Systems 2
1.2 Information Privacy – Should we care about it? 5
1.3 Organization and Contribution of this Dissertation 7

2. BACKGROUND **13**

2.1 Electronic Cash 13

2.1.1 E-cash from (Blind) Digital Signatures 15
2.1.2 E-Cash Protocols 17

2.2 Elliptic Curve Cryptography 18

3. EFFICIENT IMPLEMENTATION OF HIGH-SECURITY ELLIPTIC CURVE CRYPTOGRAPHY ON MSP430 MICROCONTROLLERS **23**

3.1 Related Work 25
3.2 The MSP430X Microcontroller Architecture 26
3.3 Review of Curve25519 28
3.4 Implementation of Modular Arithmetic in $\mathbb{Z}_{2^{255}-19}$ 30

3.4.1 Representation of Big Integers 31

3.4.2 Multiplication Using Carry-Save Representation 31
3.4.3 Operand-Caching Multiplication 33
3.4.4 Product-Scanning Multiplication 35
3.4.5 Constant-Time Karatsuba Multiplication 36
3.4.6 Squaring 37
3.4.7 Addition, Subtraction, Reduction and Multiplication with 121666 38

3.5 Performance Analysis 40
3.6 Power Consumption Analysis 43
3.7 Discussion 45

4. EFFICIENT E-CASH ON PASSIVE RFID TOKENS TARGETING PUBLIC TRANSPORTATION SYSTEMS 47

4.1 Related work 51
4.2 Review of Brands' Untraceable Offline Cash Scheme 52
4.3 Implementation of Brands' E-cash on the Moo RFID Tag 56

4.3.1 Implementation of the $\mathbb{Z}_p$ Framework 57
4.3.2 Implementation of the ECC Framework 58
4.3.3 Implementation of the $\mathbb{Z}_q$ Framework 60
4.3.4 Implementation of a Hash Function 60

4.4 Performance of Brands E-Cash on the Moo RFID Tag 61
4.5 Review of the Privacy-Preserving Pre-Payments with Refunds Scheme (P4R) 63
4.6 Performance of P4R on Moo RFID tag 68
4.7 Performance Comparison of P4R with Brands' E-cash 71
4.8 Discussion 72

5. IMPLEMENTATION OF E-CASH WITH ATTRIBUTES FOR PUBLIC TRANSPORTATION SYSTEMS 73

5.1 Related Work 76
5.2 Review of E-cash with Attributes 77

5.2.1 Brands' E-cash Allowing the Encoding and Selective Disclosure of Attributes 77

5.2.1.1 User Registration 78
5.2.1.2 Withdrawal 79
5.2.1.3 Spending 80
5.2.1.4 Deposit 81

5.2.2 ACL E-cash Allowing the Encoding and Selective Disclosure of Attributes . . . 82

5.2.2.1 User Registration . . . 82
5.2.2.2 Withdrawal . . . 82
5.2.2.3 Spending . . . 83
5.2.2.4 Deposit . . . 84

5.3 Implementation of E-cash with Attributes on a BlackBerry Bold 9900 . . . 85

5.3.1 Near Field Communication (NFC) Framework . . . 86
5.3.2 Cryptographic Framework . . . 87

5.3.2.1 Efficient Execution of EC Scalar Multiplication Using the ECDH Key Agreement . . . 88

5.4 Performance of E-cash with Attributes on an NFC-Smart-phone . . . 90
5.5 Implementation of E-cash with Attributes on a MULTOS Smartcard . . . 95

5.5.1 Framework Implementation . . . 98

5.6 Performance of E-cash with Attributes on a MULTOS Smartcard . . . 99
5.7 Discussion . . . 104

6. LIGHTWEIGHT ANONYMOUS NFC-PAYMENTS FOR E-MOBILITY . . . 106

6.1 Related Work . . . 108
6.2 E-mobility Payment Systems . . . 108
6.3 Security and Privacy Objectives . . . 110
6.4 Payment Scheme Description . . . 111

6.4.1 System Setup . . . 112
6.4.2 User Registration . . . 113
6.4.3 (Re-) Charge Electronic Wallet . . . 114
6.4.4 Start Charging . . . 115
6.4.5 Stop Charging . . . 116
6.4.6 Redeem Refund . . . 117
6.4.7 Double-Spending Detection . . . 117

6.5 Security and Privacy Analysis . . . 119
6.6 Implementation on an NFC-Smartphone . . . 122
6.7 Performance of the Proposed Payment Scheme on an NFC-Smartphone . . . 124

6.8 Variations 128
6.9 Discussion 129

7. CONCLUSION AND OUTLOOK 130

BIBLIOGRAPHY 133

LIST OF TABLES

Table **Page**

3.1 Measured cycle counts when executing the various multiplication implementations on the MSP-EXP430FR5969 Launchpad Evaluation Kit, operating the microcontroller at 8 MHz and 16 MHz respectively. The numbers marked with (a) include the function call and reduction overhead while those marked with (b) exclude it. 41

3.2 Code space (in bytes) required for the variious modular multiplication implementations (including reduction) on MSP430Xs obtained by the IAR Embedded Workbench IDE. 42

3.3 Stack space required for modular multiplication implementations (including reduction) on MSP430Xs obtained flushing RAM with a special character and observing, which RAM contents have been changed after the execution of each multiplication operation. 43

3.4 Cycle counts, code sizes and stack usage of elliptic-curve scalar-multiplication software for MSP430X microcontrollers 44

4.1 Brands' withdrawal protocol [21]. In this protocol the user randomly chooses a serial number `SN` and encodes it in a serial number tag A. A and a blinding factor B, which is also formed by the user are blindly signed by the bank. 53

4.2 Brands' spending protocol [21]. In this protocol the user sends a coin to a shop, which checks its validity as well as whether the user is in possession of it. It accepts the payment, if both checks hold. 55

4.3 Timings of the $\mathbb{Z}_p$, $\mathbb{Z}_q$, and ECC framework, and the Hash functions $\mathcal{H}$ 61

4.4 Timing results of the user sides' computation of Brands' offline cash scheme on the MSP4302618 62

4.5 Code size of the user sides' implementation of Brands' scheme on the MSP430F2618 62

4.6 P4R's BuyTAT and GetRT protocol. In the BuyTAT protocol the user randomly chooses a serial number SN and encodes it in a serial number tag A. He further generates two blinding factors B_1 and B_2 and receives a blind signature from the bank on A, B_1 and B_2. In the GetRT protocol the user receives a fresh RT. 66

4.7 P4R's ShowTAT and GetRCT protocol. In this protocol $\mathcal{U}$ presents a TAT and $\mathcal{S}$ checks its validity as well as whether $\mathcal{U}$ possesses it. The payment is accepted, if both checks hold and $\mathcal{U}$ receives an RCT. 67

4.8 P4R's ShowRCT and GetRefund protocol. In the ShowRCT protocol the user presents his RCT to the exit turnstile, which checks its validity and computes the refund w the user should receive based on it. In the GetRefund protocol the user presents a blinded version of her refund token and the turnstile adds w to it. 67

4.9 P4R RedeemRT protocol 68

4.10 Timing results for the user side's computation of The P4R protocols: BuyTAT, GetRT, ShowTAT & GetRCT, ShowRCT & GetRefund and RedeemRT 69

4.11 Storage space estimation for payment data in P4R 70

4.12 Comparison of Brands' e-cash for coins with a denomination of 10 cents and P4R (for MSP430 operating at 4 MHz when user enters/leaves and at 16 MHz when he charges his payment device) 71

5.1 Brands' with attributes user registration protocol 78

5.2 User identification phase 79

5.3 Brands' with attributes withdrawal protocol 80

5.4 Brands' with attributes spending protocol when revealing attribute L_j 81

5.5 ACL with attributes user registration protocol 83

5.6 ACL with attributes withdrawal protocol 84

5.7 ACL with attributes spending protocol revealing the attribute L_j 85

5.8 Execution times of withdrawal per coin for i) Brands without attributes, ii) Abe, which does not support the encoding of attributes, iii) Brands encoding two attributes and iv) ACL encoding two attributes. 91

5.9 Execution times of spending per coin for i) Brands without attributes, ii) Abe, which does not support the encoding of attributes, iii) Brands encoding two attributes and revealing both and iv) ACL encoding two attributes and revealing both of them. 91

5.10 Coin size (per coin data stored on user device) for the cases i) Brands without attributes, ii) Abe, which does not support the encoding of attributes, iii) Brands encoding two attributes and iv) ACL encoding two attributes. 94

5.11 Execution times of the withdrawal protocols of Brands' untraceable offline cash scheme [21] and Anonymous Credentials Light (ACL) [7] for the case of encoding two attributes into a coin based on the brainpoolP160r1 (160-bit) and the brainpoolP256r1 (256-bit) elliptic curves. 100

5.12 Execution times of the spending protocols of Brands' untraceable offline cash scheme [21] and Anonymous Credentials Light (ACL) [7] based on brainpoolP160r1 (160-bit) and brainpoolP256r1 (256-bit) elliptic curves for the case that two attributes are encoded in a coin but no attribute is revealed. 102

6.1 List of refund check values 113

6.2 P4R for e-mobility user registration protocol. 114

6.3 P4R for e-mobility (re-) charge electronic wallet protocol. 115

6.4 P4R for e-mobility start charging protocol 116

6.5 P4R for e-mobility stop charging protocol 118

6.6 P4R for e-mobility redeeming refund protocol. 119

6.7 Execution times of the various protocols of our payment scheme, based on a Google Nexus 5 NFC-smartphone, using host-based card emulation. The timings are averaged for refund values of all possible hamming weights. 125

LIST OF FIGURES

Figure **Page**

2.1 E-cash concept — A user receives electronic coins from the bank, which is the only entity able to generate coins, during a process called withdrawal. He will later use these coins to pay at a shop during a process called spending. The shop can deposit coins it received from users over a period of time to its bank account. 14

3.1 Visualisation of computing the first coefficients of the array $h \leftarrow f \times g \mod 2^{255} - 19$ using the carry-save technique. The dark grey block originates from reducing the double-sized array, which is the result of multiplying f with g, modulo $2^{255} - 19$. 32

3.2 Toy-size example of the operand-caching multiplication, when 8 general-purpose registers are available to hold operands. To compute the coefficients of the array $h \leftarrow f \times g$ one first computes the light grey block keeping all required input operands in general-purpose registers. Thereafter the dark grey area is computed. 33

3.3 Visualization of computing the first coefficients of $h \leftarrow f \times g$ using the product-scanning technique. We first compute the double-sized array, which results from multiplying f with g and then reduce this double-sized array modulo $2^{256} - 38$. 35

3.4 Visualization of the execution times of the various multiplication implementations (Table 3.1) including function call and reduction overhead for (a) 16-bit hardware multipliers and (b) 32-bit hardware multipliers on the MSP-EXP430FR5969 Launchpad Evaluation Kit. The methods are numbered according to Table 3.1. 41

3.5 A sample power trace measured from MSP-EXP430FR5969 Launchpad Evaluation Kit when running 7 different multiplications 44

4.1 Overview of a public transportation payment system. A transportation authority owns and controls vending machines, where users buy tickets, and turnstiles, which a user can pass to access a transportation system, if he presents a valid ticket. 47

5.1 Illustration of the execution times of withdrawal per coin for i) Brands without attributes, ii) Abe, which does not support the encoding of attributes, iii) Brands encoding two attributes and iv) ACL encoding two attributes. 92

5.2 Illustration of the execution times of spending per coin for i) Brands without attributes, ii) Abe, which does not support the encoding of attributes, iii) Brands encoding two attributes and revealing both and iv) ACL encoding two attributes and revealing both of them. 93

5.3 Execution times of Brands' and ACL withdrawal protocols at the 80-bit and 128-bit security levels, encoding two attributes (illustration of the results presented in Table 5.11). 100

5.4 Execution times of Brands' and ACL spending protocols at the 80-bit and 128-bit security levels, encoding two attributes but revealing no attribute (illustration of the results presented in Table 5.12). 103

6.1 Overview of a charging infrastructure for e-mobility. We differentiate between payment service providers, who process and check the correctness of payments, and electricity providers, who provide a charging station infrastructure, *i.e.* are in possession of charging stations and provide electric energy. 109

6.2 Illustration of Table 6.7, presenting the execution times of the various protocols of our payment scheme, based on a Google Nexus 5 NFC-smartphone, using host-based card emulation. The timings are averaged for refund values of all possible hamming weights. 126

6.3 Execution times of the *End Charging* protocol for different refund values, incrementing the hamming weight of the refund value in each column. 128

CHAPTER 1

INTRODUCTION

Our ever-growing globalized world largely relies on high mobility of people, and it is anticipated that transport activity will only increase in the future. At the same time, the distribution of the world's population is changing to an urban one. While in 2011 about 52% of the world's population lived in urban areas, it is anticipated that this number will increase to 67% by the year 2050. Additionally, cities grow larger which leads to an expected number of 37 mega-cities with more than 10 million inhabitants by 2025 [31].

Especially in cities, where the population density is high, it is important to find means to reduce Green House Gas (GHG) emissions in order to limit air pollution. Public transport and e-mobility (the use of electric cars) constitute an appealing alternative to fuel-based personal vehicles and can greatly reduce GHG emissions in areas where the population density is high. Additionally, public transport is favorable from a social point of view, as it gives higher mobility to people who, due to their income, do not have access to cars [83]. Electric cars avoid local emissions, thus allowing to regulate the emissions' location, and, as operated based on electric energy, renewable energies can at least partially reduce the dependence of transport on oil resources. To attract people to rely on these types of transport the provision of a well-functioning public transport infrastructure and a charging infrastructure for electric cars is of major interest.

However, providing and maintaining a well-functioning transportation infrastructure is expensive. Use-based fees and payments allow the costs of transportation

systems to be fairly passed on to their users, facilitating revenue generation and user incentives. Yet, collecting a large number of payments in a way that it does not impact the smooth operation of a transportation system is not an easy task. Well functioning transportation payment systems are required, where the transaction duration of a payment is an essential acceptance criterion. If it takes too long, it easily leads to congestion at the access points of the transportation system prolonging the overall trip time of individuals.

When compared to traditional cash payments, electronic payments have many benefits, especially if points of sales are greatly spread as is the case in the transport domain. As such they can increase security, they reduce maintenance cost, as no cash has to be carried around physically, and their execution is usually much faster. The shorter execution times as well as the fact that no cash has to be carried around greatly improves user convenience. For a transportation authority a further advantage of electronic payments is that they enable the collection of meaningful data about customer behavior, which helps to maintain and improve the provided infrastructure.

1.1 Electronic Payment Systems

Usually, three types of entities are involved in a payment system: payers, payees and a financial institution [5]. This is in contrast to the bitcoin or related systems, where no financial institution is backing the system, but rather the system is based on a peer-to-peer network and an algorithm which ensures that payments are executed correctly as long as the majority of participants plays honestly [75].

Electronic payment systems can be classified by their transaction date. In pay-later systems, such as credit card and electronic check systems, a customer receives a bill over a payment after making a purchase. In pay-now systems, such as debit card systems, the amount a customer has to pay is transferred to the merchant's account during a payment. In prepaid systems, which are similar to cash like systems, a cus-

tomer withdraws money from her bank account, and stores it in an electronic wallet. She can later use the money to make purchases [5, 80, 90]. Prepaid systems are usually used for micro-payments (payments of values below $1) and low-value payments (payments in the range of $1 to $100), whereas pay-now and pay-later systems are preferred for high-value payments. In the transportation domain prepaid systems that are realized using smart cards gain popularity.

Electronic payments can also be categorized by their verification process. Online (centralized) payment systems involve the interaction with a third party or the back-end system of the financial institution to validate a payment. This is for example the case with debit cards, whereas offline (decentralized) payments can be validated autonomously by the device accepting the payment [5, 81, 90]. Online payment systems have drawbacks with respect to availability. All devices accepting payments have to have constant access to the back-end system of the controlling financial institution. If this connection or the back-end system itself fails no payment can be executed, and it is rather expensive to introduce redundancy in a back-end system.

Electronic payment systems can further be classified by the physical characteristics of their payment devices. One can differentiate between contact payment devices (as for example magnetic stripe cards, EEPROM cards, and contact smart cards), proximity cards which do not require a physical connection but only have to be brought into close proximity of a reader, and short-range communication systems, such as radio frequency (RF) transponders or radio frequency identification (RFID) tags. Magnetic stripe cards provide only limited security but are widely used due to their extremely low cost. EEPROM cards have a little integrated circuit with hardwired functionality to read out the value stored on them. Smart cards have a microprocessor, read only memory (ROM) to store an ID and an operating system, and electrically erasable programmable read only memory (EEPROM) to store payment information and history. Thus they provide much better security and multi-

functionality. Contactless payments are often preferable, as a payer does not have to insert a payment device into a slot or otherwise physically connect it to a terminal, which can lead to significantly reduced transaction times. Contactless smart cards have an inductive coil, which is coupled to another inductive coil in a reader using RF magnetic fields with a frequency of 125 kHz or 13.56 MHz. RF-transponders or RFID-tags use backscatter technology to modulate the 900 MHz up to 2450 MHz electromagnetic field of a reader in response to his requests.

The security requirements of electronic payments are: (I) Integrity and authorization which requires that a malicious user should not be able to create, modify, or overspend monetary value and a payment should only be initiated after consent of all involved parties. (II) Confidentiality requires that payment data cannot be exploited in order to generate user profiles. Different levels of anonymity can be achieved. Information leakage to outsiders can for example be prevented by encrypting payment data. But payments can also be anonymous or even unlinkable for the payee and the financial institution. (III) Availability and reliability requires that payments can be executed at all times and that they are either executed entirely or not at all, *i.e.* parties involved in payments should not loose money due to a system crash [5, 90].

Several examples demonstrate shortcomings of electronic transportation payment systems, which are in use today. One is the use of proprietary, weak cryptographic primitives, which leave room for attacks as has for example been demonstrated in [38, 63] and [88]. Another shortcoming is that they do not incorporate means to protect the user's (location) privacy. For example, Rankl *et al.* reported that "in the period from August 2004 to March 2006 alone, the Oyster system[1] was queried 409 times" which indicates that location data about customers is collected and used by other agencies [81]. The data that is collected in these systems allows to derive fine grain

[1]Oyster-card is an electronic prepaid payment system for transport in London [35].

patterns of customers' movements and habits. We speculate that the limited effort to provide anonymous electronic payment systems originates from the facts that (a) providing anonymity is costly and (b) collecting user data is useful to merchants and financial institutions. While it has become *common* to share one's personal information, there are convincing arguments why preserving one's privacy is extremely important.

1.2 Information Privacy – Should we care about it?

> [Location privacy] is the ability of an individual to move in public space with the expectation that under normal circumstances their location will not be systematically and secretly recorded for later use. [...]. [Location privacy] is also about knowing when other people know things about you, and being able to tell when they are making decisions based on those facts.
>
> —Andrew J. Blumberg [18]

People are often willing to give up their privacy in order to reach an increase in convenience and security. When asked why they are not concerned about sacrificing their privacy the most common answer is: "I've got nothing to hide" [94]. The general perception is that privacy is only of interest to people who engage in illegal activity. However, maybe people do not fully grasp the power of data collection and the impact of disclosing certain personal data. As such, while saying that only *bad* people require privacy-preserving measures, most people do not feel comfortable to reveal every detail of their lives in public [43]. As a motivation for this dissertation, this section briefly tries to answer: What are examples of threats that arise from disclosing personal data? And related to that, why is it important to protect one's personal data?

A first argument for providing an institution with as little personal information as absolutely necessary is that one has to trust the institution to store the information securely and prevent malicious parties from having access to it. Even though the

institution itself might not use the information in a bad way, once in their possession it is in their power to decide how to secure the data to not give access to malicious parties. Multiple incidents, in which user data was stolen from companies at a large scale, e.g. [3] and [79], show that companies do not always store user data secure enough.

A second argument is that entities in possession of personal data have great power over the behavior of other people. People behave differently, if they cannot move around freely. Glenn Greenwald explains this as "human shame [being] a powerful motivator" [43]. For example, people usually reveal different information about themselves to a doctor than they would to their employers. Being anxious that information given to a doctor could at a certain point reach one's employer, would keep people from trusting their doctor. Hence, from a sociological perspective the lack of privacy will change interactions within a society as the trust level decreases [70]. This also has a great impact on democracy, if politicians can be blackmailed by institutions in possession of their personal data.

A third argument is that the power of interpretation of data is given to the entity in possession of it. In extreme cases an individual has no opportunity to clarify things and correct misinterpretations. This is in particular the case if the victim does not even know that data about her is collected.

One could assume that it is sufficient to obscure one's identifier, *i.e.* stripping off one's name from a data set, reaching what is called pseudonymity. Yet, patterns and data sets can be combined to identify an individual, who a data set belongs to. Sweeny found that in the US knowing someones ZIP code, his gender and date of birth is sufficient to uniquely identify 87% of Americans. Doing so she identified William Weld's (Massachusetts' governor at that time) pseudonymous health record in the data set of the Group Insurance Comission (GIC) [98].

Especially in an era where vast amounts of computing power are available, the sorting and interpretation of data becomes easier. A general solution is to force companies by law to delete customer data after a certain time. However, there is no guarantee for customers that the law cannot be suddenly changed. The easiest solution to combat these uncertainties is to build systems that protect a customer's privacy inherently, meaning by design.

1.3 Organization and Contribution of this Dissertation

Electronic cash, short e-cash, which maps physical cash into the digital world, has been introduced in 1982 by David Chaum as a means of secure and privacy-preserving electronic payments [24]. Following his approach e-cash has become an area of cryptographic research interest and many schemes were proposed, e.g. [10, 21, 22, 23, 26]. However, while e-cash seems to solve the issue of privacy in electronic payment systems, e-cash schemes require the execution of computationally expensive public-key operations. This can be a limiting factor in terms of usability, especially considering that potential payment devices are usually constrained in size and power and thus in computational capabilities. In this dissertation we focus on the specific case of using e-cash to solve the issue of privacy in the domain of transportation payments. We investigate the usability of specific e-cash schemes through practical analyses, *i.e.* we develop highly efficient implementations of e-cash schemes on potential payment devices and evaluate their efficiency and usability. This requires highly efficient implementations of the underlying cryptographic primitives. We base the implementations on elliptic curve cryptography (ECC) and show various approaches to implement ECC efficiently even on extremely constrained platforms. Yet, the contribution of this dissertation is not limited to demonstrating and evaluating implementation results of a theoretical concept. We present a multitude of ideas of how e-cash can fulfill

the unique requirements of transportation payment systems, thus demonstrating how e-cash can be used in practice.

A central requirement for payment devices is that they work in a contactless fashion, as contact-based payment devices conflict with the need for short transaction times. Additionally, contactless payment devices greatly increase customer convenience as they provide ease-of-use. The use of devices such as NFC-smartphones is a promising solution, due to the facts that a user can use a single device for various applications and a payment service provider only has to offer a software app rather than a hardware token to users, which greatly decreases development and deployment costs. However, it is often at least additionally desirable to offer (extremely) inexpensive payment tokens. This greatly increases the number of users that can participate in a payment system. Further, even if relying on powerful smartphones, it is desirable to execute security-critical functions on secure hardware, such as the embedded secure element of a smartphone. Those chips usually have similarly powerful hardware as is used in smart cards. In this dissertation we consider both types of platforms.

The remaining part of this dissertation is organized as follows: We first give an introduction into background knowledge, which is required for an understanding of the rest of the dissertation, in Chapter 2. We introduce e-cash basics focusing on techniques which are used in the schemes of which our implementation will be presented subsequently. We then introduce elliptic curve cryptography (ECC), which we based the implemented e-cash schemes on, as it is the most efficient established asymmetric cryptographic primitive.

Texas Instrument's MSP430 microcontrollers target low-power applications among which are wireless sensor, metering, and medical applications. These are applications in which a high level of security is required. While newer devices of the MSP430 family have AES hardware accelerators that support 256-bit AES, many security services rely on public-key cryptography. Curve25519, which builds on a 255-bit prime field, has

been proposed as an efficient, highly-secure elliptic curve, extremely suitable for executing the elliptic curve Diffie-Hellman key-exchange protocol. This protocol, which was recently named X25519, essentially executes a variable base-point single-scalar multiplication on Curve25519. Chapter 3 presents an implementation of X25519 for MSP430X microcontrollers. To combat timing attacks, we completely avoid conditional jumps and loads, making our software constant time. The scalar multiplication makes excessive use of modular multiplications in the underlying prime field. Many techniques have been proposed to efficiently implement modular multiplications on microcontrollers with only limited computational capabilities. However, the effectiveness of those techniques depends on the architecture of the processing unit. We present a comprehensive evaluation of several implementation techniques of the modular multiplication and show which ones are favorable on the MSP430X under various conditions. We further present implementation results of the Curve25519 scalar multiplication, where our best implementation requires 7 909 028 cycles on MSP430Xs having a 16-bit hardware multiplier and in 5 332 487 cycles on MSP430Xs having a 32-bit hardware multiplier.

An electronic cash scheme, particularly known for its efficiency during the spending phase, was proposed by Stefan Brands in [21]. This scheme is highly attractive for use in the domain of public transport, where the spending, which happens in front of turnstiles, is extremely time constrained, especially during rush hours. In Chapter 4 we demonstrate that, using sophisticated implementation techniques, it is possible to realize this e-cash scheme even on inexpensive payment tokens. We implement Brands' untraceable offline cash scheme for the Moo, a computational RFID-token, which closely approximates hardware that could be used in low-cost payment devices. The computation required for spending a coin can be executed in 13 ms on the Moo, which meets real-world application requirements. The computation required for receiving an electronic coin is time consuming on the Moo, which suggests to limit the

amount of coins that have to be spent for a fare. A solution would be to define zones in a transportation system, and then build a ticketing system, where a coin is worth a fare in a particular zone. It is however highly desirable to allow for flexible and dynamic prices, e.g. to adapt a fare to the distance traveled and the transport conditions. While a solution would be that a user could receive electronic coins as change, Brands' scheme makes it difficult to realize change in a privacy-preserving way. Additionally, this would diminish the attractiveness of Brands' scheme which is its user side's efficiency during the spending phase. Instead we show an implementation of the privacy-preserving pre-payments with refunds (P4R) scheme, which targets the specific requirements of public transportation systems. In this scheme an electronic coin is worth the most expensive trip in a transportation system, requiring a user to spend a single coin per trip. She then receives a refund based on her overpayment in a privacy-preserving way.

An attractive feature of some e-cash schemes is that they support the encoding and selective disclosure of users' attributes into electronic coins. This allows for additional features such as variable pricing (e.g. reduced fares for customers who can prove participation in a discount program) and privacy-preserving data collection. Supporting privacy-preserving data collection is essential in systems, where meaningful user data helps to maintain and improve a system. Chapter 5 presents an efficient implementation of Brands e-cash [21] and Anonymous Credentials Light (ACL) [7] used as e-cash, allowing the encoding of user attributes into electronic coins on an NFC-smartphone, namely the BlackBerry Bold 9900. Near field communication (NFC) is a recent popular technology for contactless electronic payments. Due to the limitation of having to use the BlackBerry Java API, we present a subtle technique to make use of the `ECDHKeyAgreement` class that is available in the BlackBerry API (and in the API of other systems) and show how the schemes can be implemented efficiently to satisfy the tight timing imposed by the transportation setting. While we show that it is

possible to execute both schemes efficiently on the BlackBerry Bold 9900, we recall from above that in practice it is highly desirable to execute security critical parts of payment applications on a secure device, such as a smartcard or the secure element of a smartphone, instead of a smartphone's main processor. A hurdle in practice is that it is surprisingly difficult for researchers to have free access to smartcards or secure elements, i.e. to program them using native code. UBM 21-Z48 cards from UbiVelox support a variety of cryptographic primitives, in particular elliptic curve cryptography, allowing the implementation of the considered protocols on them. We present an implementation of both schemes on UbiVelox UBM 21-Z48 cards for two security levels and show that when relying on a 160-bit elliptic curve, spending a coin, which encodes two attributes that are not revealed, can be executed in less than 800 ms.

In Chapter 6 we examine e-Cash in the e-mobility setting. E-mobility is a promising transportation technology, especially in densely populated cities. A crucial factor for user acceptance of e-mobility is the provision of a well functioning charging infrastructure that allows convenient use of electric mobiles. Reliable payment schemes to pay for electricity play a major role in this respect. We target the problem of privacy-preserving payments for the e-mobility domain in Chapter 6. As mentioned above, a major drawback of Brands' untraceable offline cash scheme is the difficulty of realizing change in a privacy-preserving way. If trying to limit the number of coins that have to be spent per payment, this is however a feature most wanted in a domain, where the prices greatly vary, which is the case in the e-mobility domain. A solution to this is the P4R scheme, mentioned above, which rather then relying on a user being able to receive electronic coins as change, realizes a privacy-preserving refund system. We show how P4R can be used to solve the issue of privacy in the domain of e-mobility payments. Using the same payment scheme for public transportation payments and payments in the e-mobility domain allows using the same payment service for both types of transport, which presents a key factor in achieving customer convenience.

We present a full implementation of our proposed solution on an NFC-smartphone, namely the Google Nexus 5, which supports Android 4.4 KitKat. We analyze the performance of our proposed scheme on this platform, demonstrating its attractiveness for use in the e-mobility domain.

Finally, we conclude with a discussion of our results and analyses, and provide an outlook on future research directions in Chapter 7.

CHAPTER 2

BACKGROUND

In this chapter we first review some basics of e-cash schemes in Section 2.1. The e-cash schemes that we consider are based on arithmetic in cyclic groups G of prime order q. We base the implementation of those schemes on elliptic curve cryptography (ECC) of which we review basics in Section 2.2.

2.1 Electronic Cash

E-cash schemes allow secure and private electronic payments by providing similar security and anonymity as physical cash. The general e-cash concept, which is illustrated in Figure 2.1, describes the interaction between three types of entities: the bank $\mathcal{B}$, users $\mathcal{U}$, and shops $\mathcal{S}$. The bank plays the role of the central authority checking the correctness of payments. However, it does not act as a trusted authority; security and privacy is ensured against all entities. Monetary value is represented by electronic coins, which are generated through an interactive protocol between the bank and a user during a process called *withdrawal*. In the schemes that we consider in this dissertation a coin consists of a serial number, which is randomly chosen by a user and blindly signed by the bank during the withdrawal process. That means it is signed by the bank in a way that the bank afterwards does not know neither the data it has been signing nor the signature it has produced. The serial number serves to later on identify a coin and check whether it has been spent multiple times.

Blind signatures have two important properties, which (informally stated) are:

- Blindness: The signer knows nothing about the data it is signing.

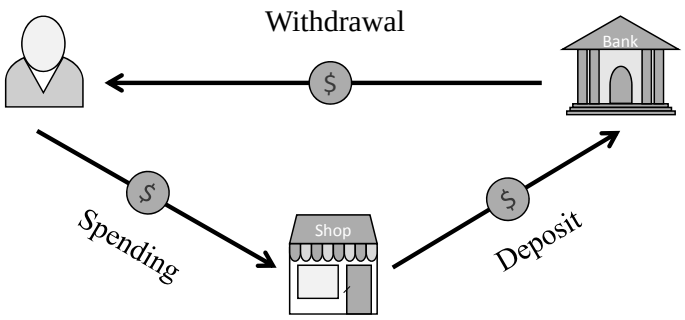


Figure 2.1. E-cash concept — A user receives electronic coins from the bank, which is the only entity able to generate coins, during a process called withdrawal. He will later use these coins to pay at a shop during a process called spending. The shop can deposit coins it received from users over a period of time to its bank account.

- Unforgeability: Only the signer can generate blind signatures. Even though a prover has received a multitude of blind signatures he should not be able to generate blind signatures on his own.

These two properties are extremely important for the e-cash scenario, as they ensure (i) that nobody but the bank, which is in possession of the secret blind signature key, can generate electronic coins and (ii) that later on a bank cannot identify whom it had given a particular coin to as it has not seen the data it has been signing. Yet, as with any digital signature scheme, anyone who is in possession of the public key of the bank can verify the validity of the signature, *i.e.* the validity of a coin, offline, without having to query the bank.

A user spends a coin at a shop. A mechanism is included that only allows someone who knows the secret key of the user to spend a coin. In a correctly set up system this should only be a user himself. The shop accepts the payment as long as the signature is valid; it cannot check whether a coin had been spent before.

At a later point in time the shop deposits the coins that it received from users to its bank account. The bank has a database collecting all used coins and checks, whether a deposited coin had been deposited before. If so it further checks whether a shop

deposited the same coin twice or whether a user double-spent the coin. Therefore a user's ID is blindly encoded in a coin in a way that, as long as the user uses the system correctly, *i.e.* spends each coin only once, his identity remains hidden. However, if he uses the same coin multiple times, his ID can be revealed from the transaction data of the coin. This mechanism is called double-spending detection.

2.1.1 E-cash from (Blind) Digital Signatures

Blind digital signatures are based on digital signatures, which were introduced by Diffie and Hellman in 1976 [32]. Digital signatures fulfill the same properties as written signatures: anyone can easily verify a signature's authenticity and it is infeasible for an attacker to forge it, *i.e.* generate a valid signature on his own. Signatures can be constructed based on public-key cryptosystems. A signer with private key $\mathsf{sk}_{\mathcal{S}}$ and public key $\mathsf{pk}_{\mathcal{S}}$ can generate a signature $\sigma(\mathsf{msg})$ on the message msg by decrypting the message with her secret key $\sigma(\mathsf{msg}) = \mathit{decrypt}_{\mathsf{sk}_{\mathcal{S}}}(\mathsf{msg})$. Using the signer's public key $\mathsf{pk}_{\mathcal{S}}$, anyone can verify the signature on msg by checking, whether

$$\mathsf{msg} \stackrel{?}{=} \mathit{encrypt}_{\mathsf{pk}_{\mathcal{S}}}(\sigma(\mathsf{msg})). \tag{2.1}$$

As described in [25] digital signatures can be used to construct a secure online electronic cash system, which, however does not preserve a user's privacy. To withdraw a coin, $\mathcal{U}$ generates a random serial number $\mathtt{SN}$, signs it with her private key, and sends $\mathtt{SN}$ as well as the signature $\sigma_{\mathcal{U}}(\mathtt{SN})$ to $\mathcal{B}$. $\mathcal{B}$ validates the signature $\sigma_{\mathcal{U}}(\mathtt{SN})$ using $\mathcal{U}$'s public key, signs $\mathtt{SN}$ using its own private key and debits $\mathcal{U}$'s account. It returns the signed coin $\sigma_{\mathcal{B}}(\mathtt{SN})$ to the user. Data could be communicated over an encrypted channel such that an attacker $\mathcal{A}$ cannot steal a coin by intercepting the communication. However, doing so would still not account for the fact that $\mathcal{A}$ could spend $\mathcal{U}$'s coins, if he could retrieve them from the device in which they are stored. A mechanism would have to be used, which only allows $\mathcal{U}$ to spend withdrawn coins.

This could be done by not removing $\mathcal{U}$'s signature from the coin and adding the bank's signature to it. $\mathcal{S}$ would then require $\mathcal{U}$ to authenticate himself, i.e. to prove knowledge of his secret key, during spending and then check both signatures. To execute a payment $\mathcal{U}$ transfers the required amount of coins to $\mathcal{S}$. After verifying the signatures $\mathcal{S}$ sends the coins to $\mathcal{B}$. $\mathcal{B}$ verifies the signatures and checks whether the coins had already been spent before. If they had not been spent it sends a signed deposit receipt to $\mathcal{S}$ and marks the coins as deposited. $\mathcal{S}$ returns goods to $\mathcal{U}$ together with a digitally signed receipt.

In this system $\mathcal{U}$ cannot generate coins without $\mathcal{B}$ and cannot use a coin multiple times. Additionally she cannot deny the withdrawal of coins from her bank account, as she provided a signature on a generated serial number. $\mathcal{B}$ cannot deny the generation of coins since it digitally signed them. Also $\mathcal{B}$ cannot deny that it accepted coins from $\mathcal{S}$, since it handed a digitally signed deposit receipt to $\mathcal{S}$. $\mathcal{S}$ cannot deny that it received a payment as $\mathcal{U}$ received a digitally signed receipt. However, as mentioned above, this system does not provide customer privacy.

Blind signatures allow the introduction of customer privacy into the electronic cash scheme presented above. The concept of blind signatures was introduced by David Chaum in 1982 [24]. The idea is that before sending a generated serial number $\mathtt{SN}$ to the bank, $\mathcal{U}$ blinds it using a random challenge $c(\mathtt{SN})$ such that $\mathcal{B}$ cannot learn anything about $\mathtt{SN}$. $\mathcal{B}$ signs this blinded serial number $\hat{\sigma}(c(\mathtt{SN}))$, debits $\mathcal{U}$'s account, and sends the signature $\hat{\sigma}(c(\mathtt{SN}))$ back to the user. $\mathcal{U}$ strips off the blinding factor $c^{-1}(\hat{\sigma}(c(\mathtt{SN}))) = \sigma(\mathtt{SN})$ and checks whether the resulting signature is a valid signature on $\mathtt{SN}$ ($\mathtt{SN} \stackrel{?}{=} \sigma^{-1}(\sigma(\mathtt{SN}))$). He can use this coin to pay at a shop. After receiving a coin $\mathcal{S}$ checks its validity and sends $\sigma(\mathtt{SN})$ to $\mathcal{B}$. Note that $\mathcal{B}$ has neither seen $\mathtt{SN}$ nor σ before and hence cannot link the deposited coin to one that it gave out to a particular user. $\mathcal{B}$ also verifies the validity of the coin and further queries its database to check

whether this coin has been spent before. If not, it accepts the payment, deposits the money into $\mathcal{S}$'s account, and informs $\mathcal{S}$ that it accepted the payment.

This blind signature concept can serve to construct privacy-preserving *online* payment systems. Yet, it is often desirable to further support offline verification of payments. This can be achieved by forcing $\mathcal{U}$ to reveal particular partial information about himself during a payment in a way that the provided partial information does not reveal anything about $\mathcal{U}$ unless he uses the same coin again, in which case the two chunks of information can be combined to reveal his identity. In this scenario it only has to be verified, whether a coin is valid, during a payment. Checking whether $\mathcal{U}$ spent a coin multiple times, can be done at a later point in time. Since $\mathcal{U}$'s identity is revealed, if he misused the system, he can be penalized after the fact. This double-spending detection concept was proposed as part of the concept of one-show blind signatures by Chaum *et al.* in [26].

2.1.2 E-Cash Protocols

Based on one-show credentials e-cash schemes can be constructed, which include the following protocols:

- Setup, in which system parameters are generated and the bank generates a private and public key pair.
- Registration, in which a user registers at the bank. After authentication, e.g. by means of a passport or credit card, the user generates a private and public key pair and proves possession of his private key to the bank. The bank stores the user information together with the user's public key, to be later used for identification purposes in case of fraud.
- Withdrawal, in which a user receives electronic coins (the currency in an electronic cash scheme) from the bank, which is the only entity able to generate

these coins. The user's bank account is debited the amount of money he received as electronic coins.

- Spending, in which a user spends electronic coins to a shop in exchange for goods. This process happens offline, *i.e.* a shop checks by cryptographic means whether she received a valid coin from the user. She cannot check whether the user has already spent the coin before.

- Deposit, in which a shop deposits coins she received from users to her bank account. During this process the bank checks whether it receives a valid coin from the shop and adds its value to the shop's bank account. Later it checks whether this coin had already been deposited before. If so, it executes the double-spending detection protocol.

- Double-Spending Detection, in which the bank checks whether the shop is trying to double-deposit a coin, or whether a user has double-spent a coin. If it detects that a user has double-spent a coin, it reveals his identity from the coin.

2.2 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is the most efficient established asymmetric cryptographic scheme, which has been independently proposed by Neal Koblitz in 1987 and by Victor Miller in 1986 [76]. Its efficiency results from the fact that, in contrast to other public-key cryptosystems, the run-time of known attacks on ECC-based cryptosystems grows exponentially with the bit length of the curve. Hence, when compared to other asymmetric cryptosystems, the same security level can be achieved with much shorter key lengths [84]. As such a 1024-bit discrete logarithm (DL) based cryptosystem reaches a comparable security level to a 160-bit ECC-based cryptosystem. While an operation on an elliptic curve requires multiple multiplications, squarings and additions in the underlying field, the reduction in field size, and

thus the lower arithmetic complexity of operations in the underlying finite field, can result in a much better over-all performance, especially on constrained microcontroller architectures. For example, using the operand-scanning algorithm, 400 single-word multiplications are required to multiply two 160-bit field elements on an 8-bit microcontroller. In comparison to this 16 384 word multiplications are required for the multiplication of two 1024-bit words on the same platform, again using the operand scanning algorithm. Thus, executing for example ten multiplications on a 160-bit field requires one forth of the number of single-word multiplications compared to one multiplication on a 1024-bit field. Another advantage is that due to the shorter key lengths less data has to be communicated between protocol partners. This is particularly beneficial in areas, where the communication bandwidth is limited. Those facts make the use of ECC desirable on platforms that are constrained in power and computational performance, which is often the case for user devices in payment systems.

We use elliptic curves that are defined over prime fields $\mathbb{Z}_p$. A prime field $\mathbb{Z}_p$ denotes the set of integers $\{0, 1, \ldots, p-1\}$ in which addition and multiplication are performed modulo a prime p. An elliptic curve is defined by the Weierstrass equation

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2.2}$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{Z}_p^*$, such that $-d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \neq 0$, with $d_2 = a_1^2+4a_2$, $d_4 = 2a_4+a_1a_3$, $d_6 = a_3^2+4a_6$ and $d_8 = a_1^2a_6+4a_2a_6-a_1a_3a_4+a_2a_3^2-a_4^2$ [46]. A pair (x, y) where $x, y \in \mathbb{Z}_p$, which fulfills Equation 2.2, is called a point on the elliptic curve.

The operation defined on an elliptic curve is the point addition. Looking at an elliptic curve defined over the set of real numbers: To add the points P and Q geometrically one draws a line through P and Q. The point, where this line intersects E a third time is mirrored at the x-axis to reach the point $P + Q$. A special case

of the point addition is the point doubling, in which a point is added to itself. To double the point P geometrically, one draws the tangent line to the elliptic curve at the point P. The point, where this line intersects the elliptic curve E a second time is mirrored at the x-axis, to reach the point $2P$ [46]. The calculation rule to execute an addition, doubling and the computation of the negative point on an elliptic curve depends on the curve type and is summarized in a so-called group law.

Additionally, a point at infinity $\mathcal{O}$ is defined. Including this point at infinity the set of points of an elliptic curve is denoted as $E(\mathbb{Z}_p)$. Together with the addition operation $E(\mathbb{Z}_p)$ forms an abelian group G, which has the following properties:

- Closure: $\forall\, a, b \in G, a \circ b \in G$
- Associativity: $a \circ (b \circ c) = (a \circ b) \circ c \;\forall\, a, b, c \in G$
- Existence of an identity: $\exists\, e \in G$ such that $a \circ e = e \circ a = a \;\forall\, a \in G$
- Existence of inverses: $\forall\, a \in G \;\exists\, b \in G$ such that $a \circ b = b \circ a = e$
- Commutativity: $a \circ b = b \circ a \;\forall\, a, b \in G$.

If a protocol is designed for abelian groups, it requires at most the above mentioned group properties and can thus be based on ECC.

Repeated execution of the point addition is called scalar multiplication $Q = [k]P$, where the point $P = (x, y)$ is added to itself $k - 1$ times. The scalar multiplication is the core operation of ECC, equivalent to the exponentiation in DL-based systems. Hence, what is denoted as exponentiation in G in the protocols listed in this dissertation is executed as scalar multiplication on an elliptic curve E. Similarly, what is denoted as a multiplication in G is executed as a point addition on E.

$E(\mathbb{Z}_p)$ forms a finite group where the number N of elements is called the order of $E(\mathbb{Z}_p)$. For an element $P \in E(\mathbb{Z}_p)$ the smallest integer t such that $[t]P = \mathcal{O}$ is called

the order of P. t is always a divisor of N, and if it equals N then $E(\mathbb{Z}_p)$ is called cyclic group and P is called generator of $E(\mathbb{Z}_p)$ [46].

The security of ECC-based cryptosystems and thus the security of e-cash schemes that are based on ECC relies on the hardness of computing the discrete logarithm in $E(\mathbb{Z}_p^*)$. Given two points $P, Q \in E(\mathbb{Z}_p)$, which are related through the equation $Q = [k]P$, the elliptic curve discrete logarithm problem (ECDLP) is defined as the problem of finding the scalar k, such that $Q = [k]P$ holds.

As mentioned above the core operation in ECC-based cryptosystems is the scalar multiplication on an elliptic curve. The scalar multiplication is also the most time consuming operation in an ECC cryptosystem, requiring a multitude of point additions. In order to reduce the number of point additions required to execute a scalar multiplication, one writes the scalar in binary representation $k = \sum_{i=0}^{n-1} k_i 2^i$. Then the scalar multiplication can be executed as

$$Q = [k]P = \sum_{i=0}^{n-1} [k_i 2^i]P = [k_0]P + 2([k_1]P + 2([k_2] + \cdots + 2([k_{n-2}]P + 2([k_{n-1}]P)) \dots)),$$

which can be formulated as Algorithm 1.

Algorithm 1: Left-to-right double-and-add algorithm for scalar multiplication on an elliptic curve [46].

Input : $k = (k_{n-1}, \dots, k_1, k_0), P$.
Output: $Q = (x_Q, y_Q) = k * P$.

```
Q ← O
for i = n − 1 downto 0 do
    Q = 2Q
    if k_i = 1 then
        Q = Q + P
    end
end
```

The issue with this algorithm is its timing variance. While a point doubling is executed in every iteration, a point addition is only executed, if the scalar bit is one.

The processing of this algorithm thus directly leaks information about the processed scalar, which can be exploited using so-called timing attacks. An algorithm avoiding this issue is the Montgomery powering ladder [71]. The Montgomery powering ladder (Algorithm 2) executes a point addition and a point doubling in each iteration, making it a regular algorithm for the scalar multiplication.

Algorithm 2: Montgomery Powering Ladder [71].

Input : $k = (k_{n-1}, \ldots, k_1, k_0), P$.
Output: $Q = (x_Q, y_Q) = k * P$.

$R_0 \leftarrow \mathcal{O}; R_1 \leftarrow P$
for $i = i = n - 2$ *downto 0* **do**
  if $k_i = 1$ **then**
    $R_0 \leftarrow R_0 + R_1,\ R_1 \leftarrow 2R_1$
  end
  else
    $R_1 \leftarrow R_1 + R_0,\ R_0 \leftarrow 2R_0$
  end
end

CHAPTER 3

EFFICIENT IMPLEMENTATION OF HIGH-SECURITY ELLIPTIC CURVE CRYPTOGRAPHY ON MSP430 MICROCONTROLLERS

In this chapter we present work, which has been published in [33] and [49]. The results arose from collaborative work with Michael Hutter, Amir Moradi, and Peter Schwabe. Results that were not developed by the author of this dissertation will be clearly marked in the respective sections.

For the remaining part of this dissertation we present implementations of e-cash schemes targeting the transportation domain, where a security level comparable to 80-bit symmetric security is sometimes sufficient. However, there are applications, where a high level of security is required even on extremely constrained devices. We thus investigate the performance of high-security public-key cryptography on hardware that could be used for such applications.

This chapter describes our implementation of Curve25519 on MSP430X microcontrollers. Texas Instruments designed MSP430 microcontrollers to target low-power applications (they can be operated at voltages of 1.8 to 3.3 Volts) and advertises their use in security-critical applications as for example the domain of medical devices [57]. Newer devices of the MSP430 family have AES hardware accelerators that support 256-bit AES. Yet, many security services that are desirable for wireless communication, especially in the domain of medical devices, rely on public-key cryptography. This naturally raises the question about the performance of public-key cryptography on MSP430 microcontrollers.

Bernstein introduced the Curve25519 elliptic-curve Diffie-Hellman (ECDH) key exchange protocol in 2006 [13]. It uses the Montgomery form of this curve, which is defined over a 255-bit prime field and achieves a 128-bit security level. Montgomery curves are known to allow for very efficient x-coordinate-only variable-base-point single-scalar multiplication, which makes their use highly attractive for ECDH key-exchange schemes.

We present a full implementation of the Curve25519 Diffie-Hellman key-exchange scheme, recently named X25519, on MSP430X microcontrollers. We differentiate those MSP430Xs with a 16-bit and those with a 32-bit hardware multiplier and developed our code for both platforms. As all previous implementations of Curve25519 we use projective coordinates for the elliptic-curve point representation. The main performance bottleneck of the variable-base-point single-scalar multiplication are then modular multiplications in the underlying prime field. We hence put our focus on optimizing the modular multiplication on the MSP430X architecture and give a comprehensive evaluation of different implementation techniques for MSP430X microcontrollers. We evaluate our implementation by executing it on Texas Instrument's MSP-EXP430FR5969 LaunchPad Evaluation Kit [60], which hosts an MSP430FR5969. The MSP430FR5969 has a MSP430X CPU, 64 kB of non-volatile memory, 2 kB SRAM and a 32-bit memory-mapped hardware multiplier [58]. We would like to point out that this microcontroller is built into the WISP 5.0 UHF computational RFID tag[1], a device that operates based on harvested power from the RF field, which indicates its ultra-low power nature. With a price of a few dollars, this microcontroller is a suitable target for wireless sensor and medical applications.

We use the Montgomery powering ladder [71] to implement the scalar multiplication on the elliptic curve, since this is a highly regular algorithm, making the

[1] `http://wisp.wikispaces.com/WISP%205.0`

executed computation independent of the scalar. Our software completely avoids input-dependent loads and branches, thus executing in constant time and combating timing attacks such as [4] or [102].

3.1 Related Work

Several implementations of ECC based on Curve25519 have been presented in the literature. Various papers describe implementations of Curve25519 on powerful processors [13, 29, 12], a recent publication describes an implementation on reconfigurable hardware [91], and [61, 15] describes an implementation, which fits into 18 tweets. Only one prior implementation shows performance results of Curve25519 on constrained devices, namely the implementation for 8-bit AVR microcontrollers by Hutter *et al.* presented in [53], while no previous work has yet shown implementation results of Curve25519 for 16-bit microcontrollers.

A plethora of literature exists on the implementation of ECC based on other elliptic curves on MSP430 microcontrollers. One of the first publications of ECC on the MSP430 architecture is by Guajardo *et al.* in 2001 [44]. They presented an implementation of a 128-bit elliptic curve and show that a scalar multiplication can be performed within 3.4 million clock cycles. In 2007, Scott *et al.* presented optimizations for finite-field multiplications underlying ECC [93]. Their 160-bit (hybrid) multiplication method requires 1 746 cycles. In 2009, Szczechowiak *et al.* presented pairing-based cryptography on the MSP430 [100]. Similar results have been reported by Gouvêa *et al.* in the same year [40]. They reported new speed records for 160-bit and 256-bit finite-field multiplications on the MSP430 requiring 1 586 and 3 597 cycles, respectively. They further presented an implementation of a 256-bit elliptic curve random point scalar multiplication needing 20.4 million clock cycles, when relying on the Montgomery powering ladder. In 2011, Wenger *et al.* compared ECC scalar multiplications on various 16-bit microcontrollers [106]. Their Montgomery-ladder based

scalar multiplication needs 23.9 million cycles using a NIST P-256 elliptic curve. Also in 2011, Pendl *et al.* presented the first ECC implementation running on the WISP UHF RFID tag [78]. Their 192-bit NIST curve implementation achieves an execution time of around 10 million clock cycles. They also reported first multi-precision multiplication results for 192 bits requiring 2 581 cycles. In 2012, Gouvêa *et al.* reported new speed records for different MSP430 architectures [41]. They improved their results from [40], namely, for the MSP architecture (with a 16-bit multiplier) their 160-bit and 256-bit finite-field multiplication implementations require 1 565 and 3 563 cycles, respectively.

Under the common assumption that the execution time of ECC grows approximately as a cubic function of the field size, our software outperforms all presented constant-time ECC implementations on MSP430X microcontrollers in speed.

3.2 The MSP430X Microcontroller Architecture

The MSP430X has a 16-bit RISC CPU with 27 core instructions and 24 emulated instructions. The CPU has 16 16-bit registers. Of those only `R4` to `R15` are freely usable working registers, and `R0` to `R3` are special-purpose registers (program counter, stack pointer, status register, and constant generator). All instructions execute in one cycle, if they operate on contents that are stored in CPU registers. However, the overall execution time for an instruction depends on the instruction format and addressing mode. The CPU features 7 addressing modes. While indirect auto-increment mode leads to a shorter instruction execution time compared to indexed mode, only indexed mode can be used to store results back to RAM.

We consider MSP430X microcontrollers, which feature a memory-mapped hardware multiplier that works in parallel to the CPU. Four types of multiplications, namely signed and unsigned multiply as well as signed and unsigned multiply-and-accumulate are supported. The multiplier registers are peripheral registers, which

have to be loaded with CPU instructions. Which type of multiplication to be executed is selected by loading the multiplicand to one of the four registers (register pairs for 32-bit multipliers) `MPY`, `MPYS`, `MAC`, and `MACS` which refer to unsigned multiply, signed multiply, unsigned multiply-and-accumulate, and signed multiply-and-accumulate respectively. The multiplicand has to be loaded first. As soon as the multiplier is loaded to the register `OP2`, the desired multiplication is executed. The hardware multiplier stores the result in two (in case of 16-bit multipliers) or four (in case of 32-bit multipliers) 16-bit registers. Further a `SUMEXT` register indicates for the multiply-and-accumulate instruction, whether a multiplication has produced a carry bit. However, it is not possible to accumulate carries in `SUMEXT`. The time required for the execution of a multiplication is determined by the time that it takes to load operands to and store results from the peripheral multiplier registers. The result of a 16×16-bit multiplication is available in 3 clock cycles on both types of MSP430X devices, those that have a 32-bit hardware multiplier as well as those that have a 16-bit hardware multiplier (cf. [58] and [56]). Thus, our measurement results can be generalized to other microcontrollers from the MSP430X family.

The MSP430FR5969 (the target under consideration) belongs to a new MSP430 series featuring Ferroelectric Random Access Memory (FRAM) technology for non-volatile memory. This technology has two benefits compared to flash memory. It leads to a reduced power consumption during memory writes and further increases the number of possible write operations. However, as a drawback, while the maximum operating frequency of the MSP430FR5969 is 16 MHz, FRAM can only be accessed at 8 MHz. Hence, wait cycles have to be introduced, when operating the MSP430FR5969 at frequencies higher than 8 MHz. For all cycle counts that we present in this chapter, where we do not specify the operating frequency of the MCU, we assume a core clock frequency of 8 MHz. Increasing this frequency on the MSP430FR5969 would incur a

penalty resulting from those introduced wait cycles. Note, that this is not the case for MSP430 devices that use flash technology for non-volatile memory.

3.3 Review of Curve25519

Curve25519 is an elliptic curve in Montgomery form. This curve has been carefully chosen to provide very high performance for Diffie-Hellman key-exchange at the 128-bit security level. It is defined by the equation $y^2 = x^3 + 486662x^2 + x$ over the prime field $\mathbb{Z}_{2^{255}-19}$. For details about the choice of curve and security see [13].

The key-exchange scheme computes a 32-byte shared secret Q_x from a 32-byte secret key n and a 32-byte public key P_x. Here Q_x and P_x are x-coordinates of points on the elliptic curve. At its core, the Curve25519 Diffie-Hellman key-exchange scheme executes a variable-base-point single-scalar multiplication on the elliptic curve, multiplying the public key P_x with the secret scalar n, to obtain the shared secret Q_x. Special conditions are given for the secret scalar n, namely that the 3 least significant bits and the most significant bit are set to zero, and the second-most significant bit is set to one [14].

We follow the suggestions of [13] for implementing the variable-base-point single-scalar multiplication on the elliptic curve. We use the Montgomery powering ladder [71] of 255 ladder steps. Each ladder step computes a differential point addition and a point doubling. Starting with the points R_1 and R_2, in each ladder step either R_2 is added to R_1 ($R_1 \leftarrow R_1 + R_2$) and then R_2 is doubled ($R_2 \leftarrow 2 \cdot R_2$), or R_1 is added to R_2 ($R_2 \leftarrow R_2 + R_1$) and then R_1 is doubled ($R_1 \leftarrow 2 \cdot R_1$). To avoid conditional load addresses that can lead to cache-timing attacks, we execute the same operations ($R_1 \leftarrow R_1 + R_2$ and $R_2 \leftarrow 2 \cdot R_2$) in each iteration, and conditionally swap the contents of R_1 and R_2 depending on the scalar bit b.

Note that for the conditional swap we do not use branch instructions. Instead, this operation is implemented as follows: An unsigned variable $\hat{b}$ is cleared. Then b is

subtracted from $\hat{b}$ leading to $\hat{b}$ being `0` or `0xffff`, depending on whether b is `0` or `1`. To swap the contents of x and y, an auxiliary variable is used to store $t_{swp} = x \oplus y$. t_{swp} is ANDed with the value stored in $\hat{b}$, resulting in $t_{swp} = x \oplus y$ for $b =$ `1` and $t_{swp} =$ `0` otherwise. Further t_{swp} is XORed with x and y leading to either the original values being stored in x and y for the case $b =$ `0`, or the swapped values for the case $b =$ `1`. Together with the constant-time field arithmetic we thus obtain a fully timing-attack protected constant-time implementation.

In [71] Montgomery presented x-coordinate-only doubling and differential-addition formulas for points on an elliptic curve defined by an equation of the form $By^2 = x^3 + Ax^2 + x$. He showed the correctness of those formulas, which rely on the standard-projective-coordinate representation of points, for the case of inputs not being equal to the point at infinity. In the standard-projective coordinate system, a point is represented by three coordinates, namely (X, Y, Z). This point corresponds to the point $(X/Z, Y/Z)$ in affine coordinates. In [13] Bernstein extended the proof of correctness of these x-coordinate-only doubling and differential-addition formulas to the case of an input being equal to the point at infinity. Using these formulas, a differential addition of two points requires 4 multiplications and 2 squarings. Point doubling requires 2 multiplications, 2 squarings, and one multiplication by the constant $(486662 + 2)/4 = 121666$. The differential-addition formula requires as input the difference of the input points. If the Z-coordinate of this difference point is one, the addition formula can be improved to require only 3 multiplications and 2 squarings. Algorithm 3 summarizes the x-coordinate-only variable-base-point single-scalar point multiplication on Curve25519 requiring 255 differential additions and doublings (ladder steps), 255 conditional swaps, and one inversion at the end to transform the result back to affine coordinates [13, 29].

Algorithm 3: X-coordinate-only variable base-point single-scalar point multiplication on Curve25519 based on the Montgomery powering ladder [13, 29].

Input : Scalar $n \in \mathbb{Z}$, P_x, x-coordinate of point P.
Output: Q_x, x-coordinate of point $Q \leftarrow n \cdot P$.

$X_1 \leftarrow P_x; X_2 \leftarrow 1; Z_2 \leftarrow 0; X_3 \leftarrow P_x; Z_3 \leftarrow 1$
for $i = 254$ *downto 0* **do**
  if $n_i \neq n_{i-1}$ **then**
    swap(X_2, X_3) `/* This conditional swapping is implemented */`
    swap(Z_2, Z_3) `/* in constant time. */`
  end
  $t_1 \leftarrow X_2 + Z_2$
  $t_2 \leftarrow X_2 - Z_2$
  $t_3 \leftarrow X_3 + Z_3$
  $t_4 \leftarrow X_3 - Z_3$
  $t_6 \leftarrow t_1^2$
  $t_7 \leftarrow t_2^2$
  $t_5 \leftarrow t_6 - t_7$
  $t_8 \leftarrow t_4 \cdot t_1$
  $t_9 \leftarrow t_3 \cdot t_2$
  $X_3 \leftarrow (t_8 + t_9)^2$
  $Z_3 \leftarrow X_1(t_8 - t_9)^2$
  $X_2 \leftarrow t_6 \cdot t_7$
  $Z_2 \leftarrow t_5(t_7 + 121666t_5)^2$
end
if $n_0 == 1$ **then**
  swap(X_2, X_3) `/* This conditional swapping is implemented */`
  swap(Z_2, Z_3) `/* in constant time. */`
end
$Z_2 \leftarrow 1/Z_2$
return $(X_2 \cdot Z_2)$

3.4 Implementation of Modular Arithmetic in $\mathbb{Z}_{2^{255}-19}$

Algorithm 3 relies on addition, subtraction, multiplication, squaring and multiplication with a constant in the underlying prime field. Among those, multiplication is the most time consuming operation and its careful implementation is thus of utter importance. Many techniques have been proposed to improve the performance of multi-precision multiplication implementations, especially for constrained devices. In

the following we describe the techniques that we implemented and compared for the MSP430X architecture in detail. To have a fair comparison, all methods were implemented in assembly and were fully unrolled. We further present our implementation of the other modular arithmetic operations.

3.4.1 Representation of Big Integers

We use an unsigned radix-2^{16} representation for the product-scanning, operand-caching [55] and the Karatsuba multiplication [54, 62], and a signed radix-$2^{\lceil 255/26 \rceil}$ representation for the carry-save implementation. In unsigned radix-2^{16} representation an n-bit integer A is represented as an array of $m = \lceil n/16 \rceil$ words in little-endian order as $(a_0, a_1, \ldots a_{m-1})$ such that $A = \sum_{i=0}^{m-1} a_i 2^{16i}$, where $a_i \in \{0, \ldots, 2^{16} - 1\}$. An element f in $\mathbb{Z}_{2^{255}-19}$ is thus represented as $f = \sum_{i=0}^{15} f_i 2^{16i} = (f_0, f_1, \ldots f_{15})$. In radix-$2^{\lceil 255/26 \rceil}$ representation an n-bit integer B is represented as an array of $\ell = \lceil 26n/255 \rceil$ 16-bit words in little-endian order as $(b_0, b_1, \ldots b_{\ell-1})$ such that $B = \sum_{j=0}^{\ell-1} b_j 2^{\lceil 255j/26 \rceil}$, where $b_j \in \{-2^{15}, \ldots, 2^{15} - 1\}$. Hence, in radix-$2^{\lceil 255/26 \rceil}$ representation an element in $\mathbb{Z}_{2^{255}-19}$ is represented using 26 16-bit words. Since inputs and outputs to the scalar multiplication on Curve25519 are 32-byte arrays, conversions to and from the used representations are only executed at the beginning and the end of the entire scalar multiplication.

3.4.2 Multiplication Using Carry-Save Representation

This implementation follows the fast arithmetic implementation presented in [13]. A multi-precision integer is represented using the signed radix-$2^{\lceil 255/26 \rceil}$ representation:

$$B = b_0 + b_1 2^{10} + b_2 2^{20} + b_3 2^{30} + b_4 2^{40} + b_5 2^{50} + b_6 2^{59} + b_7 2^{69} + b_8 2^{79} + \cdots + b_{25} 2^{246}.$$

A benefit of this representation is that an addition or subtraction can be executed without having to consider carry bits. It only requires pairwise addition or subtrac-

tion of the respective coefficients, as long as the results of coefficient additions or subtractions do not exceed the word-length.

h_0	h_1	h_2	h_3	h_4	h_5	h_6	h_7	…
$f_0\,g_0$	$f_1\,g_0$	$f_2\,g_0$	$f_3\,g_0$	$f_4\,g_0$	$f_5\,g_0$	$f_6\,g_0$	$f_7\,g_0$	…
$38\,f_{24}\,g_2$	$f_0\,g_1$	$f_1\,g_1$	$f_2\,g_1$	$f_3\,g_1$	$f_4\,g_1$	$2\,f_5\,g_1$	$f_6\,g_1$	…
$38\,f_{23}\,g_3$	$38\,f_{25}\,g_2$	$f_0\,g_2$	$f_1\,g_2$	$f_2\,g_2$	$f_3\,g_2$	$2\,f_4\,g_2$	$2\,f_5\,g_2$	…
$38\,f_{22}\,g_4$	$38\,f_{24}\,g_3$	$38\,f_{25}\,g_3$	$f_0\,g_3$	$f_1\,g_3$	$f_2\,g_3$	$2\,f_3\,g_3$	$2\,f_4\,g_3$	…
$38\,f_{21}\,g_5$	$38\,f_{23}\,g_4$	$38\,f_{24}\,g_4$	$38\,f_{25}\,g_4$	$f_0\,g_4$	$f_1\,g_4$	$2\,f_2\,g_4$	$2\,f_3\,g_4$	…
$38\,f_{20}\,g_6$	$38\,f_{22}\,g_5$	$38\,f_{23}\,g_5$	$38\,f_{24}\,g_5$	$38\,f_{25}\,g_5$	$f_0\,g_5$	$2\,f_1\,g_5$	$2\,f_2\,g_5$	…
$38\,f_{19}\,g_7$	$19\,f_{21}\,g_6$	$19\,f_{22}\,g_6$	$19\,f_{23}\,g_6$	$19\,f_{24}\,g_6$	$19\,f_{25}\,g_6$	$f_0\,g_6$	$f_1\,g_6$	…
$38\,f_{18}\,g_8$	$38\,f_{20}\,g_7$	$19\,f_{21}\,g_7$	$19\,f_{22}\,g_7$	$19\,f_{23}\,g_7$	$19\,f_{24}\,g_7$	$19\,f_{25}\,g_7$	$f_0\,g_7$	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	…

Figure 3.1. Visualisation of computing the first coefficients of the array $h \leftarrow f \times g \mod 2^{255} - 19$ using the carry-save technique. The dark grey block originates from reducing the double-sized array, which is the result of multiplying f with g, modulo $2^{255} - 19$.

Figure 3.1 exemplary presents the steps executed to compute the first 8 coefficients h_i of the array $h \leftarrow f \times g \mod 2^{255} - 19$. After transforming an integer to radix-$2^{\lceil 255/26 \rceil}$ representation, each coefficient b_i of B is within the interval $(-2^9, 2^9)$ or $(-2^{10}, 2^{10})$. We precompute $2f$ and $19g$ to easily realize constant multiplication with factors 2, 19, and 38. We then use the product-scanning technique to compute the coefficients h_i, interleaving the multiplication with the reduction, *i.e.* we compute a coefficient and reduce it right away. For the computation of each h_i, 26 products of coefficients have to be added.

This type of implementation has two disadvantages on the MSP430X architecture. First of all the MSP430X CPU has very few general-purpose registers, while the inputs have to be loaded from four different arrays $f, g, 2f$ and $19g$. This makes holding inputs in registers difficult, as many different operands have to be loaded for computation of the various coefficients. Further, while we use indirect auto-increment mode to access g and $19g$, there is no indirect auto-decrement mode on the MSP430X and we need to access the other inputs using the costly indexed mode. The other

disadvantage is the highly complex reduction of a coefficient, requiring several shift operations, which are expensive on MSP430X devices.

Since we could not achieve good performance results with this type of implementation, we tried to speed things up relying on the refined Karatsuba formulas presented in [11]. A problem occurs when trying to add the low and the high part of B in signed radix-$2^{\lceil 255/26 \rceil}$ representation. For example computing the coefficient of 2^{40} cannot be executed by adding b_4 and b_{16} as b_{16} would be the coefficient of 2^{39}. Our solution to this was to represent elements using signed radix-$2^{\lceil 256/26 \rceil}$ representation and rely on computations modulo $2^{256} - 38$. Yet still, the disadvantages of this type of implementation on the MSP430X architecture dominate the advantages.

3.4.3 Operand-Caching Multiplication

Operand-caching was proposed by Hutter *et al.* in 2011 [55]. The idea of this method is to reduce the number of load instructions by organizing the operations in a way which allows using the same operands for multiple computations.

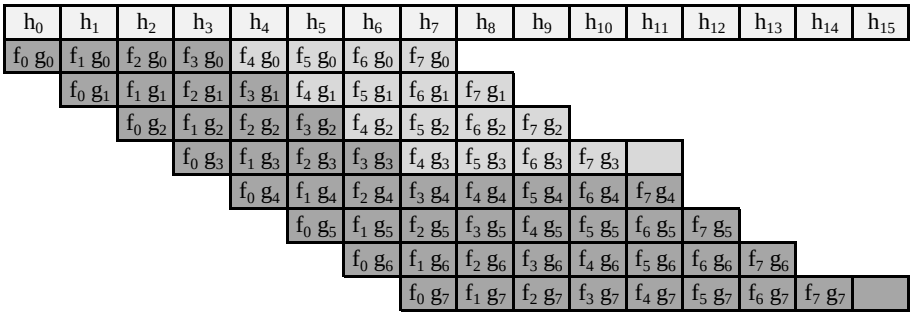


Figure 3.2. Toy-size example of the operand-caching multiplication, when 8 general-purpose registers are available to hold operands. To compute the coefficients of the array $h \leftarrow f \times g$ one first computes the light grey block keeping all required input operands in general-purpose registers. Thereafter the dark grey area is computed.

Figure 3.2 shows a toy-size example of the operand-caching multiplication. Here the execution of computations is divided into the light grey block, which is computed

first, and the dark grey area, which is computed second. The empty dark grey and light grey boxes represent space that is required for carry-bits.

As we have 8 general-purpose registers available for storing operands during the execution of the multiplication, we chose the row size to be 4. Since each input array has 16 elements, $16/4 = 4$ rows have to be computed. One peculiarity of the MSP430 hardware multiplier greatly improves the performance of the operand-caching multiplication. In order to execute a multiply-and-accumulate operation, one (two in the case of 32-bit multipliers) operand is loaded to the hardware multiplier's `MAC` register and the other to the `OP2` register. Loading `OP2` starts the execution of the multiply-and-accumulate instruction. `MAC` does not have to be loaded again, before another execution of the multiply-and-accumulate instruction, if the contents of this register do not have to be changed. By ordering the multiplication instructions, required for the computation of the coefficients of the resulting array, in a way that input operands that have been loaded last to `MAC` during the computation of one coefficient are processed first during the computation of the subsequent coefficient, we can save on load operations to the multiplier registers. For example, if for the computation of h_1, as the final step f_0 was loaded to `MAC` and g_1 to `OP2`, then we start the computation of h_2 by loading g_2 to `OP2`.

We first multiply both inputs f and g, resulting in a double-sized array and then reduce this result modulo $2^{255} - 19$. Since reduction modulo $2^{255} - 19$ requires bit shifts, we chose to reduce intermediate results modulo $2^{256} - 38$ and only reduce the final result of the scalar multiplication modulo $2^{255} - 19$. We implemented two versions of operand-caching multiplication, one making use of the 32-bit hardware multiplier (in the following called 32-bit operand-caching) and the other only loading 16-bit inputs to the multiplier (in the following called 16-bit operand-caching).

3.4.4 Product-Scanning Multiplication

To the best of the authors knowledge, the fastest previously reported result for 256-bit multiplication on MSP430X devices was presented by Gouvêa *et al.* in [41]. In their work the authors have used the product-scanning technique for the multi-precision multiplication. We thus also implemented this approach.

In our product-scanning implementation, where $h = f \times g \mod 2^{256} - 38$ is computed, we first compute the coefficients of the double-sized array, which results from multiplying f with g and then reduce this result modulo $2^{256} - 38$. We only have 7 general purpose registers available to store input operands during the multiplication operation. Hence, we cannot store all input operands in working registers, but we keep as many operands in them as possible. For the computation of a coefficient of the double-sized array, which results from multiplying f with g, one has to access the contents of f in incrementing and g in decrementing order, e. g. the coefficient h_2 is computed as $h_2 = f_0g_2 + f_1g_1 + f_2g_0$. As there is no indirect auto-decrement addressing mode available on the MSP430 microcontroller, we put the contents of g in reverse order on the stack at the beginning of the multiplication, allowing us to access g using indirect auto-increment addressing mode, which requires less clock cycles than indexed addressing mode, for the remaining part of the multiplication.

h_0	h_1	h_2	h_3	h_4	h_5	h_6	h_7	...
$f_0\ g_0$	$f_1\ g_0$	$f_2\ g_0$	$f_3\ g_0$	$f_4\ g_0$	$f_5\ g_0$	$f_6\ g_0$	$f_7\ g_0$	...
	$f_0\ g_1$	$f_1\ g_1$	$f_2\ g_1$	$f_3\ g_1$	$f_4\ g_1$	$2\ f_5\ g_1$	$f_6\ g_1$	...
		$f_0\ g_2$	$f_1\ g_2$	$f_2\ g_2$	$f_3\ g_2$	$2\ f_4\ g_2$	$2\ f_5\ g_2$	...
			$f_0\ g_3$	$f_1\ g_3$	$f_2\ g_3$	$2\ f_3\ g_3$	$2\ f_4\ g_3$	...
				$f_0\ g_4$	$f_1\ g_4$	$2\ f_2\ g_4$	$2\ f_3\ g_4$	...
					$f_0\ g_5$	$2\ f_1\ g_5$	$2\ f_2\ g_5$	...
						$f_0\ g_6$	$f_1\ g_6$	...
							$f_0\ g_7$	...

Figure 3.3. Visualization of computing the first coefficients of $h \leftarrow f \times g$ using the product-scanning technique. We first compute the double-sized array, which results from multiplying f with g and then reduce this double-sized array modulo $2^{256} - 38$.

3.4.5 Constant-Time Karatsuba Multiplication

This implementation is based on a very recent paper on the implementation of multi-precision multiplication on AVR microcontrollers [54]. Karatsuba presented a sub-quadratic multiplication method that reduces the number of required word multiplications for multi-precision multiplications [62]. [54] builds on this idea leading to a method named *subtractive Karatsuba*. This method avoids having to take extra carry bits into account by computing $|f_l - f_h|$ and $|g_l - g_h|$ instead of $f_l + f_h$ and $g_l + g_h$, which facilitates to obtain a constant-time implementation. In the following we report the method, as it was presented in [54], adapting it to the case of a 16-bit architecture. The steps for multiplying two n-byte numbers, where in our case $n = 32$, are described in detail and are summarized in Algorithm 4. Using a 16-bit architecture, we have to process arrays of $n/2 = 16$ elements. We split those arrays at $k = 16/2 = 8$.

Algorithm 4: Subtractive Karatsuba multiplication algorithm [54].

Input : f, g.
Output: $h = f \times g$.

$f_\ell + 2^{16k} f_h \leftarrow f$
$g_\ell + 2^{16k} g_h \leftarrow g$
$L \leftarrow f_\ell \cdot g_\ell$
$H \leftarrow f_h \cdot g_h$
$M \leftarrow |f_\ell - f_h| \cdot |g_\ell - g_h|$
if $M == (f_\ell - f_h) \cdot (g_\ell - g_h)$ **then**
  $t = 0$
end
else
  $t = 1$
end
$\hat{M} = (-1)^t M$;
return $(L + 2^{16k}(L + H - \hat{M}) + 2^{16n/2} H)$

We use the product-scanning technique for the computation of intermediate results of the Karatsuba implementation, i.e., the computations of L, H, and M. For the

computation of these we have seven working registers available to hold input operands. Hence, we can store almost the full input that is accessed in decrementing order in those working registers and access the eighth required operand of it using indirect addressing mode. $|f_\ell - f_h|$ is computed as follows: first we subtract with borrow all elements in f_h from those in f_ℓ and subtract with borrow from a register b_F that was cleared before. This results in $b_F = \texttt{0}$ for $f_\ell > f_h$ and $b_F = \texttt{0xffff}$ otherwise. We XOR b_F with $f_\ell - f_h$ obtaining the ones-complement of $f_\ell - f_h$. We then compute $t_F = b_F$ AND `1` add this to the ones-complement of $f_\ell - f_h$ and ripple the carry through, obtaining the two's complement of $f_\ell - f_h$, which is equal to $|f_\ell - f_h|$. $|g_\ell - g_h|$ is computed similarly. The value t required for the computation of $\hat{M}$ is obtained as $t = t_F \oplus t_G$. The same technique that was used to compute the absolute difference above is used for the computation of $\hat{M}$ from M leaving out the initial subtraction part. Again we computed the product of f and g resulting in a double-sized array and reduced the result mod $2^{256} - 38$.

3.4.6 Squaring

It turns out that on devices that have a 16-bit hardware multiplier, the constant-time Karatsuba multiplication performs best (cf. Section 3.5). We thus use constant-time Karatsuba in our implementation of squaring on MSP430X microcontrollers that have a 16-bit hardware multiplier. Intermediate results are computed using the product-scanning technique taking the aforementioned peculiarity of the MSP430's hardware multiplier unit, which is that contents of the hardware multiplier's `MAC` registers do not have to be loaded again in case the processed operands do not change, into account. This function executes in 2 427 cycles including function call and reduction overhead and in 1 937 cycles without.

On devices that have a 32-bit hardware multiplier the product-scanning technique performs better than constant-time Karatsuba, as it makes best use of the 32-bit

multiply-and-accumulate unit of the memory-mapped hardware multiplier. In order to compute $h \leftarrow f^2 \mod 2^{256} - 38$, we first compute a double-sized array obtained when squaring f and then reduce this result modulo $2^{256}-38$. During the computation of f^2 we have 8 general-purpose registers available to hold input operands. We store those operands of f that are accessed in decrementing order in working registers, and access the other operands using indirect auto-increment addressing mode. Including function call and reduction overhead our squaring implementation executes in 1 563 cycles on MSP430X microcontrollers that have a 32-bit hardware multiplier. Without this overhead it executes in 1 174 cycles.

3.4.7 Addition, Subtraction, Reduction and Multiplication with 121666

The x-coordinate-only doubling formula requires a multiplication with the constant 121666. The computation of $h = f \cdot 121666 \mod 2^{256} - 38$ greatly benefits from the aforementioned peculiarity of the MSP430 hardware multiplier. In case of having a 32-bit hardware multiplier we proceed as follows: The number 121666 can be written as $1 \cdot 2^{16} + 56130$. We store the value 1 in `MAC32H` and 56130 in `MAC32L` and then during each iteration load two consecutive coefficients of the input array f, *i.e.* f_i and f_{i+1} to `OP2L` and `OP2H` for the computation of two coefficients of the resulting array namely h_i and h_{i+1}. The array that results from computing $f \times 121666$ is only two elements longer than the input array, which we reduce as the next step. Using this method, the multiplication with 121666 executes in 352 cycles on MSP430s that have a 32-bit hardware multiplier, including function call and reduction overhead.

For the 16-bit hardware multiplier version, we follow a slightly different approach. As we cannot store the full number 121666 in the input register of the hardware multiplier, we proceed as follows: To compute $h = f \cdot 121666 \mod 2^{256} - 38$ we store the value 56130 in the hardware-multiplier register `MAC`. We then compute each h_i as $h_i = f_i \cdot 56130 + f_{i-1}$ for $i \in [1 \ldots 15]$ such that we add the $(i-1)$-th input

coefficient to the multiplier's result registers `RESLO` and `RESHI`. This step takes care of the multiplication with $1 \cdot 2^{16}$ for the $(i-1)$-th input coefficient. We further load the i-th input coefficient to the register `OP2`, thus executing the multiply-and-accumulate instruction to compute the i-th coefficient of the result. A special case is the computation of coefficient h_0, where $h_0 = f_0 \cdot 56130 + 38 \cdot f_{15}$. This method executes in 512 cycles including function call and reduction overhead.

The reduction of a double-sized array modulo $2^{256} - 38$ is implemented in a similar fashion. We store the value 38 in the `MAC`-register of the hardware multiplier. We then add the i-th coefficient of the double-sized input to the result registers of the hardware multiplier and load the $(i+16)$-th coefficient to the `OP2`-register. In the 32-bit version of this reduction implementation the only difference is that two consecutive coefficients can be processed in each iteration, i.e. the i-th and $(i+1)$-th coefficients are added to the result registers and and the $(i+16)$-th and $(i+17)$-th coefficient are loaded to the `OP2`-registers.

The modular addition $h = f + g \mod 2^{256} - 38$, which executes in 186 cycles on the MSP430X, first adds the two most significant words of f and g. It then extracts the carry and the most significant bit of this result and multiplies those with 19. This is added to the least significant word of f. All other coefficients of f and g are added with carry to each other. The carry resulting from the addition of the second most significant words of f and g is added to h_15, which was computed first. Note that this operation does not cause an overflow of h_15 as its most-significant bit was extracted before.

For the execution of $h = f - g$, g is subtracted with borrow from f. If the result of the subtraction of the most significant words produces a negative result, the carry flag is cleared, while, if it produces a positive result the carry flag is set. We add this carry flag to a register `tmp` that was set to `0xffff` before, resulting in the contents of `tmp` to be `0xffff` in case of a negative result and 0 in case of a positive result of the

subtraction. We AND `tmp` with 38, subtract this from the lowest resulting coefficient and ripple the borrow through. Again a possible resulting negative result of this procedure is reduced using the same technique, minus the rippling of the borrow. This modular subtraction executes in 199 cycles including function-call overhead.

3.5 Performance Analysis

We compiled our code using the C/C++ compiler [99], choosing the "high" profile and optimizing for speed. Note that all functions implementing arithmetic in $\mathbb{Z}_{2^{255}-19}$ were implemented in assembly, while higher level functions were implemented in C. As a first step we simulated cycle-count estimates of our software given by the IAR Embedded Workbench IDE. It turns out however that the IAR Embedded Workbench IDE does not consider wait cycles, which are arise from using the hardware multiplier. The IDE only gives cycle accurate estimates for functions executed by the main CPU of the MSP430, whereas the memory-mapped hardware multiplier counts as a peripheral of the MCU. To produce more accurate results, we measured the execution time of all multiplication implementations on the MSP-EXP430FR5969 LaunchPad Evaluation Kit [60] using the debugging functionality of the board and the IAR Embedded Workbench IDE [99].

Performance results of the various modular multiplication implementations are displayed in Table 3.1 and visualized in Figure 3.5. Note that the MSP-EXP430FR5969 LaunchPad Evaluation Kit hosts an MSP430FR5969, which has a 32-bit hardware multiplier. However, as mentioned in Section 3.2, a 16×16-bit multiplication executes in the same number of clock cycles on an MSP430 with 32-bit hardware multiplier as it would on an MSP430 hosting a 16-bit hardware multiplier.

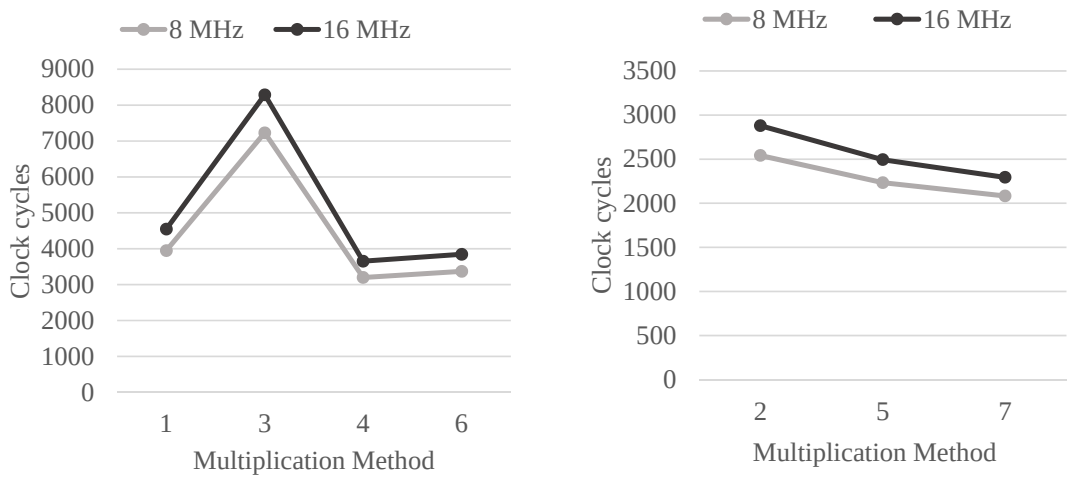
We realized during our measurements that the speed of the implementation is not doubled by increasing the operation frequency from 8 MHz to 16 MHz (Table 3.1). This is due to the limited access frequency of the FRAM. The MSP430FR5969 relies

Table 3.1. Measured cycle counts when executing the various multiplication implementations on the MSP-EXP430FR5969 Launchpad Evaluation Kit, operating the microcontroller at 8 MHz and 16 MHz respectively. The numbers marked with (a) include the function call and reduction overhead while those marked with (b) exclude it.

#	Method	8 MHz$^{(a)}$	8 MHz$^{(b)}$	16 MHz$^{(a)}$	16 MHz$^{(b)}$
1	16-bit Operand-caching	3 951	3 539	4 551	4 015
2	32-bit Operand-caching	2 541	2 119	2 879	2 399
3	16-bit Carry-save	7 230	7 202[1]	8 290	8 257[1]
4	16-bit Karatsuba	3 197	2 721	3 655	3 099
5	32-bit Karatsuba	2 233	1 843	2 494	2 049
6	16-bit Product-scanning	3 370	2 869	3 843	3 271
7	32-bit Product-scanning	2 082	1 678	2 293	1 836

[1] Here the reduction step is not excluded as it is interleaved with the multiplication computation.

on Ferroelectric Random Access Memory (FRAM) technology for non-volatile memory. The FRAM has a maximum access frequency of 8 MHz. Hence, wait cycles must be included when the MSP430FR5969 runs at frequencies higher than 8 MHz.



(a) MSP430X with 16-bit hardware multiplier (b) MSP430X with 32-bit hardware multiplier

Figure 3.4. Visualization of the execution times of the various multiplication implementations (Table 3.1) including function call and reduction overhead for (a) 16-bit hardware multipliers and (b) 32-bit hardware multipliers on the MSP-EXP430FR5969 Launchpad Evaluation Kit. The methods are numbered according to Table 3.1.

Table 3.2. Code space (in bytes) required for the variious modular multiplication implementations (including reduction) on MSP430Xs obtained by the IAR Embedded Workbench IDE.

#	**Method**	**Code Space**
1	16-bit Operand-caching	4 762 bytes
2	32-bit Operand-caching	2 878 bytes
3	16-bit Carry-save	8 448 bytes
4	16-bit Karatsuba	3 628 bytes
5	32-bit Karatsuba	2 322 bytes
6	16-bit Product-scanning	3 756 bytes
7	32-bit Product-scanning	2 088 bytes

We further present numbers for the required code space for the various multiplication implementations. Table 3.2 shows the required code space for each implementation obtained by the IAR Embedded Workbench IDE. It seems quite natural that the version making use of the 32-bit hardware multiplier is faster and requires less code space since fewer load (and store) operations to (and from) the dedicated registers of the hardware multiplier have to be executed.

Table 3.3 displays the required stack space for each multiplication implementation, which is the least for the operand-caching implementation. In terms of code space and execution time the 16-bit Karatsuba implementation performs best for devices that have a 16-bit hardware multiplier, whereas the Karatsuba implementation is outperformed by the 32-bit product-scanning implementation for devices that have a 32-bit hardware multiplier available.

We further present cycle counts for the execution of the Curve25519 variable-base-point single-scalar multiplication on the MSP430FR5969. We implemented scalar multiplication for the cases of having a 32-bit and a 16-bit hardware multiplier. For the 32-bit hardware multiplier case we used the product-scanning technique for the implementation of the modular multiplication and squaring, whereas for the 16-bit hardware multiplier case we used the subtractive Karatsuba technique for the imple-

Table 3.3. Stack space required for modular multiplication implementations (including reduction) on MSP430Xs obtained flushing RAM with a special character and observing, which RAM contents have been changed after the execution of each multiplication operation.

#	Method	Stack space
1	16-bit Operand-caching	182 bytes
2	32-bit Operand-caching	182 bytes
3	16-bit Carry-save	278 bytes
4	16-bit Karatsuba	210 bytes
5	32-bit Karatsuba	210 bytes
6	16-bit Product-scanning	214 bytes
7	32-bit Product-scanning	196 bytes

mentation of these modular arithmetic operations, as those are the fastest implementations for those cases according to Table 3.1.

On the MSP430FR5969 the scalar multiplication, which makes use of the 32-bit hardware multiplier, executes in 5 332 487 clock cycles and requires 7.1 kB of code space, whereas the 16-bit hardware multiplier version, executes in 7 909 028 clock cycles and requires 10.4 kB of code space. Our results for Curve25519 on the MSP430X microcontroller and a comparison with related previous work are summarized in Table 3.4. We only list results that target reasonably high security levels.

3.6 Power Consumption Analysis

What is described in this part was executed by Amir Moradi and is stated here to give a complete description of our results. We examined our code in terms of power consumption on the MSP-EXP430FR5969 Launchpad Evaluation Kit. For the power measurements we made use of a LeCroy WaveRunner HRO 66Zi digital sampling oscilloscope. As the MSP-EXP430FR5969 Launchpad Evaluation Kit has been developed to facilitate power measurements, we could easily place a 2.2 Ω shunt resistor at the Vdd path of the MSP430FR5969 microcontroller while no stabilizing capacitor was placed between the measurement point and the microcontroller. We

Table 3.4. Cycle counts, code sizes and stack usage of elliptic-curve scalar-multiplication software for MSP430X microcontrollers

	CPU	**Curve**	**Clk cycles @8 MHz**	**Clk cycles @16 MHz**	**Code** in bytes	**Stack** in bytes
With 16-bit hardware multiplier						
[106]	MSP430	NIST P-256	23 937 000	n/a	n/a	n/a
[105]	MSP430	NIST P-256	22 170 000	n/a	8 378	418
[41, 42]	MSP430X	NIST P-256	7 284 377[a]	n/a	n/a	n/a
This work	MSP430X	Curve25519	7 909 028	9 087 596	10 350	459
With 32-bit hardware multiplier						
[41, 42]	MSP430X	NIST P-256	5 321 776[a]	n/a	n/a	n/a
This work	MSP430X	Curve25519	5 332 487	5 968 096	7 116	457

[a] Note that the authors use the $4w$-NAF method for the scalar multiplication, which does not execute in constant time. In this chapter we focus on a constant-time implementation to thwart timing attacks. Further the authors obtained some of the cycle counts using the IAR Embedded Workbench simulator. It turns out that this simulator does not report correct timings, if the memory-mapped hardware multiplier of the MSP430 is used.

powered the Evaluation Kit by an external stable power supply and monitored the current passing through the shunt resistor by means of a LeCroy AP 033 differential probe at a sampling rate of 1 GS/s.

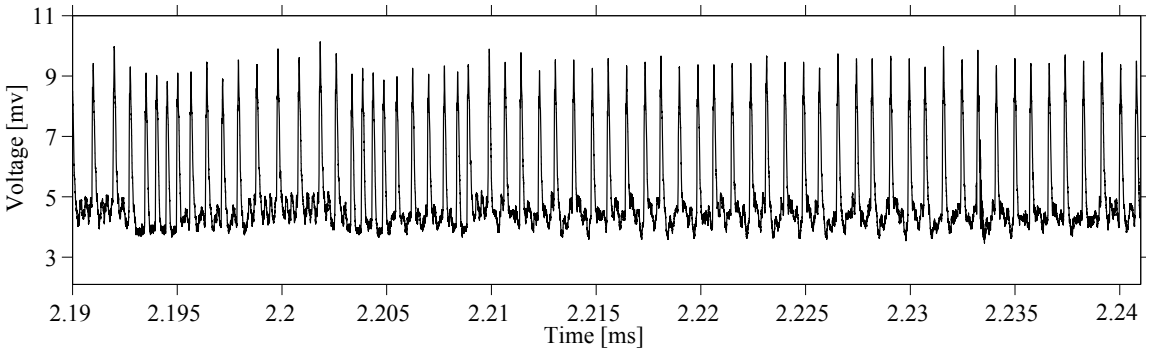


Figure 3.5. A sample power trace measured from MSP-EXP430FR5969 Launchpad Evaluation Kit when running 7 different multiplications

In Figure 3.6 we provide a zoomed view of a sample power trace to highlight several—non-periodic—high peaks which we have observed. We have observed the same peaks (but periodic) for a couple of NOP operations as well. The pattern of

these high peaks differs for different sequences of operations. It turns out that those peaks are due to FRAM accesses. FRAM reads are destructive, meaning that an FRAM read has to be followed by a write operation to keep data in memory. Due to the limited access frequency of the FRAM, TI has integrated a little cache allowing to run the CPU at higher frequencies and not limiting it to the access frequency of the FRAM. These peaks thus arise, if required data is not stored in cache (cache hit) and the FRAM has to be accessed.

We observed the voltage at each sample point using the differential probe and turned it into instantaneous power as $P = V^2/R$, where $R = 2.2\,\Omega$. The average of the instantaneous power values over the period of time gives us the power consumption of the device for that operation. We can turn this value to the amount of energy the device consumed by integrating the instantaneous power consumption values over the duration of an operation.

As stated above, using the 32-bit Karatsuba multiplication the debugging functionality of the IAR Embedded Workbench IDE reports 5 332 487 clock cycles for the execution of a scalar multiplication on Curve25519 on the board having a MSP430FR-5969. We verified this result measuring the length of the power trace. Based on our practical measurements one full execution of the algorithm takes around 672 ms with operation frequency of 8 MHz. This confirms the cycle count measured with IAR debugging functionality. To measure its power consumption we had to decrease the sampling rate to 200 MS/s due to the length of the trace (672 ms). Based on 100 measurements for random operands, in average the corresponding power consumption and energy consumption is 14.0 μW and 9.5 μJ respectively.

3.7 Discussion

This Chapter presents a full constant-time implementation of Curve25519 MSP430-X microcontrollers. In order to evaluate and improve the efficiency, we implemented

and analyzed different finite-field multiplication techniques and compared them in terms of speed and code size. Amongst all considered multiplication techniques, the subtractive Karatsuba implementation proposed in [54] performs the best for MSP430X devices that have a 16-bit hardware multiplier, while product-scanning works best for devices that have a 32-bit hardware multiplier. We analyzed our implementation with the MSP-EXP430FR5969 Launchpad Evaluation Kit. We further presented numbers for the average power and the energy consumption of the execution of Curve25519 on this platform. We showed that with an energy consumption of 9.5 μJ the execution of high-security ECC is feasible on devices operated with battery or harvested power.

CHAPTER 4

EFFICIENT E-CASH ON PASSIVE RFID TOKENS TARGETING PUBLIC TRANSPORTATION SYSTEMS

This chapter presents research, which has been published in [50, 86] and [87]. The results arose from a collaboration with Foteini Baldimtsi and Andy Rupp. Results that were not developed by the author of this dissertation will be clearly marked in the respective sections.

In this chapter we focus on privacy-preserving electronic payments for public transportation systems. A typical transportation payment system is sketched in Figure 4.1. A user can buy tickets at vending machines and use those to pay for trips at turnstiles at the entrance points of the transportation system. The payment process, *i.e.* giving out tickets and collecting payments, is controlled by the transportation authority.

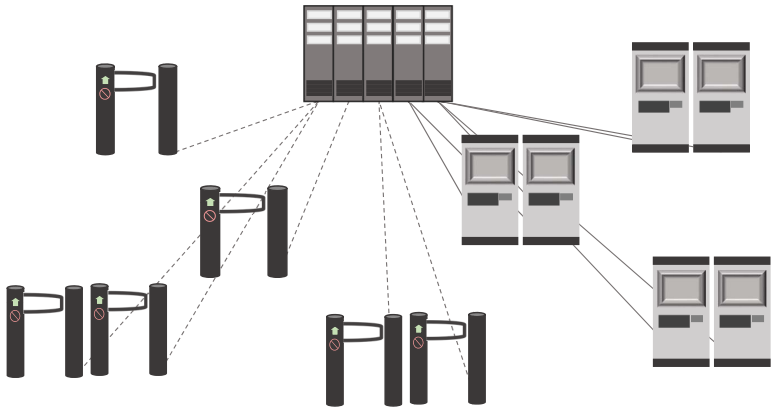


Figure 4.1. Overview of a public transportation payment system. A transportation authority owns and controls vending machines, where users buy tickets, and turnstiles, which a user can pass to access a transportation system, if he presents a valid ticket.

To avoid congestion in front of turnstiles, especially during rush hours, transactions have to be fast. A payment transaction should be executable within a few hundred milliseconds, allowing users to pay "on the go", *i.e.* when slowly passing turnstiles. Transactions at the vending machines are less time-critical, but should also not take longer than a few seconds, as this greatly reduces user convenience.

Vending machines and turnstiles, which are owned by the transportation authority, are equipped with readers responsible for conducting payment transactions with user devices. They can be equipped with powerful hardware, such as an ARM-processor or a PC-like platform. While we assume that vending machines have a constant connection to the back-end of the transportation authority, we do not expect that this holds for turnstiles. For example in case of buses this connection can be interrupted, which would impact the availability of payments if relying on it. We do however claim that an intermittent connection between turnstiles and the back-end system is available, enabling the transfer of payment data.

The use of smartphones as payment devices presents a promising solution. It is however desirable to at least additionally offer (extremely) inexpensive payment tokens. This greatly increases the number of users that can participate and provides ease-of-use. Inexpensive tokens can provide an experience similar to traditional paper tickets, while greatly adding convenience. Another central requirement for payment devices is that they can communicate with turnstiles in a contactless fashion, because contact-based communication conflicts with the need for short transaction times. Note that this is not required during interaction with a vending machine, as processes at the vending machine are less time critical. In this chapter we put our focus on fairly low-cost platforms, such as RFID transponders, contactless or hybrid smart cards, which are provided by the transportation authority.

We analyze the use of e-cash for payments in the public transport domain. E-cash offers great advantages for customers. An important one is that a user does not need

multiple payment devices to use transportation systems of different transportation authorities. Rather he can withdraw electronic coins at one transportation authority TA_1 and use them to pay for a trip in the transportation system of TA_2. Thus, TA_1 would act as the bank and TA_2 as a shop. TA_2 can later deposit the received coins to its account at TA_1. This is achieved, as no trust between shops and banks is assumed in the e-cash concept.

A particularly suitable electronic cash scheme for use in the domain of public transport, because of its exceptionally efficient spending protocol, was proposed by Brands' in 1993 [21]. However, the withdrawal protocol of Brands' scheme requires the execution of several computationally expensive public-key operations on the user side, which conflicts with the above mentioned resource constraints of user devices and the real-time requirements of payment executions. In this chapter we practically evaluate the usability of Brands' untraceable offline cash scheme for use in public transportation payment systems and demonstrate that through optimized implementation techniques we can combine the conflicting features of high computational requirements with inexpensive contactless RFID tokens and short transaction times.

We present a full implementation of Brands' e-cash scheme for a constrained RFID token, namely the *Moo* device [107]. The Moo is a computational RFID tag designed at the University of Massachusetts Amherst, which can communicate over a distance of up to several meters. It operates in the UHF band with a center frequency of 915 MHz and is passively powered, i.e., it does not rely on battery power, but harvests its energy from the RF field presented by the RFID reader that it communicates with. The Moo integrates the MSP430F2618 [56, 59], an ultra-low power MCU from Texas Instruments. The MSP430F2618 has a MSP430X CPU, 8 kB of RAM and 116 kB of flash memory. While having a larger RAM and non-volatile memory than the MSP430FR5969, which the investigations of Chapter 3 were based on, the MSP430F2618 has a 16-bit instead of a 32-bit memory-mapped

hardware multiplier. Even though the Moo is an experimental device, it is a close approximation of platforms that could be provided at low cost in future payment tokens. Based on market prices of some contactless smartcards, it can be assumed that if mass-produced, the price for a Moo will be in the range of a few dollars.

Our implementation, which builds on a 160-bit elliptic curve, shows that the scheme is fairly efficient even on this, not for our purposes optimized, hardware. Assuming a clock frequency of 4 MHz in contactless mode at turnstiles the computations required for spending can be executed in 13 ms. The computation required for the withdrawal process can be executed in 4.83 seconds on the Moo, if a core frequency of 16 MHz is used. This is a reasonable assumption since a payment device could be connected to a vending machine during the withdrawal process, thus allowing the provision of an external power source. By employing an ECC coprocessor as accelerator, the runtime of the time consuming public-key operations could further be improved.

Nevertheless, the results indicate that the number of coins that have to be withdrawn should be kept to a minimum, due to the time consuming withdrawal process. A possibility is to build a ticketing system based on Brands' scheme, where a coin is worth one ticket and thus a single coin has to be paid for a fair. This however conflicts with the desire of allowing variable and flexible prices, to e.g. account for the distance a user has traveled or to allow for reduced fares in case of delays. Setting the denomination of a coin to 1¢ certainly allows for flexible pricing but users would need plenty of them to pay for a trip. Setting the value to $2 reduces the number of required coins per trip but severely restricts the system of fares. A system would be imaginable, in which the transportation authority would pay electronic coins to a user as change. However, this cannot easily be achieved with Brands' scheme in a privacy-preserving way, which will be explained in more detail at a later point in this chapter.

Instead we follow an alternative approach, namely the privacy-preserving prepayments with refunds scheme (P4R) [86], specifically designed for transportation payment systems. In P4R which builds on Brands', an electronic coin is worth the most expensive trip in a system. At an entrance turnstile a user pays with a coin, allowing him to use the transportation system. Her actual fare cost is determined on the fly at an exit turnstile, when she leaves the transportation system. She then receives a refund based on her overpayment in a privacy-preserving way. We implemented this scheme on the same RFID token Moo and show its suitability as a promising solution for a privacy-preserving public transportation payment system, allowing dynamic and flexible prices.

4.1 Related work

The application domain of public transportation payment systems as an area of cryptographic research interest was proposed by Heydt-Benjamin *et al.* in [48]. It differs from other application areas in that payment collection costs need to be kept very low, while allowing for sufficiently secure payments. The authors propose the use of recent advances in anonymous credentials and e-cash systems, which can detect fraud while maintaining the anonymity of the honest user. In [89] Sadeghi *et al.* discuss an electronic public transport payment scheme based on RFID tokens. Relying on anonymizers their scheme protects the privacy of users against outsiders, but not against the transportation authority. Their argumentation is based on the idea that public-key cryptography cannot be executed efficiently on low-power devices such as RFID tokens. We argue that this assumption is not necessarily valid. We believe that using an efficient protocol, combined with advanced implementation methods will enable the use of electronic cash for public transportation payments. Blass *et al.* [17] proposed another offline "privacy-preserving" payment system for transit applications. This system solely relies on a 128-bit hash function and lots of precomputed

data on the back-end's side. However, again their system does not protect a user's privacy from the transportation authority.

An implementation of a payment scheme that is based on Brands' offline cash scheme is shown in [27], where a Sharp-Zaurus5600 PDA is used as the target platform. This is quite a powerful platform compared to what can be expected as payment devices for public transport ticketing. In contrast we use a passively powered device that approximates cheap payment tokens.

There have been several implementations of anonymous credentials on smartcards as for example [9] and [16]. There the authors tailor the protocol to the platform since no direct access to the integrated hardware is possible. In contrast we implement and evaluate a full-size e-cash scheme and avoid any changes to the protocol.

4.2 Review of Brands' Untraceable Offline Cash Scheme

This section briefly reviews basics of Brands' e-cash scheme. Note that in our scenario the vending machines represent the bank $\mathcal{B}$, *i.e.* they have access to keying material to generate blind signatures on coins, whereas turnstiles represent shops $\mathcal{S}$, which validate and accept payments.

Brands' e-cash scheme is set up by choosing a cyclic group G of prime order q, generators $g, g_1, g_2 \in G$, and a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. The bank's secret key is $\mathsf{sk}_\mathcal{B} \in_\mathcal{R} \mathbb{Z}_q^*$ and its corresponding public key is $\mathsf{pk}_\mathcal{B} = g^{\mathsf{pk}_\mathcal{B}}$.

To register for the payment service, a user $\mathcal{U}$ authenticates to the bank $\mathcal{B}$ by means of presenting some authentication document as for example a passport. She then chooses a secret key $\mathsf{sk}_\mathcal{U} \in_\mathcal{R} \mathbb{Z}_q^*$ and forms a public key $\mathsf{pk}_\mathcal{U} = g_1^{\mathsf{sk}_\mathcal{U}}$. If $g_1^{\mathsf{sk}_\mathcal{U}} g_2 \neq 1$ she sends her public key $\mathsf{pk}_\mathcal{U}$ to the bank, which stores it together with identifying information of the user. The bank further computes $z = (\mathsf{pk}_\mathcal{U} g_2)^{\mathsf{sk}_\mathcal{B}}$, which it sends to the user.

To initiate a withdrawal process, in which a user receives electronic coins from the bank, the user first authenticates to the bank through proving knowledge of her secret key. To generate an electronic coin, the withdrawal protocol shown in Table 4.1, in which a user receives a blind signature on some coin data, is executed between the user and the bank. This coin data is essentially a serial number $\mathtt{SN}$, which is randomly chosen by the user and encoded in a serial number tag A. The user further generates a blinding factor B, which she can later use to prove possession of the coin. The bank signs the serial number tag and the blinding factor in a way that the user gets a valid signature $\mathsf{Sign}(A, B)$, while the bank does not know neither A nor B nor the signature $\mathsf{Sign}(A, B)$. The user only accepts, if a valid coin has been generated during the protocol execution.

Table 4.1. Brands' withdrawal protocol [21]. In this protocol the user randomly chooses a serial number $\mathtt{SN}$ and encodes it in a serial number tag A. A and a blinding factor B, which is also formed by the user are blindly signed by the bank.

$\mathcal{U}$		$\mathcal{B}$
		$w \in_{\mathcal{R}} \mathbb{Z}_q$
$\mathtt{SN} \in_{\mathcal{R}} \mathbb{Z}_q^*,\ x_1, x_2, v \in_{\mathcal{R}} \mathbb{Z}_q, u \in_{\mathcal{R}} \mathbb{Z}_q^*$	$\xleftarrow{a,b}$	$a = g^w,\ b = (\mathsf{pk}_{\mathcal{U}} g_2)^w$
$A = (\mathsf{pk}_{\mathcal{U}} g_2)^{\mathtt{SN}},\ B = g_1^{x_1} g_2^{x_2}$		
$z' = z^{\mathtt{SN}}$		
$a' = a^u g^v,\ b' = b^{\mathtt{SN}u} A^v$		
$c' = \mathcal{H}(A, B, z', a', b')$		
$c = c'/u$	$\xrightarrow{c}$	
$g^r \stackrel{?}{=} \mathsf{pk}_{\mathcal{B}}^c a$	$\xleftarrow{r}$	$r = c\mathsf{sk}_{\mathcal{B}} + w$
$(\mathsf{pk}_{\mathcal{U}} g_2)^r \stackrel{?}{=} z^c b$		
$r' = ru + v$		

After execution of the withdrawal protocol the user knows a valid representation of a coin, which consists of the tuple $A, B, \mathsf{Sign}(A, B) = (z', a', b', r')$. She will store the coin in her electronic wallet together with the values $\mathtt{SN}, x_1$ and x_2, which she will later need to spend the coin. Anyone in possession of the bank's public key can verify the

signature, *i.e.* the validity of the coin, by computing the value $c' = \mathcal{H}(A, B, z', a', b')$ and checking Equation 4.1.

$$\text{Verify } \mathsf{Sign}(A, B): \; g^{r'} = \mathsf{pk}_{\mathcal{B}}{}^{c'} a' \text{ and } A^{r'} = z'^{c'} b'. \tag{4.1}$$

The user initiates a spending process by sending a coin $A, B, \mathsf{Sign}(A, B)$ to a shop. The spending process can be executed offline, meaning that no connection to the back-end system of the bank is required. Allowing payments to be verified offline is highly desirable for public transportation payment systems, due to the above mentioned fact that turnstiles do not have a constant connection to the transportation authority's back-end system. The shop validates $\mathsf{Sign}(A, B)$ and asks the user to prove ownership of the coin. In order to do so the user proves in zero-knowledge using the generated blinding factor B that she knows $\mathsf{sk}_{\mathcal{U}}$ and $\mathtt{SN}$ which are encoded in A. The shop can only verify the validity of the coin and that the user possesses it. Only the bank, which collects all spent coins in a database, can check whether it had been spent before.

A double-spending detection mechanism is used to detect, if a user used the same coin multiple times. During the spending process a user is forced to reveal partial data about herself by computing the values r_1 and r_2. This partial information by itself reveals nothing about her identity unless she spends the same coin again. In that case the bank can combine the two partial information sets and reveal the user's identity. The shop stores the received coin together with the values r_1, r_2 and ts. The spending protocol between the user and the shop is shown in Table 4.2. Here $\mathsf{ID}_{\mathcal{S}}$ denotes the shop ID and ts denotes a time stamp.

To deposit a coin to his bank account a shop sends the payment transcript $A, B, \mathsf{Sign}(A, B), r_1, r_2$ and ts to the bank. The bank computes $d = \mathcal{H}(A, B, \mathsf{ID}_{\mathcal{S}}, ts)$ and verifies the validity of the received coin. It then searches for this coin in its database of deposited coins. If it cannot find that this coin has already been de-

Table 4.2. Brands' spending protocol [21]. In this protocol the user sends a coin to a shop, which checks its validity as well as whether the user is in possession of it. It accepts the payment, if both checks hold.

$\mathcal{U}$		$\mathcal{S}$
	$\xrightarrow{A,B,\mathsf{Sign}(A,B)}$	$A \stackrel{?}{\neq} 1$
$r_1 = d(\mathsf{sk}_\mathcal{U}\ \mathtt{SN}) + x_1$	$\xleftarrow{d}$	$d = \mathcal{H}(A, B, \mathsf{ID}_\mathcal{S}, ts)$
$r_2 = d\ \mathtt{SN} + x_2$	$\xrightarrow{r_1,r_2}$	Verify $\mathsf{Sign}(A, B)$
		$g_1^{r_1} g_2^{r_2} \stackrel{?}{=} A^d B$

posited, it accepts the deposit, credits the shop's bank account and stores the payment transcript A, r_1, r_2, d in its database. If it finds A in its database this can have two reasons: (a) if $d = d'$ the shop is trying to double-deposit the coin and (b) if $d \neq d'$ a user double-spent the coin. In case (b) the bank can reveal the cheating user's identity from the two payment transcripts through

$$\mathsf{pk}_\mathcal{U} = g_1^{(r_1 - r_1')/(r_2 - r_2')}.$$

This scheme does not provide an easy way of realizing a privacy-preserving change system. A scenario is imaginable in which the user and the shop would switch roles and the shop could thus pay electronic coins to the user as change. If however the shop would act as a data broker and reveal to the bank its generated serial numbers of coins, the bank can identify which shop generated a particular electronic coin when it is deposited by a user. This entirely invalidates the privacy mechanisms of this e-cash scheme. Especially in the public transport domain, where the transportation authority owns vending machines and turnstiles, a transportation authority would know which turnstile had generated a particular serial number of a coin.

4.3 Implementation of Brands' E-cash on the Moo RFID Tag

Our implementation focuses on the withdrawal and the spending processes, as those are the time-critical parts that have to be executed frequently. While the execution of the withdrawal protocol is more efficient on the bank's side where only two exponentiations have to be executed in contrast to 12 exponentiations that have to be executed on the user's side, the spending protocol is extremely efficient on the user's side. Here only two modular additions and three modular multiplications have to executed in contrast to seven exponentiations that have to be executed on the bank's side.

We base the scheme on ECC since it is the most efficient established asymmetric cryptographic scheme, especially if compared to classical discrete logarithm (DL) based systems. This is due to the fact that, in contrast to DL and factoring-based public-key cryptosystems, the run-time of known attacks on ECC grows exponentially with the bit-length of the curve. Thus the same security level can be achieved with much shorter key lengths [84]. This can reduce the number of required computations and the amount of data that has to be communicated between protocol partners.

As we target a low-value payment system we conclude from [20] that basing the system on a 160-bit elliptic curve presents sufficient security, *i.e.* the efforts for breaking the system are high compared to the benefits that would be gained from it. To further increase security, system keys could be updated regularly. For our implementation we use the 160-bit prime curve *secp160r1* suggested by an industry consortium [82]. The underlying prime field $\mathbb{Z}_p$ of this curve is based on a generalized Mersenne prime $p = 2^{160} - 2^{31} - 1$ which allows for an efficient implementation of the curve arithmetic. The curve is given in short Weierstrass representation $E : y^2 = x^3 + ax + b$ where $a, b \in \mathbb{Z}_p$, such that $4a^3 + 27b^2 \neq 0$. The group G, generated by the base point P, is a cyclic group of prime order suitable for an implementation of Brands' offline cash scheme.

4.3.1 Implementation of the $\mathbb{Z}_p$ Framework

Similarly to what was presented in Chapter 3 we use an unsigned radix-2^{16} representation as this makes best use of the 16-bit CPU of the MSP430F2618. An element in $\mathbb{Z}_p$ is thus represented as an array of 10 words. We put our focus on the optimization of multiplication and squaring in $\mathbb{Z}_p$, as those will be used extensively in the implementation of the point multiplication in the ECC-framework.

Both functions implement two steps. In the first step the hybrid multiplication algorithm with $d = 2$ as proposed in [45] is used to compute a double-sized array, as the result of a multi-precision multiplication or multi-precision squaring of the input elements. This can be efficiently implemented, making use of the multiply-and-accumulate instruction of the hardware multiplier unit of the MSP430F2618. To achieve best performance we implemented these functions in assembly. The resulting double-sized array is reduced using the fast reduction of NIST-primes method described in [46].

To illustrate the reduction step let us assume the computation of $h = f^2 \mod p$, where f is an array of ten 16-bit words $f = f_9 2^{144} + f_8 2^{128} + \ldots + f_1 2^{16} + f_0$. Here the coefficient f_i denotes the integer that is stored in the i-th element of the array. The result $\hat{h}$ of squaring f is an array of 20 elements $\hat{h} = \hat{h}_{19} 2^{304} + \hat{h}_{18} 2^{288} + \ldots + \hat{h}_1 2^{16} + \hat{h}_0$. Using the fast reduction of NIST-primes [46], the reduction $h = \hat{h} \mod p$ is computed as

$$
\begin{aligned}
h = &[\hat{h}_9 + \hat{h}_{19} + (\hat{h}_{17}\texttt{>>1}) + (\hat{h}_{18}\texttt{<<15})]2^{144} + \ldots + [\hat{h}_4 + \hat{h}_{14} + (\hat{h}_{12}\texttt{>>1}) + (\hat{h}_{13}\texttt{<<15})]2^{64} \\
&+[\hat{h}_3 + \hat{h}_{13} + (\hat{h}_{11}\texttt{>>1}) + (\hat{h}_{12}\texttt{<<15}) + (\hat{h}_{19}\texttt{>>2})]2^{48} \\
&+[\hat{h}_2 + \hat{h}_{12} + (\hat{h}_{10}\texttt{>>1}) + (\hat{h}_{11}\texttt{<<15}) + (\hat{h}_{18}\texttt{>>2}) + ((\hat{h}_{19}\texttt{<<14})\ \texttt{\&}\ \texttt{0xc000})]2^{32}
\end{aligned}
$$

$$+[\hat{h}_1 + \hat{h}_{11} + (\hat{h}_{10}\texttt{<<15}) + (\hat{h}_{10}\texttt{<<5}) + (\hat{h}_{19}\texttt{>>1}) + ((\hat{h}_{18}\texttt{<<14}) \texttt{ \& 0x8000})]2^{16}$$
$$+\hat{h}_0 + \hat{h}_{10} + (\hat{h}_{19}\texttt{<<15}) + (\hat{h}_{18}\texttt{>>1}),$$

where $\texttt{j>>}x$ denotes a logical right shift of x by $\texttt{j}$ bits and $x\texttt{<<j}$ a logical left shift of x by $\texttt{j}$ bits, while $\texttt{\&}$ denotes a bit-wise $\texttt{AND}$ operation.

4.3.2 Implementation of the ECC Framework

Our implementation of the scalar multiplication on *secp160r1* relies on the Montgomery powering ladder [71]. Note that in contrast to Chapter 3, where we implemented the Curve25519 Diffie-Hellman (ECDH) key exchange protocol, we cannot rely on x-coordinate only scalar multiplication algorithms in the context of this implementation. Only the x-coordinate of the resulting point of a scalar multiplication is required to be used as shared secret of the ECDH key exchange as the y-coordinate provides only one extra bit of information. In contrast the y-coordinate of the result of a scalar multiplication is required for further protocol computations of Brands' scheme as multiple point multiplications and additions are executed.

Our scalar multiplication implementation is based on what was described in [84]. We rely on Jacobian coordinates for the point representation $P = (X, Y, Z)$. (X, Y, Z) is equivalent to the point $(x, y) = (X/Z^2, Y/Z^3)$ in affine coordinates. Using this representation of points, an inversion I can be exchanged for several modular multiplications M and squarings S. As can be seen in Table 4.3 the inversion is much more time consuming than multiplications and squarings, making the use of Jacobian coordinates advisable. Using Jacobian coordinates a point doubling can be compute at the cost of $4\mathsf{M}+6\mathsf{S}+8\mathsf{A}$, while point addition requires $12\mathsf{M}+4\mathsf{S}+7\mathsf{A}$ [84].

In [68] Meloni proposed an optimization for the point addition in Jacobian coordinates for the case, when two points share the same Z-coordinate, the so called co-Z addition requiring only $5\mathsf{M}+2\mathsf{S}+7\mathsf{A}$. It turns out that the Z-coordinate of an input

point can be updated to match the Z-coordinate of the resulting point at no extra cost [84]. Given two input points $P = (X_P, Y_P, Z)$ and $Q = (X_Q, Y_Q, Z)$, their sum $R = (X_R, Y_R, Z_R) = P + Q$ can be computed as

$$X_R = D - (B + C), \quad Y_R = (Y_Q - Y_P)(B - X_R) - E \quad \text{and} \quad Z_R = Z(X_Q - X_P), \tag{4.2}$$

where $A = (X_Q - X_P)^2$, $B = X_P A$, $C = X_Q A$, $D = (Y_Q - Y_P)^2$ and $E = Y_P(C - B)$. P can be updated as $P = (E, B, Z_R)$, such that it shares the same Z-coordinate with R [84], allowing repeated co-Z addition.

It was further shown in [39] that $\hat{R} = (X_{\hat{R}}, Y_{\hat{R}}, Z_R) = P - Q$, sharing the same Z-coordinate with R, can be computed at little extra cost as [84]

$$X_{\hat{R}} = F - (B + C) \quad \text{and} \quad Y_{\hat{R}} = (Y_P + Y_Q)(X_{\hat{R}} - B) - E, \tag{4.3}$$

where $F = (Y_P + Y_Q)^2$. The computation of $R = P + Q$ and $\hat{R} = P - Q$ together, where R and $\hat{R}$ share the same Z-coordinate requires 6M+3S+16A and is named conjugate co-Z addition.

The Montgomery powering ladder requires a point addition and subsequent point doubling, e.g. $R = P + Q$ and $\hat{R} = 2P$ in each iteration. This can be achieved with the above presented formulas (Equations 4.2 and 4.3) by first executing a conjugate co-Z point addition, resulting in $R_1 = P + Q$ and $R_2 = P - Q$, and a subsequent co-Z addition with update on those results, *i.e.* $R = R_1$ and $\hat{R} = R_1 + R_2 = 2P$, where the Z-coordinate of R_1 is updated to be the same as that of $\hat{R}$.

Note further from Equations 4.2 and 4.3 that the Z-coordinate of the input points is not required for the computations of $X_R, Y_R, X_{\hat{R}}$ and $Y_{\hat{R}}$. Repeated point additions can thus be executed without computing the Z-coordinate which further reduces the number of required operations. Yet, the Z-coordinate is needed at the end of the algorithm to transform the resulting point back to affine coordinates. The interested

reader is referred to [84] for detailed explanations on how the Z-coordinate of the resulting point can be recovered at the end of the Montgomery powering ladder. The idea is that after each iteration it holds that the difference between R and $\hat{R}$ is the input point P of which the affine representation is known. Due to $x = X/Z^2$ and $y = Y/Z^3$ it holds that $Z = xY/yX$.

All ideas combined can be used to achieve an efficient algorithm for point multiplication, shown as Algorithm 9 in [84], which we used for our implementation of the ECC framework on the MSP430F2618.

4.3.3 Implementation of the $\mathbb{Z}_q$ Framework

Brands' scheme requires operations in the finite field that is generated by the prime order q of G. We use an unsigned radix-2^{16} representation for elements in $\mathbb{Z}_q$ requiring 11 words for the representation of one element. The form of the prime, which is the order of the chosen elliptic curve, is not suitable to use the special NIST-reduction technique, which was used for reduction in the $\mathbb{Z}_p$ framework. Instead we implemented Barrett reduction as presented in [69] as an efficient method to reduce elements in $\mathbb{Z}_q$.

4.3.4 Implementation of a Hash Function

A hash function is required for the implementation of Brands' scheme that takes as input several points on the elliptic curve and returns as output an element in $\mathbb{Z}_q^*$. To ensure that the output of the hash function lies in $\mathbb{Z}_q^*$ we seek for a 160-bit output instead of 161 bits, which for the chosen curve is the bit-length of q. Further a coordinate on the elliptic curve is 160-bit in length. We hence seek for a hash function that takes 160-bit inputs and produces a 160-bit output, where we only process x-coordinates of points as inputs to the hash function.

We implemented the block cipher based hash function AES-hash [28]. Inputs are hashed block-wise using the previously hashed block as input and the to be hashed

Table 4.3. Timings of the $\mathbb{Z}_p$, $\mathbb{Z}_q$, and ECC framework, and the Hash functions $\mathcal{H}$

Function	**Cycle count**	**Execution time @16 MHz**	**Execution time @4 MHz**
Reduction in $\mathbb{Z}_p$	384	0.02 ms	0.10 ms
Multiplication in $\mathbb{Z}_p$	2,266	0.14 ms	0.57 ms
Squaring in $\mathbb{Z}_p$	1,678	0.11 ms	0.42 ms
Inversion in $\mathbb{Z}_p$	190,294	11.90 ms	47.60 ms
Reduction in $\mathbb{Z}_q$	11,648	0.73 ms	2.91 ms
Multiplication in $\mathbb{Z}_q$	17,101	1.07 ms	4.27 ms
Rijndael160	10,785	0.67 ms	2.69 ms
$\mathcal{H}$	54,958	3.43 ms	13.70 ms
Point addition	196,059	12.30 ms	49.00 ms
Scalar multiplication	6,312,785	395.00 ms	1,578.00 ms

block as key to the Rijndael block cipher [30]. The block ciphers output is XORed with the previously hashed block.

4.4 Performance of Brands E-Cash on the Moo RFID Tag

We developed our code and analyzed its performance using the IAR Embedded Workbench IDE for MSP430 v 5.40. We wrote higher-level code in C and sped up time critical functionality of $\mathbb{Z}_p$ using assembly language. In Table 4.3 timings for the implementation of the underlying primitives of the e-cash scheme are presented. As expected the special form of the prime, which the elliptic curve is based on, leads to a very efficient reduction and such to a fast execution of the multiplication and squaring in $\mathbb{Z}_p$.

Table 4.4 shows the execution times of the user side's computation of Brands' scheme. The results have been simulated for two frequencies, namely 4 MHz and 16 MHz. Our implementation is designed for the Moo, which operates at 4 MHz, due to the limited power that is available when relying on harvested power from the RF-field. However, the MSP430F2618 can be operated at a maximum frequency of 16 MHz. This could be achieved when providing the platform with an external power-

Table 4.4. Timing results of the user sides' computation of Brands' offline cash scheme on the MSP4302618

Cycle count Withdrawal	77,292,874
Cycle count Spending	52,050
Execution time Withdrawal @ 4 MHz	19.30 s
Execution time Spending @ 4 MHz	0.013 s
Execution time Withdrawal @ 16 MHz	4.83 s
Execution time Spending @ 16 MHz	0.003 s

Table 4.5. Code size of the user sides' implementation of Brands' scheme on the MSP430F2618

	CODE	**CONST**	**DATA**
$\mathbb{Z}_p$ framework	5,912	228	20
$\mathbb{Z}_q$ framework	2,308	158	22
ECC framework	3,088	120	20
Hash function	2,578	308	256
Protocol User	560	–	–

source. During the time-critical spending operation, which happens at turnstiles, we require a contactless operation of the Moo. The more computationally intensive part of Brands' scheme however is the withdrawal phase. During the withdrawal process the Moo could be connected to a vending machine thus allowing the higher operating frequency of 16 MHz.

The timings for the spending meet real-world requirements even for high-throughput transportation situations. An option would be to execute the withdrawal of coins at home, where the user connects the payment device to his personal computer and leaves it there for the charging period. In that case the withdrawal would not be very time-critical. Yet, there are several advantages of executing the withdrawal at a vending machine located near the entrance points of the public transportation system. Therefore the withdrawal (even of several coins) needs to be executable within a couple of seconds. However, during charging the payment device, it can be connected to the charging station allowing it to be supplied by an external power source

which supplies the passive tag with sufficient energy to power additional hardware accelerators.

The results indicate that Brands' scheme is a suitable solution for privacy-preserving payments in the public transport domain, even when relying on payment devices that are extremely constrained in computational power, if the number of coins that have to be spent for a fare is low. The scheme could serve as a ticketing system, where a user pays for a fare with a single coin. By using several blind signature key pairs at the bank, different denominations could be offered for different zones that a user can travel. However, as mentioned previously, it is highly desirable to allow for flexible and dynamic prices, *i.e.* allowing a fare price to be adapted to the distance that a user traveled. A system would be imaginable, where turnstiles would give out electronic coins to a user as change, which could again lower the number of coins that a user would have to spend, but this cannot easily be achieved with Brands' scheme. A solution to this we further present an implementation of the privacy-preserving pre-payments with refunds scheme (P4R).

4.5 Review of the Privacy-Preserving Pre-Payments with Refunds Scheme (P4R)

The privacy-preserving pre-payments with refunds scheme [86] has been developed and its security has been analyzed by Andy Rupp and Foteini Baldimtsi. We will briefly describe the scheme of which an implementation will be presented thereafter.

P4R is composed of a trip authorization token system (TAT), a refund calculation token system (RCT) and a refund token system (RT). A TAT, which a user can acquire at a vending machine, is a credential worth the most expensive fare and allows a user to make an arbitrary trip in the transportation system. When entering the transportation system the user shows an unused TAT where the showing does not reveal his identity. However, the identity of the user is encoded in the TAT

in a way that it can be revealed, if he is trying to use the same TAT to enter the transportation system multiple times. If the TAT is valid the user is granted access and he receives a stamped receipt RCT which contains a MAC on the spent TAT, the date and time, and the ID of the entrance turnstile $\mathsf{ID}_{\mathcal{S}}$. When leaving the system the user presents his RCT which is used to determine the turnstile the user passed to enter the transportation system and calculate his fare. To prevent that the user re-uses an RCT and thus can claim a higher refund in future trips, the RCT is bound to the ticket which a user has to present together with the RCT when claiming a refund at an exit turnstile. This requires the double showing of a TAT without revealing the user's identity. A user accumulates refunds in a refund token RT. He presents a blinded version of RT to the exit turnstile which adds the refund to it. When a user decides to redeem the collected refund she presents her RT to a vending machine which redeems the RT if it is not already marked as cashed in the database.

The TAT system is based on Brands' e-cash scheme (cf. Section 4.2) adapted to allowing the showing of an electronic coin twice (without revealing the identity of the user). To achieve this a user generates two blinding factors B_1 and B_2 during the withdrawal process and receives a signature $\mathsf{Sign}(A, B_1, B_2)$ on both. She can use those to proof possession of the TAT in zero-knowledge twice, once using blinding factor B_1 and once using B_2.

The refund token system builds on Boneh-Lynn-Schacham (BLS) signatures [19] to ensure that only a signer (an entity in possession of the private BLS signature key) can add refunds to a refund token. BLS signatures are based on bilinear pairings. Let G and G_T be cyclic groups of prime order q then a mapping $e : G \times G \rightarrow G_T$ is called bilinear pairing, if (i) it satisfies bilinearity: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for $g_1, g_2 \in G$ and $a, b, \in \mathbb{Z}_q$, (ii) it satisfies non-degeneracy: if g is a generator in G then $e(g, g) \neq 1$ and (iii) it is efficiently computable: an efficient algorithm exists to compute $e(g_1, g_2)$. In the BLS signature scheme a signer randomly picks a secret key $\mathsf{sk}_{\mathcal{S}} \in_{\mathcal{R}} \mathbb{Z}_q$ and computes

a public key $\mathsf{pk}_{\mathcal{S}} = g^{\mathsf{sk}_{\mathcal{S}}}$. A signature $\sigma(\mathsf{msg})$ is generated as $\sigma(\mathsf{msg}) = \mathsf{msg}^{\mathsf{sk}_{\mathcal{S}}}$ and can be verified by verifying whether

$$e(g, \sigma(\mathsf{msg})) \stackrel{?}{=} e(\mathsf{pk}_{\mathcal{S}}, \mathsf{msg}).$$

In P4R a user receives a refund token $\mathsf{RT} = \mathtt{SN}_{\mathsf{RT}} \in_{\mathcal{R}} G$ from the transportation authority when withdrawing a number of coins. The transportation authority can add a refund value w to this token as $\mathsf{RT}' = \mathsf{RT}^{\mathsf{sk}_{\mathcal{S}}^{w}}$. Multiple executions of this equation accumulate refund values w_i in RT's exponent

$$\mathsf{RT}' = \mathsf{RT}^{\mathsf{sk}_{\mathcal{S}}^{\sum w_i}}.$$

Assuming that users do not check collected refunds, which is reasonable in the public transportation domain where refunds usually have low values, we do not need to rely on a bilinear pairing group for the implementation of the refund token system, as the only entity checking refunds is the transportation authority, who is in possession of the secret BLS-signature key. We implemented the scheme for this scenario and rely on the same ECC framework that was used for the implementation of plain Brands' scheme.

The set-up and user registration of P4R are essentially the same as those of Brands' scheme (cf. Section 4.2). During set up the transportation authority additionally has to choose a message authentication code MAC and has to generate a MAC-key $\mathsf{sk}_{\mathsf{MAC}}$ and a BLS signature key sk_{BLS}, which it provides to its turnstiles.

To initiate the BuyTAT process, in which a user receives TATs and a fresh RT from a vending machine, the user first proves knowledge of his secret key. For each TAT he then executes the BuyTAT protocol (first part of Table 4.6), in which he receives a blind signature on a generated serial number tag A and two blinding factors B_1 and B_2. After execution of the BuyTAT protocol the user knows a valid representation of a

TAT consisting of the tuple $A, B_1, B_2, \mathsf{Sign}(A, B_1, B_2) = (z', a', b', r')$. She stores TAT together with the values $\mathtt{SN}, x_1, x_2, y_1, y_2$ which she will use to spend the TAT later on. She further receives a fresh refund token RT (second part of Table 4.6) which she stores together with R, which is used to accumulate randomization values, and the value of the refund token v_{RF}, which is initialized to zero.

Table 4.6. P4R's BuyTAT and GetRT protocol. In the BuyTAT protocol the user randomly chooses a serial number $\mathtt{SN}$ and encodes it in a serial number tag A. He further generates two blinding factors B_1 and B_2 and receives a blind signature from the bank on A, B_1 and B_2. In the GetRT protocol the user receives a fresh RT.

$\mathcal{U}$		$\mathcal{B}$
		$w \in_{\mathcal{R}} \mathbb{Z}_q$
$\mathtt{SN} \in_{\mathcal{R}} \mathbb{Z}_q^*,\ x_1, x_2, y_1, y_2, v \in_{\mathcal{R}} \mathbb{Z}_q, u \in_{\mathcal{R}} \mathbb{Z}_q^*$	$\xleftarrow{a,b}$	$a = g^w,\ b = (\mathsf{pk}_{\mathcal{U}} g_2)^w$
$A = (\mathsf{pk}_{\mathcal{U}} g_2)^{\mathtt{SN}}$		
$B_1 = g_1^{x_1} g_2^{x_2},\ B_2 = g_1^{y_1} g_2^{y_2}$		
$z' = z^{\mathtt{SN}}$		
$a' = a^u g^v,\ b' = b^{\mathtt{SN}u} A^v$		
$c' = \mathcal{H}(A, B_1, B_2, z', a', b')$		
$c = c'/u$	$\xrightarrow{c}$	
$g^r \stackrel{?}{=} \mathsf{pk}_{\mathcal{B}}^c a$	$\xleftarrow{r}$	$r = c\mathsf{sk}_{\mathcal{B}} + w$
$(\mathsf{pk}_{\mathcal{U}} g_2)^r \stackrel{?}{=} z^c b$		
$r' = ru + v$		
$\mathsf{RT} = \mathtt{SN}_{\mathsf{RT}}, R = 1, v_{\mathsf{RF}} = 0$	$\xleftarrow{\mathtt{SN}_{\mathsf{RT}}}$	$\mathtt{SN}_{\mathsf{RT}} \in_{\mathcal{R}} G$, add $\mathtt{SN}_{\mathsf{RT}}$ to database

When the user enters the transportation system he executes the $\mathsf{ShowTAT}$ and GetRCT protocols together with the turnstile (Table 4.7). The user first shows an unused TAT and proves possession of it. If it is valid, *i.e.* Equation 4.1 holds, access is granted and the user receives an RCT.

When leaving the transportation system, the user shows her RCT, and proves possession of its contained TAT, this time using blinding factor B_2. If it is valid the refund amount w is calculated by the exit turnstile and added to to the randomized refund token also presented by the user.

Table 4.7. P4R's ShowTAT and GetRCT protocol. In this protocol $\mathcal{U}$ presents a TAT and $\mathcal{S}$ checks its validity as well as whether $\mathcal{U}$ possesses it. The payment is accepted, if both checks hold and $\mathcal{U}$ receives an RCT.

$\mathcal{U}$		$\mathcal{S}$
	$\xrightarrow{A,B_1,B_2,\mathsf{Sign}(A,B_1,B_2)}$	$A \stackrel{?}{\neq} 1$
$r_1 = d(\mathsf{sk}_\mathcal{U}\ \mathtt{SN}) + x_1$	$\xleftarrow{d}$	$d = \mathcal{H}(A, B_1, B_2, \mathsf{ID}_\mathcal{S}, ts)$
$r_2 = d\ \mathtt{SN} + x_2$	$\xrightarrow{r_1,r_2}$	Verify $\mathsf{Sign}(A, B_1, B_2)$
		$g_1^{r_1} g_2^{r_2} \stackrel{?}{=} A^d B_1$
	$\xleftarrow{\mathsf{RCT}}$	$\mathsf{RCT} = (\mathsf{TAT}, ts, \mathsf{ID}_\mathcal{S},$
		$\mathsf{MAC}_{\mathsf{sk}_\mathsf{MAC}}(\mathsf{TAT}, ts, \mathsf{ID}_\mathcal{S}))$

Table 4.8. P4R's ShowRCT and GetRefund protocol. In the ShowRCT protocol the user presents his RCT to the exit turnstile, which checks its validity and computes the refund w the user should receive based on it. In the GetRefund protocol the user presents a blinded version of her refund token and the turnstile adds w to it.

$\mathcal{U}$		$\mathcal{S}$
	$\xrightarrow{A,B_1,B_2,\mathsf{Sign}(A,B_1,B_2)}$	$A \stackrel{?}{\neq} 1$
$r'_1 = d'(\mathsf{sk}_\mathcal{U}\ \mathtt{SN}) + y_1$	$\xleftarrow{d'}$	$d' = \mathcal{H}(A, B_1, B_2, \mathsf{ID}_\mathcal{S}, ts)$
$r'_2 = d'\ \mathtt{SN} + y_2$	$\xrightarrow{r'_1,r'_2}$	Verify $\mathsf{Sign}(A, B_1, B_2)$
		$g_1^{r'_1} g_2^{r'_2} \stackrel{?}{=} A^{d'} B_2$
$r \xleftarrow{\$} \mathbb{Z}_p^*$	$\xleftarrow{w}$	determine refund $w \in \mathbb{Z}_{p-1}$
$\mathsf{RT}' = \mathsf{RT}^r$	$\xrightarrow{\mathsf{RT}'}$	
$v_\mathsf{RF} = v_\mathsf{RF} + w$, $R = Rr \bmod p$	$\xleftarrow{\mathsf{RT}''}$	$\mathsf{RT}'' = \mathsf{RT}'^{\mathsf{sk}^w_{BLS}}$
$\mathsf{RT} = \mathsf{RT}''$		

To redeem a refund a user randomizes her refund token and presents it to the vending machine. If it belongs to this user, correctly encodes v_RF and is not marked as cashed in the database, the transportation authority returns the money to the user and marks RT as cashed in its database.

Table 4.9. P4R RedeemRT protocol

$\mathcal{B}$		$\mathcal{U}$
		$\mathsf{rnd} \in_{\mathcal{R}} \mathbb{Z}_q$
Check validity of $\mathtt{SN}_{\mathsf{RT}}$ and whether it had been given to this user	$\xleftarrow{\mathsf{RT}', \mathtt{SN}_{\mathsf{RT}}, v_{\mathsf{RF}}, R, ts,}$	$\mathsf{RT}' = \mathsf{RT}^{\mathsf{rnd}},\ R = R \times \mathsf{rnd}$
$v \overset{?}{<} q - 1$		
$\mathsf{RT}' \overset{?}{=} \mathtt{SN}_{\mathsf{RT}}^{R\mathsf{sk}_{BLS}^{v_{\mathsf{RF}}}}$	$\xrightarrow{\text{money}}$	

4.6 Performance of P4R on Moo RFID tag

We evaluate the scheme's performance, by presenting implementation results of the scheme for the Moo RFID tag [107] and estimating its required storage space. We choose the same ECC framework for the implementation that we used for our implementation of Brands (Section 4.3). For our storage estimation we assume affine coordinate representation of points. We list how much storage space is required for the different data bundles (Table 4.11).

Implementation results for the user side's computation on the Moo are presented in Table 4.10 for 4 MHz and 16 MHz. We note that ShowTAT & GetRCT and ShowRCT & GetRefund are considered the time-sensitive operations as they happen during the actual transportation, e.g., at a turnstile during rush hour. The results show that the protocols for entering the system (ShowTAT & GetRCT) can be executed efficiently even on the chosen unoptimized prototyping device. The execution time for receiving a refund (ShowRCT & GetRefund) is more time critical, but could easily meet real-world requirements, when making use of dedicated hardware or allowing for clock rates higher than 4 MHz. The most time consuming, but also less time critical step is buying TATs (BuyTAT). Yet, buying TATs is done at vending machines, where the device could be physically connected to the machine and hence additional power would be available to power hardware accelerators, that can speed up the execution of an elliptic curve point multiplication by an order of magnitude or more, resulting in

Table 4.10. Timing results for the user side's computation of The P4R protocols: BuyTAT, GetRT, ShowTAT & GetRCT, ShowRCT & GetRefund and RedeemRT

	Cycle Count	Execution time @ 4 MHz in s	Execution time @ 16 MHz in s
BuyTAT	83,567,483	20.89	5.22
GetRT	242	≈ 0	≈ 0
ShowTAT & GetRCT	35,657	0.009	0.002
ShowRCT & GetRefund	5,786,013	1.45	0.36
RedeemRT	5,519,689	1.38	0.34

a total execution time of a second or less. Of course, this would increase the cost of a payment device. However, in the case of millions of payment devices being rolled out, which can be assumed for metropolitan areas, the cost of such dedicated hardware would be reasonably low.

We assume the storage of points in affine coordinate representation, where each coordinate is an element in the 160-bit field $\mathbb{Z}_p$. We thus require 40 bytes of storage for an element in G. The order of the group generated by the base point of *secp160r1* spans a 161-bit prime field $\mathbb{Z}_q$. We thus require 21 bytes for storing an element of $\mathbb{Z}_q$. Based on these parameter choices, our storage space estimations for P4R are summarized in Table 4.11. The total storage requirements on a user device to make 20 trips is at most 1×0.06 kB $+ 20 \times 0.37$ kB $+ 1 \times 0.12$ kB $+ 1 \times 0.04$ kB $= 7.62$ kB.

A user's secret key comprises of two elements, an element in G and one in $\mathbb{Z}_q$, which requires 61 bytes of storage. This data is generated once, when the user opens an account and will not be deleted for the lifetime of the user device. For each TAT 6 elements in G and $\mathbb{Z}_q$ have to be stored respectively. This sums up to 0.37 kB of data that has to be stored for each TAT and can be deleted after using it. To collect refunds the user receives a refund token and maintains some other associated data, consisting of two elements in G and two in $\mathbb{Z}_q$, which requires 0.12 kB of storage. This data can be deleted when redeeming the refund. Please note that refunds are accumulated using the 161-bit variable v_{RF}, which leads to a more than sufficiently

Table 4.11. Storage space estimation for payment data in P4R

	Elements	Storage	Comments
$pk^{\mathcal{U}}$, $sk^{\mathcal{U}}$	$\mathsf{ID}_{\mathcal{U}}$, $g_1^{\mathsf{ID}_{\mathcal{U}}}$	0.06 KB	Stored once
TAT	$A_i, B_i, C_i, \mathsf{Sign}(A_i, B_i, C_i)$ $s_i, x_i, y_i, x_i', y_i'$	0.37 KB	Can be deleted after use
RT	$\mathsf{RT}, \mathtt{SN}_{\mathsf{RT}}, R, v$	0.12 KB	One for a bundle of TATs
RCT	$\mathsf{MAC}_K(\mathsf{TAT}_i, \mathrm{ts}, \mathsf{ID}_{\mathcal{R}}), \mathsf{ID}_{\mathcal{R}}, \mathrm{ts}$	0.04 KB	Stored only during a trip

large upper bound for the refund amount. During a trip, an RCT token has to be stored. The TAT belonging to this RCT is already contained in the memory of the user device and does not require any extra storage. We estimate ts and $\mathsf{ID}_{\mathcal{S}}$ to require 10 bytes of memory each, while storing $\mathsf{MAC}_K(\mathsf{TAT}, ts, \mathsf{ID}_{\mathcal{S}})$ requires 21 bytes, which adds up to additional 41 bytes of data.

We further estimate the size of the four databases required for P4R. The user database stores identifying information for each user along with his public key and is a comparably small database. The TAT database and the RCT database keep track of the used TATs and RCTs of a user, in order to detect double-spending. Here we store the values A, z_1, z_2 per TAT and A, z_1', z_2' per RCT, which results in 82 bytes per TAT and RCT token, respectively. Additionally, the transportation authority uses an RT database to store all RTs. Per RT token a serial number is stored along with a boolean value, indicating whether this RT has been redeemed, resulting in 41 bytes per RT.

Let us now consider an average ridership of 1.28 million passengers per day, which was the case in the MBTA system in Boston in February 2013 [67]. Then the TAT and the RCT databases grow by 105 MB per day or 38.3 GB per year, respectively. Assuming that on average a user buys bundles of 10 TATs and uses a single RT token per bundle to collect the corresponding refunds, 47 million RTs are issued per year. This results in a growth rate of 1.9 GB of storage per year. Thus, the overall storage requirements for the databases involved in our system are pretty modest. Also, note

Table 4.12. Comparison of Brands' e-cash for coins with a denomination of 10 cents and P4R (for MSP430 operating at 4 MHz when user enters/leaves and at 16 MHz when he charges his payment device)

	Charging device	Enter System	Exit System
Arbitrary trip in P4R	5.22 s	0.009 s	1.45 s
$1 trip with Brands' e-cash	42.57 s	0.081 s	-
$2.30 trip with Brands' e-cash	98 s	0.19 s	-
$4.90 trip with Brands' e-cash	209 s	0.40 s	-

that the size of these databases can be limited by changing system parameters on a regular basis or restricting the lifetime of TAT and RT tokens by attaching expiration dates.

4.7 Performance Comparison of P4R with Brands' E-cash

In this section, we briefly compare the performance of P4R and the transportation payment system directly built from Brands' e-cash, when allowing prices of fares to be flexible. We consider Brands' e-cash with only a single and small denomination value to allow for flexible pricing and avoid the problem of overpayments and privacy-preserving change. Of course we could allow Brands' coins to have different denominations however it would still not be a single coin per trip as in P4R. A further benefit of e-cash with refunds is that, as a trip is charged at the end of a trip, a truly flexible pricing scheme can be applied, i.e. lower prices on overcrowded buses. With Brands' e-cash the cost for a trip would have to be determined on entrance of the system.

Table 4.12 shows the performance of both schemes for different fares assuming Brands e-coins with a denomination of 10 cents. Clearly, the runtime of withdrawal and spending in Brands' scheme grows linearly with the number of required coins and thus with the fare of a trip, whereas in P4R we have a constant runtime independent of the fare. Last, assuming an average fare of $2.50, the database for Brands' scheme

(required for double-spending detection) would grow by 957.8 GB each year compared to 78.5 GB in total per year for all three databases required for the P4R scheme.

4.8 Discussion

We presented an implementation of Brands' offline cash scheme for a computational RFID tag. Spending is considered the time-sensitive operation as this happens during the actual transportation, e.g., at a turnstile during rush hour. The achieved spending timings meet real-world application requirements even for high throughput transportation situations. The withdrawal of coins could be done at home, where the user connects the payment token to his personal computer, and leaves it there for the charging period. In that case the withdrawal would not be very time-critical. Yet, there are several advantages of the withdrawal taking place at charging stations located near the entrance points of the public transportation system. In that case, the withdrawal (even of several coins) needs to be executable in a couple of seconds. Yet, during withdrawal the payment device could be connected to the charging station, which supplies the passive tag with sufficient energy to power additional hardware accelerators, which could greatly speed up execution timings.

Nevertheless, the results of the implementation of Brands' scheme advise that the number of coins that a user has to withdraw to make a fare has to be kept to a minimum, which conflicts with the desire to allow for flexible and dynamic prices. This can be solved relying on the privacy-preserving pre-payments with refunds scheme, which was designed to target the specific requirements of the public transportation setting. We presented an implementation of this scheme and compared our results with the implementation of Brands' scheme. Our results demonstrate that lightweight e-cash schemes do present an attractive solution for privacy-preserving payments in the public transportation domain, even when relying on extremely constrained hardware.

CHAPTER 5

IMPLEMENTATION OF E-CASH WITH ATTRIBUTES FOR PUBLIC TRANSPORTATION SYSTEMS

In this chapter we present work, which has been published in [52] and was extended in [51]. The results arose from collaborative work with Foteini Baldimtsi, Felix Riek and Christian T. Zenger. Results that were not developed by the author of this dissertation will be clearly marked in the respective sections.

Another attractive feature of some e-cash schemes is that they allow the encoding of user attributes into electronic coins. An attribute could for example be the user's age, zip code, or his eligibility to receive a discount. The user can selectively disclose an attribute when spending an electronic coin, meaning he can decide while spending an electronic coin, whether he would like to reveal a particular attribute or not. This mechanism is highly attractive for payment systems in the transportation domain as it (i) can be used to realize discount systems, *i.e.* a user receives a discounted fare, if he is in possession of an attribute proving his participation in the discount program and (ii) privacy-preserving data mining, *i.e.* the collection of meaningful data about customer behavior in the system. For a transportation authority it is highly desirable to collect user data to maintain and improve a transportation infrastructure based on users' needs. The encoding and selective disclosure of user attributes allows collecting user data in a privacy-preserving way as it allows a coarse-grained collection of user data, without revealing the identity of the user. In particular, it allows a user not to participate in the data collection at any time, if he does not feel comfortable sharing information in a particular situation.

In this chapter we extend our considerations of the previous chapter using e-cash in the public transport domain, presenting an implementation of two e-cash schemes, which allow the encoding and selective disclosure of user attributes, on potential payment devices that suit the transportation setting. We consider Brands' untraceable offline cash scheme [21] and Anonymous Credentials Light (ACL) [7] used as e-cash.

To satisfy the tight timing requirements imposed by the transportation setting, we choose a payment device that can communicate with an access point in a contactless fashion. A standard for contactless communication integrated in modern smartphones is Near Field Communication (NFC) [1]. It allows a smartphone to communicate with other NFC-enabled devices within a range of a few centimeters. While the throughput is moderate, the benefit of this type of communication is its simple and hence fast establishment, as electronic devices can be connected with a simple touch. There are predictions that in the long run NFC devices will replace the multitude of smartcards that many users carry around currently. For transportation authorities the advantage of relying on users' NFC-enabled smartphones is that no additional (electronic) tokens will have to be handed out. Instead only a software-app has to be provided that the user can download to his phone. This contributes to decreasing the revenue collection cost, and further allows the payment system to be updated easily. If a change to the system is made, the transportation authority only needs to provide a software update, rather than a hardware roll-out.

Our implementation is based on elliptic curve cryptography (ECC). As smartphone we use the BlackBerry Bold 9900 featuring a Qualcomm Snapdragon MSM8655 processor running at 1.2 GHz. It is equipped with the operating system BlackBerry OS 7 and is programmed using Java SDK API 7.1.0 provided by Research in Motion (RIM). We show how the schemes can be implemented efficiently using the functionality that is provided by the BlackBerry Java API since version 3.6.0. We have

developed a subtle technique that enables us to use the `ECDHKeyAgreement` class, which is present in this and other Java APIs, for computing the scalar multiplication on an elliptic curve. While developed and shown for this device, the use of this technique is not limited to the BlackBerry Bold 9900. It can be applied to devices that support efficient implementations of ECC, but only allow access to most commonly used results, as ECDH key agreement or ECDSA (as for example some Java smart cards). While the ECDH key agreement essentially executes a scalar multiplication, the APIs often only give access to the x-coordinate of the resulting point, since only this is needed as shared secret. Our technique shows, how to efficiently recover the y-coordinate for those cases. We show that both schemes execute efficiently on the BlackBerry Bold 9900, while a limiting factor is the NFC-communication bandwidth.

For employment it is however highly desirable to execute payment applications on a secure device, such as the SIM card or the embedded secure element of a smartphone, instead of a smartphone's main processor. Those devices are usually equipped with a relatively small microprocessor, similar to hardware that can be found in smartcards. Access to SIM cards or embedded secure elements is highly restricted, such that it is difficult for researchers to implement protocols on them. As an additional hurdle in practice, is it surprisingly difficult for researchers to have free access to smartcards, *i.e.* to program smartcards using native code. Usually, if at all, only high-level access is provided [101]. This has made it quite difficult to analyze the performance of non-standard cryptographic protocols on smartcards. While certain cryptographic primitives (such as AES or 3DES encryption), which are required for many security applications, are supported by many smartcard APIs, advanced cryptographic protocols, especially those requiring the execution of computationally complex arithmetic, cannot easily be implemented on these platforms in an acceptable execution time.

The ST23ZL48 is a microcontroller from STMicroelectronics for secure smartcard applications [97]. This chip is integrated in Ubivelox MULTOS smartcards, namely

UBM21-Z48 cards[1], whose API supports ECC functionality. We present an implementation of the same e-cash schemes allowing the encoding of attributes on this off-the-shelf smartcard. Our implementation demonstrates that Brands' untraceable offline cash scheme [21] and the Anonymous Credentials Light scheme (ACL) [7] supporting the encoding of two attributes into coins can be implemented efficiently on low-cost smartcards in practice meeting the stringent timing requirements of modern payment systems. Our implementation on a UbiVelox MULTOS smartcard achieves an execution time of the time-critical spending phase in 800 ms for both schemes.

5.1 Related Work

Multiple works present an implementation of anonymous credentials on smartcards. Bichsel *et al.* presented an implementation of the Camenish-Lysyanskaya anonymous credential system on a Java Card [16]. However, their best result requires more than 7 seconds for the issuance of a credential. Sterckx *et al.* presented an implementation of the Direct Anonymous Attestation scheme on Java Cards [95], where their execution of the signing protocol requires 4.2 s. Vullers *et al.* presented an implementation of an Idemix based credential scheme on a MULTOS smartcard, which they called IRMA card [104]. When encoding two attributes, their implementation, which is based on a 1024-bit modulus, executes in 1.1 s if not revealing an attribute and 0.9 s, when revealing both attributes. Finally, Mostowski and Vullers presented an implementation of the U-Prove system on MULTOS smartcards in [72]. Their issuance of a U-Prove token requires 3.6 s, when allowing the encoding of two attributes into the token, while showing the credential without revealing the encoded attributes requires 550 ms.

[1] `http://www.multos.com/products/approved_platforms/`

Only few implementations of full e-cash schemes on mobile platforms exist. While a variation of Brands' untraceable offline cash scheme has been implemented on a PDA by Clemente-Cuervo *et al.* [27], no implementation of ACL has been presented in the literature.

5.2 Review of E-cash with Attributes

For completeness, we review the basics of the implemented protocols, namely Brands' untraceable offline cash scheme [21] and Anonymous Credentials Light (ACL) [7] allowing the encoding of attributes, in this section. The description of those protocols has been developed by Foteini Baldimtsi and was published in [52].

A transportation payment system based on e-cash is described as an interaction between users $\mathcal{U}$ and a transportation authority. We envision the same scenario as has been described in Chapter 4, depicted in Figure 4.1. We assume that a transportation authority owns vending machines and turnstiles, where vending machines have a constant connection to the back-end servers of the transportation authority, and turnstiles have an intermittent connection. The vending machines together with the back-end system of the transportation authority take the role of the bank $\mathcal{B}$ of an e-cash scheme. Here a user can recharge his electronic wallet. Turnstiles, which are positioned at the entrance points of a transportation system take the roles of shops $\mathcal{S}$, granting a user access, if he presents a valid ticket (electronic coin).

5.2.1 Brands' E-cash Allowing the Encoding and Selective Disclosure of Attributes

In Brands' e-cash scheme [21] when allowing the encoding of attributes the bank $\mathcal{B}$ chooses a cyclic group G of prime order q, and $n + 2$ generators of this group $g_0, g, g_1, g_2, \ldots, g_n$, where n is the number of attributes supported by the system.

It further chooses a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q^*$, and generates a key pair by choosing a secret key $\mathsf{sk}_\mathcal{B} \in_\mathcal{R} \mathbb{Z}_q^*$ and computing its public key as $\mathsf{pk}_\mathcal{B} = g^{\mathsf{sk}_\mathcal{B}}$.

5.2.1.1 User Registration

To use the payment system, a user $\mathcal{U}$ has to register at the bank. To do so, he presents some form of identification, e.g. a passport, and generates a commitment C for his attributes $L_2, \ldots, L_n \in \mathbb{Z}_q$ and his secret key $\mathsf{sk}_\mathcal{U} = L_1 \in_\mathcal{R} \mathbb{Z}_q^*$. His public key is $\mathsf{pk}_\mathcal{U} = g_1^{\mathsf{sk}_\mathcal{U}}$. The user would either have to reveal his attributes to the bank when committing to them or obtain them from some other trusted authority and then prove knowledge of them to the bank. Then using a standard Schnorr *AND proof* of knowledge [92] of several discrete logarithms, the user proves that he knows an opening of the commitment C and that the same value L_1 has been used to generate C and $\mathsf{pk}_\mathcal{U}$. The bank checks the validity of the proof, stores the user's information and computes $z = (\mathsf{C}h)^{\mathsf{sk}_\mathcal{B}}$ which it sends to the user. The user registration process is summarized in Table 5.1.

Table 5.1. Brands' with attributes user registration protocol

$\mathcal{B}$		$\mathcal{U}$
		$\mathsf{sk}_\mathcal{U} = L_1 \in_\mathcal{R} \mathbb{Z}_q^*$
		$\mathsf{C} = g_1^{L_1} \ldots g_n^{L_n},\ \mathsf{pk}_\mathcal{U} = g_1^{\mathsf{sk}_\mathcal{U}}$
		If $\mathsf{C} \neq 1$:
		$w_1, \ldots, w_n \in_\mathcal{R} \mathbb{Z}_q$
		$\mathsf{C}' = g_1^{w_1} \ldots g_n^{w_n},\ pk'_\mathcal{U} = g_1^{w_1}$
		$k = \mathcal{H}(\mathsf{C}/\mathsf{pk}_\mathcal{U}, \mathsf{C}'/pk'_\mathcal{U}, \mathsf{pk}_\mathcal{U}, pk'_\mathcal{U}, ts)$
		$s_1 = w_1 + kL_1$
	$\xleftarrow{\mathsf{C}/\mathsf{pk}_\mathcal{U}, \mathsf{C}'/pk'_\mathcal{U}, \mathsf{pk}_\mathcal{U}}$	$\vdots$
$\mathsf{pk}_\mathcal{U}{}^k pk'_\mathcal{U} \stackrel{?}{=} g_1^{s_1}$	$\xleftarrow{\mathsf{pk}_\mathcal{U}, pk'_\mathcal{U}, ts, s_1, \ldots, s_n}$	$s_n = w_n + kL_n$
$(\mathsf{C}/\mathsf{pk}_\mathcal{U})^k \mathsf{C}'/pk'_\mathcal{U} \stackrel{?}{=} \prod_{i=2}^n g_i^{s_i}$		
Store identifying information of $\mathcal{U}$ together with C		
$z = (\mathsf{C}g_0)^{\mathsf{sk}_\mathcal{B}}$	$\xrightarrow{z}$	Store z

5.2.1.2 Withdrawal

To withdraw coins the user first has to authenticate himself to the bank and claim how many coins he wishes to withdraw. This is necessary as otherwise a user could later deny his wish to withdraw electronic coins and further an attacker could initiate the withdrawal process, harming the user by debiting money from his bank account, if the user set up some direct debit service. Note that the attacker could not spend the withdrawn electronic coins, as knowledge of the user's secret key is required for spending.

The user authenticates himself and claims, how many coins he wishes to withdraw, by computing a Schnorr signature [92] $\sigma(\mathsf{msg})$ on a message msg of the form: $\mathsf{msg} =$"N, ts", where N is the number of coins he would like to withdraw and ts is a time stamp. The bank will authorize the withdrawal process, if this signature verifies under the user's public key. The identification phase is presented in Table 5.2.

Table 5.2. User identification phase

$\mathcal{B}$		$\mathcal{U}$
		$\mathsf{msg} =$"N, ts"
		$r \in_{\mathcal{R}} \mathbb{Z}_q$
		$k = \mathcal{H}(\mathsf{msg}, g_1^r),\ y = r + \mathsf{sk}_{\mathcal{U}} \times k$
Check $\sigma_{\mathsf{pk}_{\mathcal{U}}}$:	$\xleftarrow{\mathsf{msg}, \sigma_{\mathsf{pk}_{\mathcal{U}}}(\mathsf{msg})}$	$\sigma_{\mathsf{pk}_{\mathcal{U}}}(\mathsf{msg}) = (y, k)$
$\mathcal{H}(\mathsf{msg}, \frac{g_1^y}{\mathsf{pk}_{\mathcal{U}}{}^k}) \stackrel{?}{=} k$		

For the withdrawal of each coin, the withdrawal protocol, summarized in Table 5.3 is executed. In this protocol a user generates a serial number $\mathtt{SN}$, encodes it in a serial number tag A, generates a blinding factor B, and receives a blind signature $\mathsf{Sign}(A, B)$ on A and B from the bank. The blinding factor B encodes random values that a user can use later on to prove possession of the coin, and possession of the respective attributes.

Table 5.3. Brands' with attributes withdrawal protocol

$\mathcal{B}$		$\mathcal{U}$
$w \in_{\mathcal{R}} \mathbb{Z}_q$		
$a = g^w,\ b = (\mathsf{C}g_0)^w$	$\xrightarrow{a,b}$	$\mathtt{SN} \in_{\mathcal{R}} \mathbb{Z}_q^*$
		$A = (\mathsf{C}g_0)^{\mathtt{SN}},\ z' = z^{\mathtt{SN}}$
		$x_0, x_1, \ldots, x_n, u, v \in_{\mathcal{R}} \mathbb{Z}_q$
		$B = A^{x_0} g_1^{x_1} \ldots g_n^{x_n}$
		$a' = a^u g^v,\ \ b' = b^{\mathtt{SN}u} A^v$
		$c' = \mathcal{H}(A, B, z', a', b')$
	$\xleftarrow{c}$	$c = c'/u \mod q$
$r = c\mathsf{sk}_{\mathcal{B}} + w$	$\xrightarrow{r}$	$g^r \stackrel{?}{=} pk_{\mathcal{B}}^c a,\ (\mathsf{C}h)^r \stackrel{?}{=} z^c b$
		$r' = ru + v$

For each coin, the user stores the values $A, B, \mathsf{Sign}(A,B) \hat{=} A, B, z', a', b', r'$ together with $\mathtt{SN}$ and $x_0, x_1, \ldots, x_n$. The signature can be verified through Equation 5.1.

$$\text{Verify } \mathsf{Sign}(A,B)\text{: } g^{r'} = \mathsf{pk}_{\mathcal{B}}{}^{c'} a' \quad \text{and} \quad A^{r'} = z'^{c'} b' \tag{5.1}$$

5.2.1.3 Spending

To spend a coin $A, B, \text{sign}(A,B)$ the user sends it to a shop $\mathcal{S}$ with ID $\mathsf{ID}_{\mathcal{S}}$. He then proves a representation of A, *i.e.* that he is in possession of the coin, using his blinding factor B. If the user wants to reveal an attribute L_j, he does not compute r_j but sends this attribute in the clear together with the value x_j generated during withdrawal. Note that revealing a larger number of attributes decreases the number of equations r_i that have to be computed on the user device, which however have to be computed by the shop, but increases the amount of communicated data. After the transaction and if the signature verifies, *i.e.* the coin is valid, the shop saves the payment transcript consisting of $A, B, \text{sign}(A,B), R, r_1, \ldots, r_n$ and the time stamp ts. This transaction is summarized in Table 5.4.

Table 5.4. Brands' with attributes spending protocol when revealing attribute L_j

$\mathcal{U}$		$\mathcal{S}$
	$\xrightarrow{A,B,\mathsf{Sign}(A,B)}$	$A \stackrel{?}{\neq} 1$
$r_1 = -dL_1 + x_1$	$\xleftarrow{d}$	$d = \mathcal{H}_0(A, B, \mathsf{ID}_\mathcal{S}, ts)$
$\vdots$		
$r_{j-1} = -dL_{j-1} + x_{j-1}$		
$r_{j+1} = -dL_{j+1} + x_{j+1}$		
$\vdots$		
$r_n = -dL_n + x_n$		
$R = d/\mathtt{SN} + x_0$	$\xrightarrow{R,r_1,\ldots r_{j-1},r_{j+1},\ldots,r_n,L_j,x_j}$	$r_j = -dL_j + x_j$
		$g_1^{r_1} \ldots g_j^{r_j} \ldots g_n^{r_n} g_0^{-d} \stackrel{?}{=} A^{-R}B$
		Verify $\mathsf{Sign}(A, B)$

5.2.1.4 Deposit

In order to deposit a coin, the shop submits the payment transcript $A, B, \text{sign}(A, B)$, $R, r_1, \ldots, r_n$ and the time stamp ts to the bank. The bank verifies the validity of the coin, *i.e.* whether it has a valid signature and whether a user had proven possession of it. It then queries its database, where all deposited coins are recorded, to check whether this coin had been deposited before. This double spending check does not need to happen during the deposit phase. The bank could run it at specific time intervals for all the coins in the database. If the deposited coin had not been recorded in the database before, the bank will store $A, d, R, r_1, \ldots, r_n, ts, \mathsf{ID}_\mathcal{S}$ in her database.

However, if the coin had been recorded, it means that it had been deposited before. The bank then checks whether $ts \stackrel{?}{=} ts'$ and $\mathsf{ID}_\mathcal{S} \stackrel{?}{=} \mathsf{ID}'_\mathcal{S}$. If so, the shop is trying to deposit the same coin twice. If not, the user spent the coin twice. Then the identity of the user can be revealed by computing

$$\mathsf{C} = g_1^{(r_1 - r'_1)/(R-R')} \ldots g_n^{(r_n - r'_n)/(R-R')},$$

which was stored together with some identifying information of the user during the registration process.

5.2.2 ACL E-cash Allowing the Encoding and Selective Disclosure of Attributes

Anonymous Credentials Light [7] is a recent, very efficient "linkable" anonymous credential system, which was constructed on top of Abe's blind signature scheme [2]. In a linkable credential system multiple showings of a credential are linkable, thus a user can show a credential only once, if he does not want transactions to be linkable. This however, is suitable to be used as e-cash where credentials, *i.e.* electronic coins, are used only once.

During the set up phase of ACL e-cash the bank chooses a group G of order q, a generator g and a hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q$. It also chooses $z, h, h_0, h_1, h_2, \ldots h_n \in_{\mathcal{R}} G$, where n is the maximum number of attributes that will be encoded into coins. The secret key of the bank is $\mathsf{sk}_{\mathcal{B}} \in_{\mathcal{R}} \mathbb{Z}_q^*$, whereas its public key is $\mathsf{pk}_{\mathcal{B}} = g^{\mathsf{sk}_{\mathcal{B}}}$ and z is the *tag public key*.

5.2.2.1 User Registration

When a user $\mathcal{U}$ with attributes $L_2, \ldots, L_n$, secret key $\mathsf{sk}_{\mathcal{U}} = L_1 \in_{\mathcal{R}} \mathbb{Z}_q^*$ and public key $\mathsf{pk}_{\mathcal{U}} = h_1^{L_1}$ wants to open an account at the bank $\mathcal{B}$ he presents a valid identification document and commits to his attributes and public key, as shown in Table 5.5. He then proves knowledge of his secret key to the bank and that he formed the commitment correctly. For each user the bank stores $\mathsf{pk}_{\mathcal{U}}, \mathsf{C}/\mathsf{pk}_{\mathcal{U}}$ and a copy of his identification document. Additionally to his secret key, attributes, public key and commitment the user has to store the randomness R that corresponds to his commitment C.

5.2.2.2 Withdrawal

To withdraw k coins from his account the user authenticates to the bank as it was presented in Table 5.2. Then the user runs the ACL blind signature protocol (Table 5.6) k times, once for each coin.

Table 5.5. ACL with attributes user registration protocol

$\mathcal{B}$		$\mathcal{U}$
		$R \in_{\mathcal{R}} \mathbb{Z}_q,\ \mathsf{sk}_{\mathcal{U}} = L_1 \in_{\mathcal{R}} \mathbb{Z}_q$ $\mathsf{C} = h_0^R \prod_{i=1}^{n} h_i^{L_i}$ $w_0, \ldots, w_n \in_{\mathcal{R}} \mathbb{Z}_q$ $\mathsf{C}' = h_0^{w_0} \ldots h_n^{w_n},\ pk'_{\mathcal{U}} = h_1^{w_1}$ $k = \mathcal{H}(\mathsf{C}/\mathsf{pk}_{\mathcal{U}}, \mathsf{C}'/pk'_{\mathcal{U}}, \mathsf{pk}_{\mathcal{U}}, pk'_{\mathcal{U}}, ts)$ $s_0 = w_0 + kR$ $s_1 = w_1 + kL_1$
	$\xleftarrow{\mathsf{C}/\mathsf{pk}_{\mathcal{U}}, \mathsf{C}'/pk'_{\mathcal{U}}, \mathsf{pk}_{\mathcal{U}}}$	$\vdots$
$\mathsf{pk}_{\mathcal{U}}{}^k pk'_{\mathcal{U}} \stackrel{?}{=} h_1^{s^1}$ $(\mathsf{C}/\mathsf{pk}_{\mathcal{U}})^k \mathsf{C}'/pk'_{\mathcal{U}} \stackrel{?}{=} h_0^{s_0} \prod_{i=2}^{n} g_i^{s_i}$ Store identifying information of $\mathcal{U}$ together with C	$\xleftarrow{\mathsf{pk}_{\mathcal{U}}, pk'_{\mathcal{U}}, ts, s_1, \ldots, s_n}$	$s_n = w_n + kL_n$

Through execution of the withdrawal protocol the user obtains a $coin = (\zeta, \zeta_1, \rho, \omega, \rho'_1, \rho'_2, \omega')$, which he stores together with rnd, τ and γ. For each withdrawn coin, the bank stores z_1 together with the user's public key. This is later used in order to identify a user in case he double-spends the coin.

5.2.2.3 Spending

The ACL e-cash spending protocol is presented in the upper part of Table 5.7. A user spends a coin to a shop which validates its signature. If a user wants to reveal an attribute L_j during the spending process, he will execute the revealing attribute L_j protocol, which is presented in the lower part of Table 5.7. In this protocol a user proves that the attributes encoded in his commitment are the same as are encoded in the withdrawn coin, *i.e.* in value ζ_1. The shop saves the payment transcript consisting of $coin, \varepsilon_p, \mu_p, desc$ and ts.

Table 5.6. ACL with attributes withdrawal protocol

$\mathcal{B}$		$\mathcal{U}$
$\mathsf{rnd} \in_{\mathcal{R}} \mathbb{Z}_q$		
$z_1 = \mathsf{C}g^{\mathsf{rnd}}$, $z_2 = z/z_1$		
$u, c', r'_1, r'_2 \in_{\mathcal{R}} \mathbb{Z}_q$		
$a = g^u$, $a'_1 = g^{r'_1} z_1^{c'}$		
$a'_2 = h^{r'_2} z_2^{c'}$	$\xrightarrow{\mathsf{rnd}, a, a'_1, a'_2}$	$z_1 = \mathsf{C}g^{\mathsf{rnd}}$
		$\gamma \in_{\mathcal{R}} \mathbb{Z}_q^*$
		$\zeta = z^\gamma$, $\zeta_1 = z_1^\gamma$, $\zeta_2 = \zeta/\zeta_1$
		$\tau \in_{\mathcal{R}} \mathbb{Z}_q$, $\eta = z^\tau$
		Check whether $a, a'_1, a'_2 \in G$
		$t_1, t_2, t_3, t_4, t_5 \in_{\mathcal{R}} \mathbb{Z}_q$
		$\alpha = a g^{t_1} y^{t_2}$, $\alpha'_1 = a_1'^{\gamma} g^{t_3} \zeta_1^{t_4}$
		$\alpha'_2 = a_2'^{\gamma} h^{t_5} \zeta_2^{t_4}$
		$\varepsilon = \mathcal{H}(\zeta, \zeta_1, \alpha, \alpha'_1, \alpha'_2, \eta)$
$c = e - c'$	$\xleftarrow{e}$	$e = \varepsilon - t_2 - t_4$
$r = u - cx$	$\xrightarrow{c, r, c', r'_1, r'_2}$	$\rho = r + t_1$
		$\omega = c + t_2$
		$\rho'_1 = \gamma r'_1 + t_3$
		$\rho'_2 = \gamma r'_2 + t_5$
		$\omega' = c' + t_4$

5.2.2.4 Deposit

To deposit *coin* the shop sends the coin as well as $\varepsilon_p, \mu_p, desc$ and ts to the bank. The bank verifies, whether the coin is valid and whether *desc* correctly encodes $\mathsf{ID}_{\mathcal{S}}$ and ts.

In order to check, whether a coin has been spent before, which can be done at a later point in time, the bank checks, whether *coin* already exists in its database of deposited coins. If so, it will check, whether $desc \stackrel{?}{=} desc'$. If they are equal it means that shop deposited the same coin twice. If they are different the bank can compute

$$\gamma = (\mu'_p - \mu_p)/(\epsilon_p - \epsilon'_p) \quad \text{and} \quad z_1 = \zeta_1^{1/\gamma}.$$

It can then identify the cheating user, by checking whom z_1 has been given to during the withdrawal protocol.

Table 5.7. ACL with attributes spending protocol revealing the attribute L_j

$\mathcal{U}$		$\mathcal{S}$
$\epsilon_p = \mathcal{H}(z^\tau, coin, desc)$	$\xleftarrow{desc}$	$desc = \mathcal{H}(\mathsf{ID}_\mathcal{S}, ts)$
$\mu_p = \tau - \epsilon_p\gamma$	$\xrightarrow{\varepsilon_p, \mu_p, coin}$	$\zeta \stackrel{?}{\neq} 1$
		$\epsilon_p \stackrel{?}{=} \mathcal{H}(z^{\mu_p}\zeta^{\epsilon_p}, coin, desc)$
		$\omega + \omega'$
		$\stackrel{?}{=} \mathcal{H}(\zeta, \zeta_1, g^\rho y^\omega, g^{\rho'_1}\zeta_1^{\omega'}, h^{\rho'_2}\zeta_2^{\omega'}, z^{\mu_p}\zeta^{\epsilon_p})$
$\mathsf{rnd}' \in_\mathcal{R} \mathbb{Z}_q$		
$\mathsf{C}' = h_j^{L_j\gamma} g^{\mathsf{rnd}'\gamma}$		
$r, r', r_0, \ldots, r_n \in_\mathcal{R} \mathbb{Z}_q$		
$\widetilde{\zeta_1} = h_0^{r_0} \ldots h_n^{r_n} g^r$		
$\widetilde{\mathsf{C}'} = h_j^{r_j} g^{r'}$		
$c = \mathcal{H}(\zeta_1, \widetilde{\zeta_1}, \mathsf{C}', \widetilde{\mathsf{C}'}, ts')$		
$s_0 = r_0 + cR\gamma$		
$s_1 = r_1 + cL_1\gamma$		
$\vdots$		
$s_n = r_n + cL_n\gamma$		
$s = r + c\ \mathsf{rnd}\ \gamma$	$\xrightarrow{L_j, \zeta_1, \widetilde{\zeta_1}, \mathsf{C}', \widetilde{\mathsf{C}'}}$	
$s' = r' + c\ \mathsf{rnd}'\ \gamma$	$\xrightarrow{s_0, \ldots, s_n, s, s', ts'}$	$c = \mathcal{H}(\zeta_1, \widetilde{\zeta_1}, \mathsf{C}', \widetilde{\mathsf{C}'}, ts')$
		$\widetilde{\zeta_1}\zeta_1^c \stackrel{?}{=} h_0^{s_0} \ldots h_n^{s_n} g^s$
		$\widetilde{\mathsf{C}'}\mathsf{C}'^c \stackrel{?}{=} h_j^{s_j} g^{s'}$

5.3 Implementation of E-cash with Attributes on a BlackBerry Bold 9900

This section arose from collaborative work with Christian T. Zenger during his master thesis, which was supervised by the author of this dissertation.

We will now describe important aspects of our NFC-smartphone implementation of the described e-cash with attributes schemes. In our measurement setup the terminal, which represents the vending machines, *i.e.* the bank, as well as the turnstiles, *i.e.* shops, is composed of a personal computer and an OMNIKEY smart card reader from HID Global that is connected to the computer via USB. The user's payment device is represented by a BlackBerry Bold 9900, featuring NFC-capabilities. The

BlackBerry Bold 9900 is programmed using the BlackBerry Java SDK API 7.1.0[2] provided by Research in Motion.

5.3.1 Near Field Communication (NFC) Framework

All aspects of NFC are specified in ISO/IEC standards. We use the card-emulation mode provided by the BlackBerry API. In this mode the BlackBerry Bold emulates a standard-conform smartcard. Building the payment system on standard appliances, makes it conform to already installed payment infrastructure and hence facilitates deployment. The underlying standard for communication between the smartphone and the reader in card-emulation mode is ISO/IEC 14443-A. This standard describes the communication signal interface of contactless smart cards, operating at 13.56 MHz with a bandwidth of 106 kbit/s. Both the Java SDK API 7.1.0 of the BlackBerry device, and the JRE 6 System Library of the terminal support this standard.

Data is exchanged between the terminal and the (emulated) smartcard using so-called Application Protocol Data Units (APDUs). The reader initializes the communication by sending a command APDU to the smartcard. The smartcard executes this command and replies with a response APDU. The communication procedure between a smartcard and a smartcard reader is specified in standard ISO/IEC 7816-4. Note that the size of an APDU is limited to 256 bytes.

Several attacks on NFC enabled mobile phones have been demonstrated, yet most of them target the use of passive tags [103, 73] while we make use of the card emulation mode. The security of the card emulation mode greatly depends on its implementation, in our case the Blackberry Java API. In [47] the authors identified five threats that are introduced by the NFC communication link. Eavesdropping can be considered little problematic in our setting, as intercepted data is of no use to an attacker. This is because (1) no user information is sent, apart from user attributes, which are

[2]http://www.blackberry.com/developers/docs/7.1.0api/

assumed not to reveal private information about users and (2) if an attacker would know the representation of a user's coin, he could still not use it to pay with it for a trip, as the private key of the user is required during the spending phase. An attacker could harm a user, by corrupting or modifying sent data, and hence not letting him execute the payment. Preventing denial-of-service attacks is very hard and would occur with other contactless communication links also.

A relay attack is imaginable, in which an attacker acts as a man-in-the-middle, by having two NFC-enabled devices that constitute a communication link, bringing one of them in close proximity to a turnstile and the other in close proximity to a user device and then paying with the coins of the user for a trip, if he manages to confirm the payment of coins on the user device. But, in the transportation domain this setting is very hard to realize in practice, as the attacker would have to bring one device in close proximity to the payment machine and another one in close proximity to the user device, while additionally having to pass the turnstile. A user can prevent this attack by deactivating the NFC functionality, which is a further benefit of relying on an NFC smartphone rather than on contactless smart cards.

5.3.2 Cryptographic Framework

We base the schemes on elliptic curve cryptography, and deduce from [20] that a 160-bit elliptic curve presents sufficient security for a micro-payment system. Similar to the previous chapter we chose the curve *secp160r1* from [82]. On the terminal side we use the Bouncy Castle Crypto Library version 1.5[3]. This library provides a general elliptic curve framework supporting the use of many different curves. A dedicated implementation of the elliptic curve functionality for the terminal's hardware could lead to a better performance of the execution of the payment schemes and is realistic in the transportation setting. However, our investigations focus on the execution of

[3]http://www.bouncycastle.org/

the protocols on the user device and the communication of the protocols, which is why we chose to use a standard library for the terminal side's implementation.

The representation of finite field elements differs in the CryptoInteger class on the BlackBerry and the BigInteger class that the Bouncy Castle library on the terminal is based on. While BigInteger is a signed variable CryptoInteger is unsigned, which has to be regarded during the conversion between those two types. The data is sent as byte arrays over the NFC communication link, where an element in $\mathbb{Z}_p$ is represented as an array of 20 bytes. Since the size of the byte-array representation of the integer values can be shorter than the designated 20 bytes, we pad with leading `0x00` when receiving an element.

The BlackBerry API 7.1.0. supports the chosen curve *secp160r1*, *i.e.* an implementation of the ECDH key agreement based on this curve is provided by the API. Yet, the BlackBerry API does not implement all functionality necessary for the implementation of the proposed e-cash schemes, and hence had to be extended. Point addition and doubling were implemented in Java making use of the modular arithmetic functionality provided in the BlackBerry API. The implementation method to execute the scalar multiplication efficiently is described in detail in the following. Note, the implementation is customized for curve *secp160r1*, but could easily be adapted to all other curves supported by BlackBerry API since version 3.6.0, which range from 160 to 571-bit curves.

5.3.2.1 Efficient Execution of EC Scalar Multiplication Using the ECDH Key Agreement

An implementation of the scalar multiplication $Q_k = [k]P$ in Java, making use of the modular arithmetic functionality provided in the BlackBerry API, leads to an execution time for the scalar multiplication of about 141 ms. Fortunately, the API contains an `ECDHKeyAgreement` class, which offers the method `generateSharedSecret`.

This method, which computes the scalar multiplication of an input point $P = (x_P, y_P)$ with an input scalar k, executes in 1 ms, but returns only the x-coordinate x_{Q_k} of the resulting point Q_k. In the protocols of the considered payment schemes multiple scalar multiplications and point additions have to be executed. Hence, knowledge of the y-coordinate y_{Q_k} of the resulting point is essential for further computations.

This drawback can be overcome. Starting from the short Weierstrass equation $y^2 = x^3 + ax + b$, over which the chosen elliptic curve is defined, the absolute value of y_{Q_k} can be computed as $|y_{Q_k}| = \sqrt{x_{Q_k}^3 + ax_{Q_k} + b} \mod p$. Algorithm 3.36 in [69] describes how to efficiently compute the square root in $\mathbb{Z}_p$, if $p \equiv 3 \mod 4$, which holds for the chosen curve. This results in two options for the resulting point Q_k, namely

$$Q_k^+ = (x_{Q_k}, +|y_{Q_k}|) \quad \text{and} \quad Q_k^- = (x_{Q_k}, -|y_{Q_k}|).$$

To choose the correct option Q_k^+ or Q_k^- for the resulting point Q_k we use the coherence $Q_{k+1} = [k+1]P = Q_k + P$. We executed the `generateSharedSecret` function on $(k+1)$ and P to compute Q_{k+1}. We further pick the positive result $Q_k^+ = (x_{Q_k}, +|y_{Q_k}|)$ and compute the x-coordinate of Q_{k+1}^+ as $Q_k^+ + P$, using the group law for point addition on the chosen elliptic curve [46]

$$x_{Q_{k+1}}^+ = \left(\frac{+y_{Q_k} - y_P}{x_{Q_k} - x_P}\right)^2 - x_P - x_{Q_k} \mod p. \tag{5.2}$$

We then check whether this result is equal to the x-coordinate of Q_{k+1} returned when executing the `generateSharedSecret` function on $(k+1)$ and P. While this algorithm, which is summarized as Algorithm 4.1, executes the `generateSharedSecret` method twice and computes a square root in the prime field $\mathbb{Z}_p$, it still achieves a major speed-up in execution time, when compared to the Java implementation based on the arithmetic functionality that is provided in the BlackBerry API, i.e. yields an execution time of around 4 ms.

Algorithm 5: Recovering the y-coordinate, when using the `ECDHKeyAgreement` class of the BlackBerry API to execute a scalar multiplication

Input : input point P, input scalar k
Output: x- coordinate and y-coordinate of resulting point $Q = [k]P = (x_Q, y_Q)$

$x_{Q_k} \leftarrow$ `generateSharedSecret`(k, P)
$x_{Q_{k+1}} \leftarrow$ `generateSharedSecret`$(k+1, P)$
$|y_{Q_k}| = \sqrt{x_{Q_k}^3 + a \cdot x_{Q_k} + b} \mod p$
$x_{Q_{k+1}}^+ = \left(\frac{+|y_{Q_k}| - y_P}{x_{Q_k} - x_P}\right)^2 - x_P - x_{Q_k} \mod p$
if $(x_{Q_{k+1}} == x_{Q_{k+1}}^+)$ **then**
  return $(x_{Q_k}, +|y_{Q_k}|)$
end
else
  return $(x_{Q_k}, -|y_{Q_k}|)$
end

5.4 Performance of E-cash with Attributes on an NFC-Smartphone

The time critical phases of the e-cash schemes are the withdrawal and especially the spending phase, as those have to be executed frequently, whereas the account opening only happens once for each user (or for example once per user per year, if new system keys are generated on a yearly basis, to increase security). We thus limit the discussion of our results to the time critical withdrawal and spending phase.

The results for the execution of the withdrawal phase for all schemes are presented in Table 5.8 whereas results for the spending of all schemes are presented in Table 5.9. We present four cases: i) Brands' e-cash scheme, when not allowing the encoding of attributes, ii) Abe's scheme, which does not allow the encoding of attributes, iii) Brands' scheme when allowing the encoding of two attributes and iv) ACL when allowing the encoding of two attributes. The spending results present timings for the case of revealing both encoded attributes. Note that the private key of the user is not counted as an attribute, *i.e.* a user encodes two attributes L_2 and L_3 additionally to his private key L_1. Of course, our implementation could support a bigger number of

Table 5.8. Execution times of withdrawal per coin for i) Brands without attributes, ii) Abe, which does not support the encoding of attributes, iii) Brands encoding two attributes and iv) ACL encoding two attributes.

Scheme	Terminal	Communication	Smartphone	Total
Brands (without attributes)	66.1 ms	45.1 ms	123.8 ms	235 ms
Abe	93.6 ms	69.6 ms	137.5 ms	301 ms
Brands (with attributes)	73.2 ms	44.1 ms	128.7 ms	246 ms
ACL (with attributes)	93.6 ms	69.9 ms	137.5 ms	301 ms

Table 5.9. Execution times of spending per coin for i) Brands without attributes, ii) Abe, which does not support the encoding of attributes, iii) Brands encoding two attributes and revealing both and iv) ACL encoding two attributes and revealing both of them.

Scheme	Terminal	Communication	Smartphone	Total
Brands (without attributes)	58.8 ms	96.8 ms	1.4 ms	157 ms
Abe	79.3 ms	81.0 ms	10.7 ms	171 ms
Brands (with attributes)	87.3 ms	114.8 ms	2.0 ms	204 ms
ACL (with attributes)	151.2 ms	221.4 ms	11.4 ms	384 ms

attributes if the transportation system required that, but keep in mind that there is a trade-off between the number of attributes and the spending time.

Figures 5.1 and 5.2 illustrate those results, where the execution times of the different protocols have been summarized to *Terminal*: all computation executed on the terminal side, *Communication*: execution time of the entire communication, and *Smartphone*: all computation executed on the BlackBerry smartphone. In our implementation all steps are executed serially, *i.e.* while waiting for the terminal the execution on the smartphone is suspended. This resembles the execution on a standard smart card. Due to the extended capabilities of the smartphone, computations on the smartphone and the terminal could be parallelized, which would lower the total execution time of the presented protocols. For example in the withdrawal protocol of ACL the bank could send the number rnd to the user right after generating it. Then, while the bank computes a, a'_1 and a'_2 the user could at the same time compute

z_1, ζ, ζ_1 and ζ_2. Thus, what is summarized as the total time is an upper limit of the execution time of the protocols.

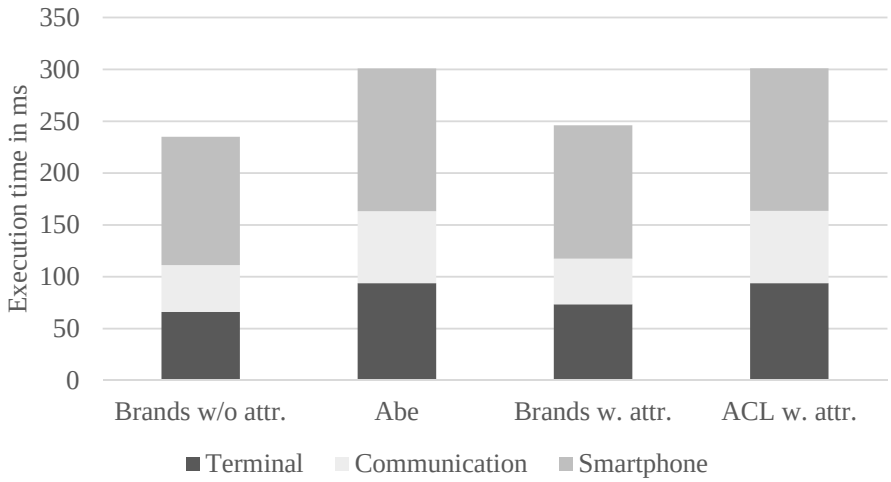


Figure 5.1. Illustration of the execution times of withdrawal per coin for i) Brands without attributes, ii) Abe, which does not support the encoding of attributes, iii) Brands encoding two attributes and iv) ACL encoding two attributes.

An advantage of the ACL scheme is that for the revealing attributes process (second part of Table 5.7) the values $\mathsf{C}', \widetilde{\zeta_1}, \widetilde{\mathsf{C}'}$ can be precomputed, which has been realized for the implementation at hand. The computation time for those precomputed values is 39 ms and is not included in the presented results. By doing so the total execution time for spending a coin of all schemes does not exceed 400 ms, which is close to the acceptance threshold for spendings in the transportation domain, which is 300 ms [81]. The results show that it is feasible to spend a coin meeting the extreme time constraints of the transportation setting. Spending several coins serially exceeds those time constraints. Yet, the execution time could further be reduced by batching the executions that are required for spending several coins.

While the terminal side is represented by a powerful computer, in our experimental set-up the execution of a scalar multiplication on the terminal takes longer than on the BlackBerry device; on the BlackBerry an execution of the point multiplication takes 4 ms, whereas on the computer it takes 6 ms. As mentioned in Section 5.3.2

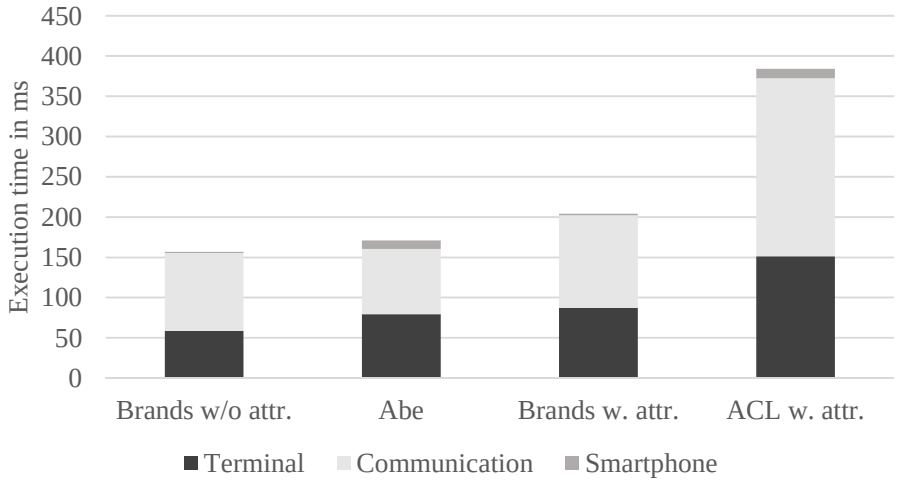


Figure 5.2. Illustration of the execution times of spending per coin for i) Brands without attributes, ii) Abe, which does not support the encoding of attributes, iii) Brands encoding two attributes and revealing both and iv) ACL encoding two attributes and revealing both of them.

we focus on the execution time on the payment device and on the communication overhead. The implementation results could be improved when not relying on a Java implementation for the terminal side, which is a realistic scenario, since the transportation authority has full control over vending machines and turnstiles.

Surprisingly, a limiting factor for the execution of the different protocols is the card emulation mode supported by the BlackBerry device. The communication bandwidth is limited to 106 kbits/s, while the maximum bandwidth supported by the NFC standard is 424 kbits/s. An additional deceleration limits the practical bandwidth on the application layer to 62.5 kbit/s. Since the communication plays an integral part in the execution of the spending protocol, a faster communication could significantly improve the execution timings. Moreover, the length of an APDU is limited to 256 bytes, which is why for some protocol steps data had to be split up into multiple APDUs. Hence, the overall execution time could be improved when allowing longer APDUs.

Table 5.10. Coin size (per coin data stored on user device) for the cases i) Brands without attributes, ii) Abe, which does not support the encoding of attributes, iii) Brands encoding two attributes and iv) ACL encoding two attributes.

Scheme	**Coin Elements**	**Coin Size in bytes**
Brands (without attributes)	$A, B, z', a', b', r', s, x_1, x_2$	289
Abe	$\zeta, \zeta_1, \rho, \omega, \rho'_1, \rho'_2, \omega', \tau, \gamma$	229
Brands (with attributes)	$A, B, z', a', b', r', s, x_0, x_1, x_2, x_3$	331
ACL (with attributes)	$\zeta, \zeta_1, \rho, \omega, \rho'_1, \rho'_2, \omega', \tau, \gamma, rnd, rnd', C', \widetilde{\zeta_1}, \widetilde{C'}$	394

A further observation of Brands' with attributes spending protocol is that the number of equations that have to be computed on the user device decreases with the number of attributes that are revealed. Yet, at the same time the communicated data increases. For our implementation the increase in time for the communicated data dominates the decrease in processing time. This could be different for other platforms, where the communication plays a less important role in the overall execution time of the protocol.

Table 5.10 shows the coin size for each of the schemes, i.e. the data that needs to be stored on the user device for each coin. Since for the ACL scheme $\mathsf{C}', \widetilde{\zeta_1}, \widetilde{\mathsf{C}'}$ have been precomputed, they need to be stored together with rnd' on the device as part of the coin. If storage space would be more critical in comparison to the execution time, those values could be computed on-the-fly when spending a coin, which would lead to the same storage amount for a coin as in Abe's scheme, but longer execution times.

In the following we estimate the database requirements for our e-cash based transportation payment system. We base our estimations on the Massachusetts Bay Transportation Authority (MBTA) system. The MBTA reports an average ridership of 1.28 million trips per day for February 2013 [67].

In the case of Brands' e-cash with attributes scheme the bank needs to store the values $A, d, (R, r_1, \ldots, r_n)$ for each coin in the database, in order to detect double-

spending at a later point in time. Assuming the encoding of two attributes, this results in 146 bytes per coin and an average of 178 MB per day that need to be stored in the database.

In case of ACL the bank has to have two databases. One stores the values z_1 together with the public key $\mathsf{pk}_{\mathcal{U}}$ of a user for each withdrawn coin and another one that stores $\zeta_1, desc, \varepsilon_p$ and μ_p for each spent coin. This results in 186 bytes per coin and an average of 227 MB that need to be stored daily in the database assuming an average of 1.28 million trips per day.

Managing large databases does not primarily depend on the number of data records and the size of the related storage. It greatly depends on the complexity of the *search-and-join* algorithm. In our case the database just has to operate with primary-key related search requests – in the case of Brands' scheme it is the 21 byte value A. Executing the *search-and-join* algorithm to add a set of 1.28 million data records to a database that has 1 billion entries should be executable within a couple of minutes.

5.5 Implementation of E-cash with Attributes on a MULTOS Smartcard

This section arose from collaborative work with Felix Riek, during his master thesis, which was supervised by the author of this dissertation.

We demonstrated that both schemes execute efficiently on the chosen NFC-smartphone, achieving execution times that suffice to be applied in the public transportation domain. It is however highly desirable to execute payment applications on secure hardware, such as the SIM card or the embedded secure element of a smartphone, instead of a smartphone's main processor. These devices are usually equipped with smartcard-like chips. We thus further evaluate the performance of the considered schemes on an off-the-shelf smartcard.

MULTOS International Pte Ltd. is one of the leading providers of technology and solutions for the smartcard industry [74]. The consortium provides MULTOS modules, or chips to card manufacturers. Their application ranges from banking, over transit payments to government ID. The hardware of MULTOS cards may differ for different types of MULTOS cards, depending on the chip manufacturer, but it has to be ensured that it supports a specified MULTOS operating system. Communication with the card and provision of the virtual machine, which handles code execution on the card as well as memory management and loading and deletion of applications, are provided by the MULTOS operating system [66]. It is claimed by the MULTOS consortium that they provide "the world's most secure multi-application smartcard operating system" [74]. A multi-application smartcard has many benefits, as it can host multiple applications in a single device. It can thus combine several payment and authentication services, such as transit payments, credit card payments and government ID in a single card. Besides security the focus lies on an open access, which is why those cards were chosen for this work. In particular we use UbiVelox UBM 21-Z48 cards as the API of these MULTOS smartcards support a variety of cryptographic primitives, in particular elliptic curve cryptography, allowing the implementation of advanced cryptographic protocols on them.

To allow the secure execution of multiple applications on the card, the operating system provides an application separating firewall, ensuring that applications on the card cannot interfere with each other. This firewall provides each application with memory area for code, data and session data. The firewall ensures that the code memory space cannot be read or written by another application. Application data, which has to persist in memory over multiple executions of the application is stored in data memory. Code and data are stored in EEPROM of the chip. Session data, which is only needed within an execution of a specific application, is held in the much faster RAM on the card. Information is exchanged between reader and card

using Application Protocol Data Units (APDU). A reader sends a command APDU to which the card answers with a corresponding APDU response. A command APDU has a 4 byte header. This header encodes a class byte CLS, which selects the class within an instruction, an instruction byte INS, which selects the instruction within a class and two parameter bytes P1 and P2, which specify parameters within the instruction. It further has an optional body consisting of a byte Lc, which specifies the length of the command data that follows, the command data itself and a byte Le, which indicates the expected length of the cards response.

We used a UBM21-Z48 developer card manufactured by UbiVelox for our analysis. This card, which supports MULTOS version 4.2.1, contains an ST23ZL48 chip from STMicroelectronics[4]. The ST23ZL48 chip has a 8/16-bit ST23 CPU core, 300 kB of ROM, 6 kB of RAM, and 48 kB of EEPROM. The CPU can be operated at frequencies of up to 27 MHz. It further has an AIS-31 class P2 compliant true random number generator and the enhanced NESCRYPT crypto-processor [97]. The coprocessor provides high-performance native support for GF(p) and GF(2^n) arithmetic and includes dedicated instructions to accelerate execution of the SHA-1 and SHA-2 family of hash functions. It thus allows high-performance implementations of public-key cryptosystems [96]. We chose this card due to its elliptic curve cryptography (ECC) support, on which we base the implemented e-cash schemes.

MULTOS requires that the function `getRandomNumber` is supported by its approved platforms but leaves its implementation to the card manufacturer. While the NESCRYPT coprocessor offers true random number generation, we do not know, whether this is used to generate random numbers directly, or whether it is used as seed for a pseudo random number generator (PRNG). We checked the statistical properties of the generated random numbers on the card using the National Institute

[4] `http://www.multos.com/products/approved_platforms/`

of Standards and Technology's test suite [85]. We generated a 126 MB random number sequence and ran all tests on it. Based on these tests, we found no violations against true random behavior irrespective of the test. We conclude that the statistical properties of the generated random numbers are good.

5.5.1 Framework Implementation

In our implementation the terminal, which is represented by an HP ProBook 650G1 notebook with an Intel i5-4200M processor having 8 GB RAM and running Windows 7 (64 Bit), represents the bank as well as the shop. Communication is executed with the notebook's integrated Alcor Micro Smart Card Reader. The terminal implementation is based on Java and was developed using the Eclipse Luna Service Release 1. While a C implementation would lead to faster execution times, our main focus is on evaluating the execution time of the computation on the smartcard side. Terminals could easily be equipped with stronger or dedicated hardware, greatly reducing the execution time on the terminal. Java provides the *javax.smartcardio* package allowing a comfortable implementation of communication with smartcards via APDUs. This package provides an API to set up connection with the smartcard reader as well as sending APDUs to and receiving from a smartcard. For the implementation of cryptographic functionality we relied on the Bouncy Castle library version 1.5.1. This library provides a great variety of cryptographic functionality, including the implementation of elliptic curve cryptography and hash generation. We further relied on the MySQL 5.5 database management system to log the execution times of each protocol.

We developed code for the smartcard using the MULTOS SmartDeck application development system for MULTOS cards and loaded the application to the card using MULTOS Utility (MUtil). SmartDeck contains the required compiler, a generator to generate Application Load Units (ALUs) and a debugger for code development. A

developer key is required to load the application to the card. Code for the card can be written directly in MULTOS Assembly Language (MAL) or in a higher level language (C or Java) which is then translated by the compiler. We choose C as programming language, since the set up is easier than the one for Java and code can be written much more efficiently than in MAL. While more efficient code could be written in MAEL we anticipate that the speed-up would be limited as we mostly relied on functionality provided by the card's API.

Our implementation is based on the Brainpool elliptic curves at various security levels. Those curves, which were proposed by the Brainpool consortium, are for example implemented in the German identity card and have recently been standardized for use in TLS [65]. While we measured execution times for elliptic curves of four security levels, namely brainpoolP160r1 (reaching a security level comparable to 80-bit symmetric security), brainpoolP192r1 (reaching a security level comparable to 96-bit symmetric security), brainpoolP224r1 (reaching a security level comparable to 112-bit symmetric security), and brainpoolP256r1 (reaching a security level comparable to 128-bit symmetric security), we only present implementation results for brainpoolP160r1 and brainpoolP256r1, to make our results comparable to the previous results of this and the preceding chapter.

5.6 Performance of E-cash with Attributes on a MULTOS Smartcard

We first present the execution times of both schemes on the chosen platform and then give estimates for the number of coins that could be stored in the chosen smartcard. Our analysis targets the evaluation of the execution times of the withdrawal and spending protocols. While registration is executed only once, withdrawal and spending are executed repeatedly over a period of time, and thus greatly determine

Table 5.11. Execution times of the withdrawal protocols of Brands' untraceable offline cash scheme [21] and Anonymous Credentials Light (ACL) [7] for the case of encoding two attributes into a coin based on the brainpoolP160r1 (160-bit) and the brainpoolP256r1 (256-bit) elliptic curves.

Scheme	Terminal	Communication	Card	Total
Brands (160-bit)	118 ms	194 ms	2,114 ms	2,426 ms
ACL (160-bit)	158 ms	360 ms	2,430 ms	2,948 ms
Brands (256-bit)	179 ms	279 ms	2,996 ms	3,454 ms
ACL (256-bit)	204 ms	544 ms	3,438 ms	4,186 ms

user acceptance. We obtained the execution times by executing the protocols 100 times and averaging the results.

The execution times for the withdrawal protocol for Brands and ACL encoding two attributes into coins are presented in Table 5.11. We present the times for computations on the card, computations on the terminal, the communication overhead, and the total execution time. Results are presented at two security levels, namely at the 80-bit security level and the 128-bit security level.

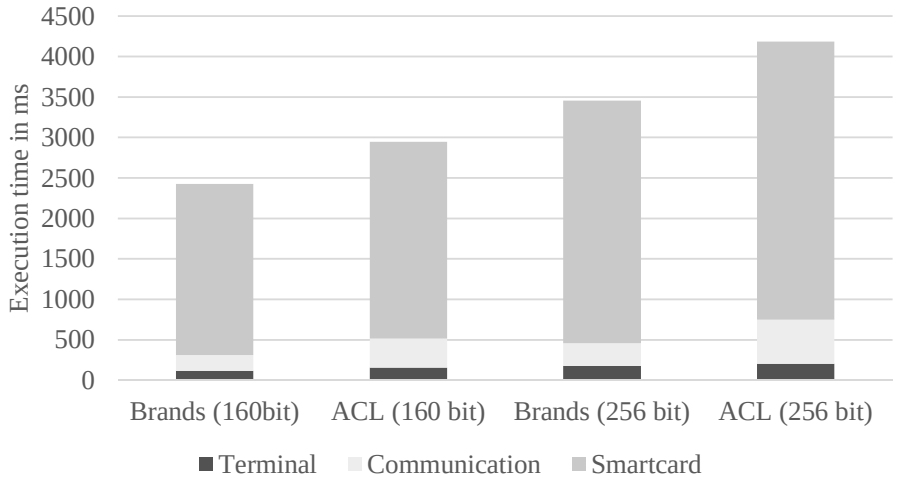


Figure 5.3. Execution times of Brands' and ACL withdrawal protocols at the 80-bit and 128-bit security levels, encoding two attributes (illustration of the results presented in Table 5.11).

As mentioned above an 80-bit security level should be sufficient for the deployment of a low-value payment scheme, and thus the withdrawal of a coin can be executed in

under 2.5 seconds and 3 seconds respectively. While for both protocols the computation on the card clearly presents the bottleneck in the scheme, we observe a noticeable difference between the execution times of the two withdrawal protocols. ACL requires the communication of a greater number of elements as well as the more computation of arithmetic in G on the card. Please note however, that ACL has a full prove of security, which comes at the expense of only slightly less efficiency. We would further like to emphasize that the withdrawal process is not time critical and could be executed at home. A user could be provided with a desktop banking application, which would allow him to charge his payment device using his personal smartcard reader. He could start the charging process and could leave the card inserted in the reader until the withdrawal process is completed. Even if this takes multiple minutes this could be acceptable for users.

The timings for spending a coin, not revealing an attribute, are presented in Table 5.12. Note that in our study we present timings of the spending protocols without revealing attributes only. In that case 9 exponentiations and 3 comparisons in G have to be executed for Brands' spending protocol on the terminal, while only 8 exponentiations in G and 3 comparisons in $\mathbb{Z}_q$ have to be executed on the terminal for the ACL scheme, leading to lower computation times on the terminal. Additionally, in Brands' spending protocol the communication is split in two parts, and the communicated data is 5 elements in G and 5 elements in $\mathbb{Z}_q$, while ACL requires only the communication of 2 elements in G and 7 elements in $\mathbb{Z}_q$, thus leading to lower communication times for ACL. For the 256-bit curve based Brands' spending protocol the communicated data needs to be split into a greater number of APDUs thus leading to proportionally higher communication times. Overall, when not revealing an attribute both spending protocols can be executed within 800 ms when relying on a 80-bit security level.

Table 5.12. Execution times of the spending protocols of Brands' untraceable offline cash scheme [21] and Anonymous Credentials Light (ACL) [7] based on brainpoolP160r1 (160-bit) and brainpoolP256r1 (256-bit) elliptic curves for the case that two attributes are encoded in a coin but no attribute is revealed.

Scheme	**Terminal**	**Communication**	**Card**	**Total**
Brands (160-bit)	110 ms	433 ms	259 ms	769 ms
ACL (160-bit)	76 ms	310 ms	407 ms	793 ms
Brands (256-bit)	257 ms	660 ms	297 ms	1,063 ms
ACL (256-bit)	92 ms	466 ms	510 ms	1,068 ms

If an attribute was revealed during the spending, we would save on one modular multiplication and addition on the card side for Brands' protocol, which would then have to be executed on the terminal side, but would have to send one more element in $\mathbb{Z}_q$. This would only slightly increase the communication time. By measurements we figured out that the communication of one byte requires 1.18 ms plus an initial 26 ms for the initialization of an APDU. The communication would thus increase by less then 50 ms. If an attribute was revealed during the spending of ACL's scheme, 7 random numbers in $\mathbb{Z}_q$ would have to be generated, 9 exponentiations in G, 14 modular multiplications and 6 modular additions in $\mathbb{Z}_q$, as well as one hash function would have to be computed. Additionally, 3 elements in G and 7 elements in $\mathbb{Z}_q$ would have to be sent to the terminal, resulting in an increase in communication time of about 330 ms, when relying on the 160-bit elliptic curve. Thus when revealing an attribute the spending of the ACL scheme would be outperformed by Brands' spending protocol.

Those results demonstrate that keeping the number of encoded attributes to a minimum is required to reach an acceptable execution time. However, we would like to emphasize that in the 160-bit implementation an attribute can encode 160 bits of data. One could hence encode multiple attribute values into one coin attribute. For example two attributes of 64 bits and another one of 32 bits could be encoded in one

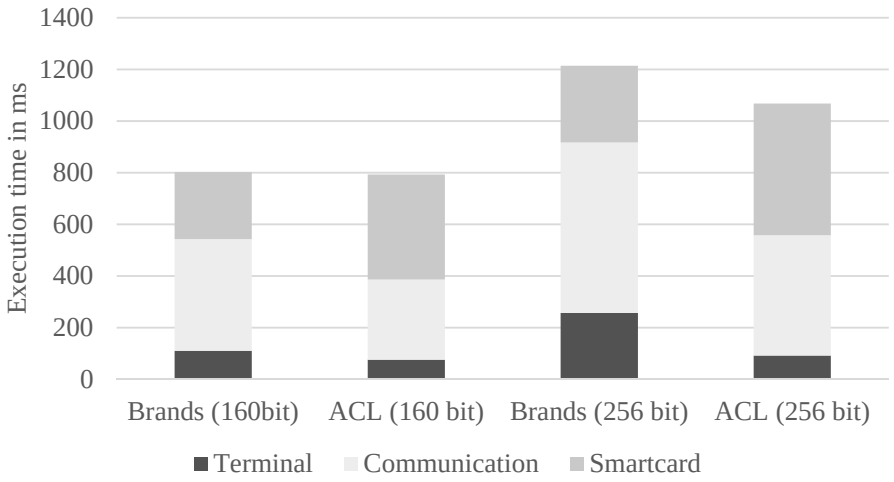


Figure 5.4. Execution times of Brands' and ACL spending protocols at the 80-bit and 128-bit security levels, encoding two attributes but revealing no attribute (illustration of the results presented in Table 5.12).

coin attribute. This would greatly limit the number of attributes that would have to be encoded into a coin.

While the EEPROM size of the chip integrated in the card is given as 48 kB, we found out that only 44.7 kB are available for writing on the used card. The implementation of Brands' protocol requires 7 kB of code space, leaving 37.7 kB available for storing electronic coins. A coin in Brands' protocol, encoding two attributes, requires storing 5 elements in G and 6 elements in $\mathbb{Z}_q$. For the 160-bit implementation this adds up to 320 bytes per coin, while 512 bytes are required for storing a coin based on the 256-bit implementation. Next to other data, such as the keying material and so on, this allows to store around 110 of Brands' coins on the card, when relying on a 160-bit implementation and 65 when relying on the 256-bit implementation, if no other application would be implemented.

ACL requires 8.4 kB of code space, thus leaving 36.3 kB available for storing electronic coins. Storing coin data in ACL requires storing 2 elements in G and 8 elements in $\mathbb{Z}_q$. For the 160-bit implementation this adds up to 240 bytes that have to be stored on the card per coin, while 384 bytes have to be stored per coin using

the ACL scheme based on a 256-bit implementation. Thus around 150 of ACL coins can be stored on the card, when relying on the 160-bit key size, while only 90 coins can be stored when relying on the 256-bit key size, if the card is solely used for this payment application.

5.7 Discussion

Brands' [21] and ACL [7] e-cash schemes, allowing the encoding of attributes are suitable payment schemes for use in public transport, due to their efficiency during the spending phase. While the ACL scheme is a little less efficient than Brands' with attributes, it is the only practical e-cash with attributes that comes with a formal proof of security. This chapter presented a full implementation of Brands' with and without attributes, Abe's and ACL on a BlackBerry Bold 9900. We proposed a method that allows the use of the `ECDHKeyAgreement` class of the BlackBerry API to compute the result of a scalar multiplication on an elliptic curve, by recovering the y-coordinate of the resulting point, which led to transaction times that meet real-world requirements of transportation payment systems for all considered schemes. Surprisingly, a limiting factor of the transaction is the NFC communication bandwidth.

We further presented an implementation of both e-cash schemes on an off-the-shelf MULTOS smartcard. Due to their support of elliptic curve cryptography and open API access, we used the UBM21-Z48 developer cards from UbiVelox for our investigations. We implemented both schemes in a way that the encoding of two attributes was allowed, while we did not reveal them during spending. We however give estimates of the execution times, when allowing revealing an attribute. Our results are promising. When relying on an implementation, of which the security is comparable to 80-bit symmetric security, which we consider sufficient for low-value payment schemes, a spending can be executed within 800 ms for both schemes, while

the cards would allow storing 110 and 150 coins respectively for both schemes. We further presented execution times at a high security level, namely comparable to 128-bit symmetric security, and show that the performance of our implementation is acceptable for high-security applications.

CHAPTER 6

LIGHTWEIGHT ANONYMOUS NFC-PAYMENTS FOR E-MOBILITY

This chapter presents results that arose from collaborative work with Olga Korobova, Andy Rupp and Sven Schäge. Results that were not developed by the author of this dissertation will be clearly marked in the respective sections.

The use of electric mobiles has many benefits in particular in densely populated areas. Electric cars avoid local emissions, decrease noise pollution, allow transport to partially be based on renewable energies, and can achieve the independence of transport from petroleum, thus ensuring the long-term availability of transport.

While this makes e-mobility sound like a promising solution, especially in densely populated areas, there are many difficulties yet to be solved. An important one is the provision of a charging infrastructure for electric mobiles and a payment system to pay for charged energy. Relying on electronic payments seems unavoidable due to the great distribution of charging stations and the associated high maintenance cost of cash payments. Yet, especially in the e-mobility domain, revealing a user's identity during a payment greatly infringes his location privacy, as the location of a user's car is closely connected to the location of the user himself [77]. It was recently found out that privacy is a key factor for user acceptance of e-mobility [34].

In this chapter we analyze the use of lightweight e-cash schemes in the e-mobility domain. A scheme, well-known for its efficiency during the spending phase, which has also been analyzed for its application in the public transport domain in the preceding chapters, is Brands' untraceable offline cash scheme [21]. However, a major drawback

of this efficient electronic cash scheme is the difficulty of realizing change in a privacy-preserving way.

Chapter 4 presented a performance analysis of the privacy-preserving pre-payments with refunds scheme (P4R) [87] to be used in the public transport domain based on a practical implementation. Rather then relying on a user being able to receive electronic coins as change, this scheme realizes a privacy-preserving refund system. In P4R a user pays using an electronic coin, and if the coin value exceeds the value of his fare, the user receives a refund in a privacy-preserving way, which he can redeem at the bank at a later point in time. To increase privacy and limit the required storage space, a user accumulates several refunds in a single refund token.

We now analyze, how P4R can be used to solve the issue of privacy in the domain of e-mobility payments. Using the same payment scheme for public transportation payments and payments in the e-mobility domain allows using the same payment service for both types of transport, which presents a key factor in achieving customer convenience. We first describe a charging infrastructure in the e-mobility domain and identify which security and privacy objectives should be satisfied in this setting. We then present our scheme, which extends P4R to allow the encoding of user attributes into electronic coins. A peculiarity of payments in the e-mobility domain is the fact that a charging process takes at least half an hour in which a user should not be forced to stay at his car. It must however be ensured that the user cannot leave the charging station without paying for the received charge and that an attacker cannot interfere with the charging process during the user's absence. We show, how this is solved by our system. We further analyze our proposed scheme, discussing its performance and informally describing which security and privacy objectives can be met.

6.1 Related Work

The first work on privacy-preserving payments in the e-mobility domain has been presented by Liu *et al.* in [64]. Our approach is different in that we try to limit the payment device's computation required during the spending to a minimum. This is beneficial in scenarios, where even extremely constrained payment tokens such as smartcards are supported. We emphasize that, even if a user pays using her mobile phone, a payment application should be executed on a trusted device such as the smartphone's secure element, which usually has the same computational capabilities as a smartcard's processor. We build on the privacy-preserving pre-payments with refunds scheme proposed by Rupp *et al.* for use in public transportation systems in [86]. While it is extremely desirable to allow for the same payment scheme to be applicable in several scenarios, there are distinct differences between public transportation payment systems and payments in the e-mobility domain, which we will point out in this chapter. A very different approach to reaching privacy in the e-mobility domain has been studied in [36]. In their billing payment scheme the authors preserve users' location privacy by obscuring the identity of a charging station during a billing process instead of obscuring a user's identity.

6.2 E-mobility Payment Systems

In Figure 6.1 we present an overview of a charging infrastructure for e-mobility. We differentiate between payment service providers, who process and check the correctness of payments, and electricity providers, who provide a charging station infrastructure, *i.e.* are in possession of charging stations and provide electric energy. By allowing for independence of electricity providers and payment service providers, a widespread charging infrastructure can be provided to e-mobility users, not limiting their driving range to the charging station coverage of a local electricity provider.

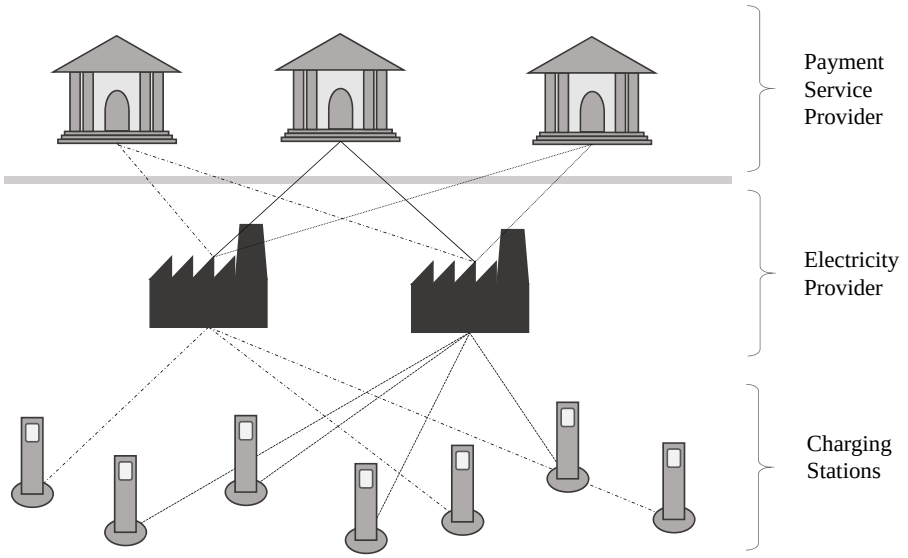


Figure 6.1. Overview of a charging infrastructure for e-mobility. We differentiate between payment service providers, who process and check the correctness of payments, and electricity providers, who provide a charging station infrastructure, *i.e.* are in possession of charging stations and provide electric energy.

A payment service provider can either offer some kind of billing payment system, where a user is billed after having received electric energy from a charging station, or a prepaid system, where a user receives payment tokens of specific values that he can later use in order to pay for electric energy at a charging station. In this work, we concentrate on the prepaid system, as this can be constructed based on e-cash schemes, protecting a user's privacy by design.

In order to be able to pay at a charging station, a user registers with a payment service provider, deposits money, and receives a bundle of payment tokens in exchange, which he stores in his electronic wallet. Tokens of several values can be offered, such that at a charging station the user can choose to pay with a suitable payment token depending on the amount of charge he wants to receive. The user would pay for those tokens using his credit card, or would set up a service that would directly withdraw the money from his bank account.

We allow the encoding and selective disclosure of user attributes into payment tokens. As described in the previous chapter, this feature supports participation in discount programs and further supports "privacy-preserving" data-mining. An important benefit of attributes that can be selectively disclosed is that a user can decide during the spending, if he wishes to disclose an attribute or not.

At a charging station the user prepays for the electric energy he will receive using one of his payment tokens. The charging station can monitor the current price for electricity and the car can monitor the amount of required charge, such that the user can choose a suitably valued payment token. The user inserts a charging cable into the plugs at the charging station and his car respectively, and starts the charging process. The charging station will lock its screen, such that the user can leave the scene during the charging process. During this time his charging cable is locked to the car and the charging station such that nobody can steal it.

The charging station will charge the car until the price of the charge reaches the value of the payment token, until the user cancels the charging process, or until the car is fully charged. In case the token's value is reached, the charging station stops charging, but does not release the charging cable. Only if the user authenticates himself with the same payment token again, he will be able to finalize the charging process, which includes releasing the charging cable. In case the charging process is stopped before the received energy reaches the value of the payment token, he receives a refund based on his overpayment.

6.3 Security and Privacy Objectives

The focus of this chapter lies on presenting practical performance results of a privacy-preserving payment scheme suitable for the e-mobility domain, which is based on P4R but additionally supports the encoding of user attributes into payment tokens. We only informally discuss the desired security and privacy objectives. A formal

model and full security proof is left as future work. Informally speaking, an e-mobility payment scheme should satisfy the following security and privacy objectives:

I. A user must not lose money. Summarizing all charging processes of a user, she should not pay more than the cost for the overall received electric energy, *i.e.* the overall received electric energy $\sum v_{\mathsf{EE}_i}$ and all received refunds $\sum v_{\mathsf{RF}_i}$, minus the money a user deposited for all used payment tokens $\sum v_{\mathsf{PT}_i}$ should be greater or equal to zero: $\sum v_{\mathsf{EE}_i} + \sum v_{\mathsf{RF}_i} - \sum v_{\mathsf{PT}_i} \geq 0$.

II. A user's location privacy must be preserved, *i.e.* the process of executing payments or collecting refunds must not reveal her identity.

III. A payment service provider must not lose money. It should thus hold that $\sum v_{\mathsf{PT}_j} - \sum v_{\mathsf{DP}_j} - \sum v_{\mathsf{RF}_j} \geq 0$, *i.e.* the sum of all money received from users in exchange for payment tokens $\sum v_{\mathsf{PT}_j}$, minus the money deposited to electricity providers' bank accounts $\sum v_{\mathsf{DP}_j}$, minus all refunds paid to users $\sum v_{\mathsf{RF}_j}$ must not be smaller than zero.

IV. An electricity provider must not lose money, *i.e.* the value for electric charge that he provided to users $\sum v_{\mathsf{EE}_k}$ should not be greater than the overall money that was credited to his bank account $\sum v_{\mathsf{DP}_k}$: $\sum v_{\mathsf{DP}_k} - \sum v_{\mathsf{EE}_k} \geq 0$

6.4 Payment Scheme Description

In contrast to the public transport domain where the transportation authority controls vending machines, which give out payment tokens and redeem refunds, and turnstiles, which accept payments and give out refunds, in the e-mobility domain we differentiate between payment service providers ($\mathcal{P}$) and electricity providers ($\mathcal{E}$). Charging stations belong to an electricity provider, whereas payments are controlled by a payment service provider. In order to realize a modular system, which in specific can function across country borders, we do not assume trust between those entities.

However, we do assume that an electricity provider has a constant communication link to a payment service provider's server, which is equipped with the secret BLS signature key, and can be queried to add a refund to a refund token. We do not assume that the payment service provider's database can be queried in real-time to check whether a coin has been spent before. This is because the payment service provider's coin database presents a bottleneck in the system. One can equip multiple servers with keying material to check payments and add refunds, but there can only be one database to store and check multiple occurrences of payment transcripts. Instead a *double-spending check* is used to identify cheating users, which can be executed at any point in time.

In [86] it is assumed that users ($\mathcal{U}$) do not check, whether they received a correct refund. This can be assumed for low refund values as they usually occur in public transportation systems, it however does not hold for the e-mobility domain. In our system a user receives a refund token, for which he can prove that it correctly encodes the pzyment service provider's private BLS signature key, *i.e.* that it is a valid refund token. For each received refund he will check whether it was added correctly. This check further ensures, that the refund token remains valid.

We rely on Schnorr's signature scheme [92] to generate receipts rc. Given a generator $g \in G$, a private key $\mathsf{sk} \in \mathbb{Z}_q^*$ and a corresponding public key $\mathsf{pk} = g^{\mathsf{sk}}$ a signature $\sigma(\mathsf{msg}) = (y, k)$ on a message msg is generated by choosing a random value $r \in \mathbb{Z}_q$, and computing $k = \mathcal{H}(\mathsf{msg}, g^r)$ and $y = r + \mathsf{sk} \times k$. The signature can be verified through

$$\mathcal{H}(\mathsf{msg}, \frac{g^y}{\mathsf{pk}^k}) \stackrel{?}{=} k. \tag{6.1}$$

6.4.1 System Setup

We base the scheme on an asymmetric bilinear pairing $G_1, G_2 \rightarrow G_T$, where G_1 and G_2 have order q. Generators $g, g', g_0, \ldots, g_n \in G_1$ and $h \in G_2$ are selected, where

n is the number of attributes that will be encoded in a payment token, and a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q^*$ is chosen.

The payment service provider's secret payment token system key is $\mathsf{sk}_{\mathcal{P}} \in_{\mathcal{R}} \mathbb{Z}_q$ with corresponding public key $\mathsf{pk}_{\mathcal{P}} = g^{\mathsf{sk}_{\mathcal{P}}}$. If payment tokens of different values are supported the payment service provider generates several secret and public key pairs, one for each value of the tokens. He further sets up the refund token system with a private BLS signature key $\mathsf{sk}_{\mathcal{P}_{\mathsf{BLS}}} \in_{\mathcal{R}} \mathbb{Z}_q$ and corresponding public BLS signature key $\mathsf{pk}_{\mathcal{P}_{\mathsf{BLS}}} = h^{\mathsf{sk}_{\mathcal{P}_{\mathsf{BLS}}}}$. He then generates a list in which he lists refund check values $\mathsf{RC}_i = h^{\mathsf{sk}^{2^i}_{\mathcal{P}_{\mathsf{BLS}}}}$ for $i = 1 \ldots \lfloor \log_2 v_{\mathsf{PT}} \rfloor$, where v_{PT} is the value of a payment token. Note that no separate refund check value is required for $i = 0$ as this is equal to the public BLS signature key $\mathsf{RC}_0 = \mathsf{pk}_{\mathcal{P}_{\mathsf{BLS}}}$. If the unit is cents and $v_{\mathsf{PT}} = \$10$, then the list has 9 entries.

If an electricity provider wants to use the payment service provider's service, he registers and receives $\mathsf{ID}_{\mathcal{E}}$. He then generates a private and public key pair $\mathsf{sk}_{\mathcal{E}} \in \mathbb{Z}_q, \mathsf{pk}_{\mathcal{E}} = g^{\mathsf{sk}_{\mathcal{E}}}$ and proves knowledge of his secret key to the payment service provider.

Table 6.1. List of refund check values

Refund value	RC_i
$2^1 = 2$ ¢	$\mathsf{RC}_1 = h^{\mathsf{sk}^2_{\mathcal{P}_{\mathsf{BLS}}}}$
$2^2 = 4$ ¢	$\mathsf{RC}_2 = h^{\mathsf{sk}^4_{\mathcal{P}_{\mathsf{BLS}}}}$
$\vdots$	$\vdots$
$2^9 = 512$ ¢	$\mathsf{RC}_9 = h^{\mathsf{sk}^{512}_{\mathcal{P}_{\mathsf{BLS}}}}$

6.4.2 User Registration

During registration a user generates a secret key $\mathsf{sk}_{\mathcal{U}} \in_{\mathcal{R}} \mathbb{Z}_q$, computes a public key $\mathsf{pk}_{\mathcal{U}} = g_0^{\mathsf{sk}_{\mathcal{U}}}$, and commits to the attributes $L_1, \ldots, L_n$ in a commitment C. He proves knowledge of his secret key and that he correctly encoded his attributes and secret key into C to the payment service provider. We assume that the payment service

provider controls, whether a user is eligible to participate in a discount program, and whether the user can participate in a privacy-preserving data-mining system and thus checks, which attributes a user can include in his commitment. A user thus reveals his attributes to the payment service provider, who can compute C on his own (using a user's public key). The payment service provider stores identifying information of a user together with the user's public key and commitment C in his user database. A user stores his key pair, the commitment C and the value z he received from the payment service provider. The registration process is shown in Table 6.2, where ts denotes a time stamp.

Table 6.2. P4R for e-mobility user registration protocol.

$\mathcal{P}$		$\mathcal{U}$
	$\xrightarrow{\mathsf{pk}_{\mathcal{P}},\mathsf{pk}_{\mathcal{P}_{\mathsf{BLS}}},\mathsf{RC}_0,\ldots,\mathsf{RC}_m}$	$\mathsf{sk}_{\mathcal{U}}, w_0 \in_{\mathcal{R}} \mathbb{Z}_q^*,\ \mathsf{pk}_{\mathcal{U}} = g_0^{\mathsf{sk}_{\mathcal{U}}}$ $\mathsf{C} = \mathsf{pk}_{\mathcal{U}} g_1^{L_1} g_2^{L_2} \ldots g_n^{L_n}$, If $\mathsf{C} \neq 1$: $\mathsf{pk}'_{\mathcal{U}} = g_0^{w_0}$ $k = \mathcal{H}(\mathsf{C}, \mathsf{pk}_{\mathcal{U}}, \mathsf{pk}'_{\mathcal{U}}, ts)$
$k = \mathcal{H}(\mathsf{C}, \mathsf{pk}_{\mathcal{U}}, \mathsf{pk}'_{\mathcal{U}}, ts)$ $\mathsf{C} \stackrel{?}{=} \mathsf{pk}_{\mathcal{U}} g_1^{L_1} g_2^{L_2} \ldots g_n^{L_n}$ $\mathsf{pk}_{\mathcal{U}}^k \mathsf{pk}'_{\mathcal{U}} \stackrel{?}{=} g_0^{s_0}$	$\xleftarrow{\mathsf{C},\mathsf{pk}_{\mathcal{U}},\mathsf{pk}'_{\mathcal{U}},ts,L_1,\ldots,L_n,s_0}$	$s_0 = w_0 + k \times \mathsf{sk}_{\mathcal{U}}$
$z = (\mathsf{C}g')^{\mathsf{sk}_{\mathcal{P}}}$	$\xrightarrow{z}$	Store z

6.4.3 (Re-) Charge Electronic Wallet

To receive payment tokens a user authenticates to the payment service provider and claims how many payment tokens he would like to receive by signing a message containing the number N_{PT} of payment tokens he would like to receive and a time stamp ts (cf. upper part of Table 6.3).

For each token the user and the payment service provider engage in Brands' with attributes withdrawal protocol adapted to allowing two showings of a coin without revealing the user's identity. This is presented in the lower part of Table 6.3. During this process the user generates the serial number tag A of the coin and two

Table 6.3. P4R for e-mobility (re-) charge electronic wallet protocol.

$\mathcal{P}$		$\mathcal{U}$
	$\xleftarrow{N_{\mathsf{PT}},ts,\mathsf{rc}}$	$\mathsf{rc} = \sigma_{\mathsf{sk}_{\mathcal{U}}}(N_{\mathsf{PT}}, ts)$
$w \in_{\mathcal{R}} \mathbb{Z}_q,\ a = g^w,\ b = (\mathsf{C}g')^w$	$\xrightarrow{a,b}$	$s \in_{\mathcal{R}} \mathbb{Z}_q^*,\ A = (\mathsf{C}g')^s,\ z' = z^s$ $x, x_0, x_1, \ldots, x_n, x',$ $x'_0, x'_1, \ldots, x'_n, u, v \in_{\mathcal{R}} \mathbb{Z}_q$ $B = A^x g_0^{x_0} \ldots g_n^{x_n}$ $B' = A^{x'} g_0^{x'_0} \ldots g_n^{x'_n}$ $a' = a^u g^v,\ b' = b^{su} A^v$
	$\xleftarrow{c}$	$c' = \mathcal{H}(A, B, B', z', a', b')$ $c = c'/u$
$t = c \times \mathsf{sk}_{\mathcal{P}} + w$	$\xrightarrow{t}$	$g^t \stackrel{?}{=} \mathsf{pk}_{\mathcal{P}}^c a,\ (\mathsf{C}g')^t \stackrel{?}{=} z^c b$ $t' = t \times u + v$
Choose $\mathtt{SN} \in_{\mathcal{R}} \mathbb{Z}_q$ $\mathtt{SN}_{\mathsf{RT}} = g^{\mathtt{SN} \times \mathsf{sk}_{\mathcal{P}_{\mathsf{BLS}}}},\ T = g^{\mathtt{SN}}$	$\xrightarrow{\mathtt{SN}_{\mathsf{RT}},T}$	$e(\mathtt{SN}_{\mathsf{RT}}, h) \stackrel{?}{=} e(T, \mathsf{pk}_{\mathcal{P}_{\mathsf{BLS}}})$ $\mathsf{RT} = \mathtt{SN}_{\mathsf{RT}},\ R = 1,\ v = 0$

blinding factors B and B' and receives a signature $\mathsf{Sign}(A, B, B')$ on those. After execution of the protocol the user knows a representation of the payment token $A, B, B', \mathsf{Sign}(A, B,\ B') = A, B, B', z', a', b', t'$, which he stores together with the values $s, x,\ x_0, x_1, \ldots, x_n, x', x'_0, x'_1, \ldots, x'_n$ in his e-wallet.

Together with a bundle of payment tokens the user receives a refund token RT. The payment service provider chooses a serial number for the refund token $\mathtt{SN} \in_{\mathcal{R}} \mathbb{Z}_q$, computes $\mathtt{SN}_{\mathsf{RT}}$ together with a valid BLS signature on it and sends it to the user, who checks the signature and stores the refund token together with T, R and v. The payment service provider stores in his database that $\mathtt{SN}_{\mathsf{RT}}$ belongs to the user.

6.4.4 Start Charging

To start a charging process the user sends a payment token to a charging station, and proves possession of it using the blinding factor B. The electricity provider checks whether the received token is valid by computing $c' = \mathcal{H}(A, B, B', z', a', b')$ and verifying $\mathsf{Sign}(A, B, B')$ through

$$g^{t'} \stackrel{?}{=} \mathsf{pk}_{\mathcal{P}}^{c'} a' \quad \text{and} \quad A^{t'} \stackrel{?}{=} z'^{c'} b'. \tag{6.2}$$

He then saves the payment transcript $A, B, B', \mathsf{Sign}(A, B, B'),\ R, r_0, \ldots, r_n, d$. This process is presented in Table 6.4 and was before described in [52].

Table 6.4. P4R for e-mobility start charging protocol

$\mathcal{U}$		$\mathcal{E}$
	$\xrightarrow{A,B,B',\mathsf{Sign}(A,B,B')}$	$A \stackrel{?}{\neq} 1$
$r_0 = -d \times \mathsf{sk}_{\mathcal{U}} + x_0$		
$r_2 = -d \times L_2 + x_2$	$\xleftarrow{d}$	$d = \mathcal{H}(A, B, B', \mathsf{ID}_{\mathcal{E}}, ts)$
$\vdots$		
$r_n = -d \times L_n + x_n$		
$R = d/s + x$	$\xrightarrow{(R,r_0,r_2,\ldots,r_n)(L_1,x_1)}$	$r_1 = -d \times L_1 + x_1$
		$g_0^{r_0} g_1^{r_1} \ldots g_n^{r_n} g'^{-d} \stackrel{?}{=} A^{-R} B$
		Verify $\mathsf{Sign}(A, B,\ B')$

6.4.5 Stop Charging

To again authenticate with the same charging station the user sends the same payment token, used to start the charging process, to the charging station and proves possession of it this time using blinding factor B'. The electricity provider checks whether the received token is the same as was presented during the start charging process, and whether it is a valid token belonging to this user.

The user then presents his randomized refund token to the electricity provider, who computes the refund value as the value of the payment token minus the value of the energy that a user received $v_{\mathsf{RF}} = v_{\mathsf{PT}} - v_{\mathsf{EE}}$. The electricity provider signs $\mathsf{RT}', A, v_{\mathsf{RF}}$ and a time stamp and sends the payment transcript as well as the signature and the blinded refund token to the payment service provider, who checks the validity of the payment token and, if the signature verifies, adds v_{RF} to the refund token and credits $v_{\mathsf{PT}} - v_{\mathsf{RF}}$ to the electricity provider's bank account. The payment service provider

further sends a receipt to the electricity provider stating that $v_{\mathsf{PT}} - v_{\mathsf{RF}}$ was added to his bank account (cf. Table 6.5).

The payment service provider saves the payment transcript $A, B, B', \mathsf{Sign}(A, B, B')$, $R, r_0, \ldots, r_n, ts$ and the receipt he received from the electricity provider. Due to giving out this receipt the electricity provider cannot deny that he asked the payment service provider to add v_{RF} to the refund token and credit $v_{\mathsf{PT}} - v_{\mathsf{RF}}$ to the electricity provider's bank account.

The user checks whether the refund was added correctly. If the pairing equations hold, and if RT''_i was a valid refund token, then RT''_{i+1} is also a valid refund token additionally encoding v_{RF_i}.

6.4.6 Redeem Refund

To redeem his collected refund a user again randomizes his refund token and presents $\mathsf{RT}', \mathtt{SN}_{\mathsf{RT}}, v, R$ together with a receipt that she is trying to redeem her refund token to the payment service provider, who checks whether this token belongs to the user and whether the pairing equation in Table 6.6 is satisfied. If the check verifies the payment service provider credits the refund value v to the user's bank account, and marks the refund token as redeemed in his database.

6.4.7 Double-Spending Detection

During this process the payment service provider checks, whether he had already received a payment token with this particular serial number tag, *i.e.* whether a payment transcript with the same serial number tag A occurs in his database. If so he will check whether the same value d is part of the payment transcript, which indicates that the electricity provider deposited the same payment token twice. If not, the same payment token has been used twice by a user. The payment service provider can then reveal the cheating user's identity by computing

Table 6.5. P4R for e-mobility stop charging protocol

$\mathcal{U}$		$\mathcal{E}$
	$\xrightarrow{A,B,B',\mathsf{Sign}(A,B,B')}$	Check if the same A was presented during start charging.
$r'_0 = -d' \times \mathsf{sk}_{\mathcal{U}} + x'_0$		
$r'_2 = -d' \times L_2 + x'_2$	$\xleftarrow{d'}$	$d' = \mathcal{H}(A, B, B', \mathsf{ID}_{\mathcal{E}}, ts')$
$\vdots$		
$r'_n = -d' \times L_n + x'_n$		
$R' = d'/s + x'$	$\xrightarrow{R',r'_0,r'_2,\ldots,r'_n,L_1,x'_1}$	$r'_1 = -d' \times L_1 + x'_1$ $g_0^{r'_0} g_1^{r'_1} \ldots g_n^{r'_n} g'^{-d'} \stackrel{?}{=} A^{-R'} B'$ Verify $\mathsf{Sign}(A, B, B')$
$\mathsf{rnd}, \in_{\mathcal{R}} \mathbb{Z}_q^*,\ \mathsf{RT}' = \mathsf{RT}^{\mathsf{rnd}}$ $R = R \times \mathsf{rnd}$	$\xrightarrow{\mathsf{RT}'}$	

$\mathcal{E}$		$\mathcal{P}$
$\mathsf{rc}' = \sigma'_{\mathsf{sk}_{\mathcal{E}}}(\mathsf{RT}', A, v_{\mathsf{RF}}, \hat{ts})$	$\xrightarrow{\mathsf{RT}',A,B,B',\mathsf{Sign}(A,B,B')}$	
	$\xrightarrow{ts,R,r_0,\ldots r_n,d,v_{\mathsf{RF}},\hat{ts},\mathsf{rc}'}$	$g_0^{r_0} g_1^{r_1} \ldots g_n^{r_n} h^{-d} \stackrel{?}{=} A^{-R} B$ Verify $\mathsf{Sign}(A, B, B')$ and rc' Credit $v_{\mathsf{PT}} - v_{\mathsf{RF}}$ to $\mathcal{E}$s account $\mathsf{rc}'' = \sigma_{\mathsf{sk}_{\mathcal{P}}}(v_{\mathsf{PT}} - v_{\mathsf{RF}}, \mathsf{ID}_{\mathcal{E}}, ts)$ $v_{\mathsf{RF}} = \sum_{i=0}^{m} v_{\mathsf{RF}_i} 2^i$, $\mathsf{RT}''_0 = \mathsf{RT}'$ For $(i = 0$ to $m)$ If $(v_{\mathsf{RF}_i} \neq 0)$: $\mathsf{RT}''_{i+1} = (\mathsf{RT}''_i)^{\mathsf{sk}_{\mathcal{P}_{\mathsf{BLS}}}^{2^i}}$
	$\xleftarrow{\mathsf{RT}''_0,\ldots,\mathsf{RT}''_m,\mathsf{rc}'',ts}$	Else: $\mathsf{RT}''_{i+1} = \mathsf{RT}''_i$

$\mathcal{U}$		$\mathcal{E}$
	$\xleftarrow{\mathsf{RT}''_1,\ldots,\mathsf{RT}''_{m+1},v_{\mathsf{RF}}}$	
$v_{\mathsf{RF}} = \sum_{i=0}^{m} v_{\mathsf{RF}_i} 2^i$ For $i = 0$ to m If $(v_{\mathsf{RF}_i} \neq 0)$: $e(\mathsf{RT}''_{i+1}, h) \stackrel{?}{=} e(\mathsf{RT}''_i, \mathsf{RC}_i)$ $\mathsf{RT} = \mathsf{RT}''_m,\ v = v + v_{\mathsf{RF}}$		

$$\mathsf{C} = g_0^{(r_0 - \tilde{r}_0)/(R - \tilde{R})} \ldots g_n^{(r_n - \tilde{r}_n)/(R - \tilde{R})}.$$

Table 6.6. P4R for e-mobility redeeming refund protocol.

$\mathcal{P}$		$\mathcal{U}$
		$\mathsf{rnd}, w \in_{\mathcal{R}} \mathbb{Z}_q$ $\mathsf{RT}' = \mathsf{RT}^{\mathsf{rnd}},\ R = R \times \mathsf{rnd}$
Verify and store rc'''	$\xleftarrow{\mathsf{RT}', \mathtt{SN}_{\mathsf{RT}}, v, R, ts, \mathsf{rc}'''}$	$\mathsf{rc}''' = \sigma_{\mathsf{sk}_{\mathcal{U}}}(\mathsf{RT}', \mathtt{SN}_{\mathsf{RT}}, v, ts)$
$v \overset{?}{<} q$ $e(\mathtt{SN}_{\mathsf{RT}}^{R}, h^{\mathsf{sk}_{\mathcal{P}_{\mathsf{BLS}}}^{v}}) \overset{?}{=} e(\mathsf{RT}', h)$	$\xrightarrow{\text{money}}$	

6.5 Security and Privacy Analysis

In this section, we provide the main (informal) arguments why our scheme satisfies the security objectives stated in Section 6.3. A formal security proof based on the proof for the basic P4R scheme given in [86] is left as future work. While out of the scope of this paper, it would also have to be ensured that the scheme's implementation does not invalidate the security objectives. This could be verified by an independent trusted party, or the code could be put in the public domain.

Objective I requires that (a) payment tokens are only generated, if a user proves his wish to receive them, (b) when depositing money, a user receives valid payment tokens, which only he can spend, and (c) a user is paid correct refunds, which only he can redeem. (a) is ensured as a withdrawal process is only initiated if a user sends a signature on the number of coins he would like to receive to the payment service provider, thus indirectly authenticating and proving his wish to receive a certain amount of payment tokens. (b) is satisfied, as a user only accepts the withdrawal of a payment token if he received a valid one, and only he can spend them as knowledge of his secret key is required during the spending procedure. (c) is achieved because a user checks during the (re-)charge e-wallet process, whether he received a valid refund token. Using the list of published refund check values from the payment service provider he checks during or after each stop charging process, whether a correct refund was added to his refund token. An electricity provider could display the price of electricity before a charging process is started and a user's car could measure the

received energy, such that the user can compute the refund he should receive himself. A user thus ensures that he has a valid refund token at all times. If he would further store all intermediate RT' that he received from the payment service provider until he redeems RT, he could prove to a judge that he is in possession of a valid RT encoding v by showing that based on respective RT' the subsequently generated RT'' was formed correctly. Secondly, only a user himself is able to redeem his (correctly received) RT as he has to authenticate before the redeem refund process and the serial number of his RT was bound to his identity. The payment service provider cannot claim that a user has already redeemed a refund token, as he would need to present a receipt from the user to prove this statement. Note that, as we only consider passive adversaries, an adversary may not exchange a user's refund token for his own when a user is about to collect his refund from an electricity provider (cf. Table 6.5).

Objective II requires that a user can (d) anonymously and unlinkably (w.r.t. the withdrawal of the payment tokens or other charging transactions) present payment tokens, (e) authenticate at a charging station again, after leaving during a charging process, and (f) collect and redeem refunds in a privacy-preserving manner. (d) is ensured by the blindness property of Brands' blind signature scheme. Thus, when giving out a payment token the payment service provider does not know whom it had given it to. Due to allowing the encoding of attributes into payment tokens, achieving this objective becomes more difficult. It needs to be ensured that the collected data does not allow conclusions about a user's identity. (e) is achieved due to the possibility of showing a payment token twice, without revealing the identity of the user, which allows a privacy-preserving double-authentication. (f) is ensured as a refund token is blinded each time it is presented to the payment service provider or an electricity provider. This makes sure that the payment service provider and the electricity providers cannot track an RT due to its appearance.

Objective III requires that (g) only a payment service provider can generate valid payment tokens, (h) a user cannot not deny her wish to withdraw payment tokens after a withdrawal process, (i) a payment token can be used to pay for only one charging process, (j) an electricity provider cannot claim to be credited more money than what she claimed to be the value of charge given to a user during a charging process, and (k) a user can only redeem refunds that he correctly collected over a period of time. (g) is ensured due to the security/unforgeability of blind signatures. (h) is ensured as a user provides a signature on the number of payment tokens he would like to receive during the (re-)charge e-wallet process. The payment service provider can use the signature to prove to a judge that he acted correctly. (i) is achieved by the double-spending detection mechanism. If the user uses the same payment token for another charging process, her identity can be revealed from the payment transcripts and she can be fined. (j) is ensured as an electricity provider cannot deposit payment tokens multiple times without this malicious behavior being revealed and further she cannot claim that a wrong value had been credited to her account, as she provided a receipt over this value. (k) is achieved as a user can only redeem a refund token, of which the serial number is stored in the payment service provider's refund token database. Further refunds can only be added to the refund token by the payment service provider. And third a user gives a receipt to a payment service provider proving that she has redeemed a particular refund token. She can thus not redeem a refund token multiple times. Note that the payment service provider could exclude a user from the payment service, if he would not provide this receipt.

Objective IV requires that an electricity provider only accepts valid payment tokens and that he is credited the value of electric charge he gave out. To ensure this an electricity provider checks the validity of a payment token when receiving it and receives a receipt from the payment service provider that $v_{\mathsf{PT}} - v_{\mathsf{RF}}$ will be credited

to her account. He can then prove to a judge that she should had been credited the amount.

6.6 Implementation on an NFC-Smartphone

This section arose from collaborative work with Olga Korobova, who was supervised during a student project by the author of this dissertation.

A popular technology for mobile payments is NFC due to its fast and simple communication set-up, which only requires that NFC-enabled devices are brought into close proximity with each other. With a market share of almost 80%, Android is the dominating smartphone operating system in use today[1]. We hence analyze the performance of the proposed payment system on this very popular platform. In particular we used a Google Nexus 5 smartphone, which has a Qualcomm Snapdragon S800 Quadcore processor with 2 GB of RAM running at 2.3 GHz. This phone supports Android 4.4 KitKat, which is the first Android version to support host-based card emulation. Host-based card emulation allows to emulate the functionality of a contactless smartcard on the main CPU of the device, relying on NFC for communication with a contactless reader. We developed a hybrid Java/C application for the smartphone using the Android Developer Tools.

As terminal we used a laptop featuring an Intel Core i5 CPU running at 1.4 GHz. The terminal represents a charging station of an electricity provider as well as the back-end of the payment service provider. This leaves out the communication between a charging station and the payment service provider's back-end via the Internet. We emphasize that our focus is on measuring the required execution time on the mobile payment device as well as the time for communicating protocol data between the payment device and a terminal via NFC, which has a quite limited bandwidth

[1]Smartphone OS Market Share, Q1 2015. `http://www.idc.com/prodserv/smartphone-os-market-share.jsp`

compared to a strong Internet connection. Terminals can be equipped with powerful hardware, dedicated to execute the required protocols, whereas the phone is powered with a battery and thus limited in computational power. In the implementation whose performance results we present in Section 6.7 the data that is communicated between an electricity provider and a payment service provider is only 806 bytes in length. An Internet connection should thus not represent a bottleneck in the system. The NFC communication was set up with an HID OMNIKEY 5321 v2 smartcard reader that was connected to the PC via USB. We developed a Java application for the terminal, using the Java Smart Card I/O API summarized in the `javax.smartcardio` package. This package allows an easy implementation of communication with a smartcard and thus the host-based emulated card on the NFC-smartphone.

We based the cryptographic functionality on the free portable C Pairing Based Cryptography (PBC) library[2] version 0.5.13. We implemented all protocol functions in C and then called those from our Java applications using the Java Native Interface (JNI). We use asymmetric pairings as none of our building blocks requires $G_1 = G_2$. Note that in our scheme most protocol steps only require computation in G_1. Thus asymmetric pairings present a better choice, as pairing parameters can be chosen in a way that computation in G_1 relies on a very limited bit length and thus allows efficient computations in this group [37]. We rely on the Type F pairing parameters given in the PBC library. Those are parameters for pairing-friendly elliptic curves of prime order proposed by Barreto *et al.* in [8] achieving an embedding degree of 12. As mentioned in the PBC manual only 160 bits are required for elements in G_1, whereas 320 bits are required for elements in G_2.

[2] `http://crypto.stanford.edu/pbc/`

6.7 Performance of the Proposed Payment Scheme on an NFC-Smartphone

We implemented the scheme for a payment token value of \$10, thus allowing refund values of up to \$10. We further set the refund unit to be 1 ¢ allowing refund values of 0 ¢ up to 1000 ¢. We ran the scheme 200 times choosing the system parameters, *i.e.* the generators of G_1 and G_2 as well as the secret keys randomly each time and collected timings varying the refund values in a deterministic way. For each checked refund token value we ran the protocol 10 times. Note that the protocol timings depend on the hamming weight of v_{RF}. This is because the more v_{RF_i} are one, the more RT''_{i+1} have to be computed on the terminal side and have to be sent to the user and even more importantly the more pairing equations have to be checked by the user to verify whether the correct refund value has been encoded in her refund token.

We implemented the scheme for the case of encoding two attributes into a payment token, and revealing only one attribute. Revealing another attribute would increase the communicated data that a user has to send to the electricity provider by 24 bytes. In our implementation 468 bytes are communicated via NFC during spending, which requires 33.5 ms (cf. Table 6.7). We thus anticipate that the increase in communication time would be low, if another attribute was revealed.

We first verified by practical analysis that the location of the one bit in v_{RF} does not influence the execution time, by measuring execution times for the refund values $v_{\mathsf{RF}} = 2^i$ for $0 < i < 10$. We then collected timings in which we increased the hamming weight of the refund value during each iteration, *i.e.* starting from $v_{\mathsf{RF}} = 0$ we measured timings for each $v_{\mathsf{RF}} = 2^i + 1$ in which $0 < i < 10$. Table 6.7 presents the average results for running the scheme, where the hamming weight of the refund value was incremented in every tenth iteration, thus presenting the average execution time. We give numbers for the overall computation on the phone (column Smartphone),

Table 6.7. Execution times of the various protocols of our payment scheme, based on a Google Nexus 5 NFC-smartphone, using host-based card emulation. The timings are averaged for refund values of all possible hamming weights.

	Smartphone	**Terminal**	**Communication**	**Total**
User Registration	273.6 ms	5.1 ms	716.0 ms	994.2 ms
(Re-) Charge E-wallet	412.3 ms	25.8 ms	139.6 ms	577.7 ms
Start Charging	0 ms	12.6 ms	33.5 ms	46, 1 ms
Stop Charging	1614.7 ms	42.9 ms	176 ms	1833, 6 ms
Redeem Refund	11.9 ms	134.3 ms	73.3 ms	219, 5 ms

the overall communication time (column Communication), the overall computation on the terminal (column Terminal) and the overall execution time of each protocol step (column Total). These results are illustrated in Figure 6.2.

The results demonstrate that all transactions can be executed within 2 seconds. We emphasize that the time critical transactions are the start charging and stop charging process, as those are executed regularly at a charging station at the beginning and end of a charging process. The registration on the other hand is executed only once per user, whenever new system parameters are generated, which could be done at a regular basis once a year. (Re-)charging the electronic wallet could be executed by the payment application running in the background on the phone. We thus focus our analysis on the start charging and stop charging process. While the start charging process clearly presents an acceptable execution performance, the execution of the stop charging process is quite long, with an average execution time of 1.8 seconds. We extended our analysis for this part and present solutions to this problem.

We present timings for the *Stop Charging* protocol, incrementing the hamming weight of the refund value in Figure 6.3. Note that without computing the pairing equations to check, whether the correct refund has been encoded in RT' the *End Charging* protocol executes in 223.1 ms on average. A solution would thus be that a user checks whether he received correct refunds at a later point in time. The payment service provider could instead send a receipt to the user stating which value was added

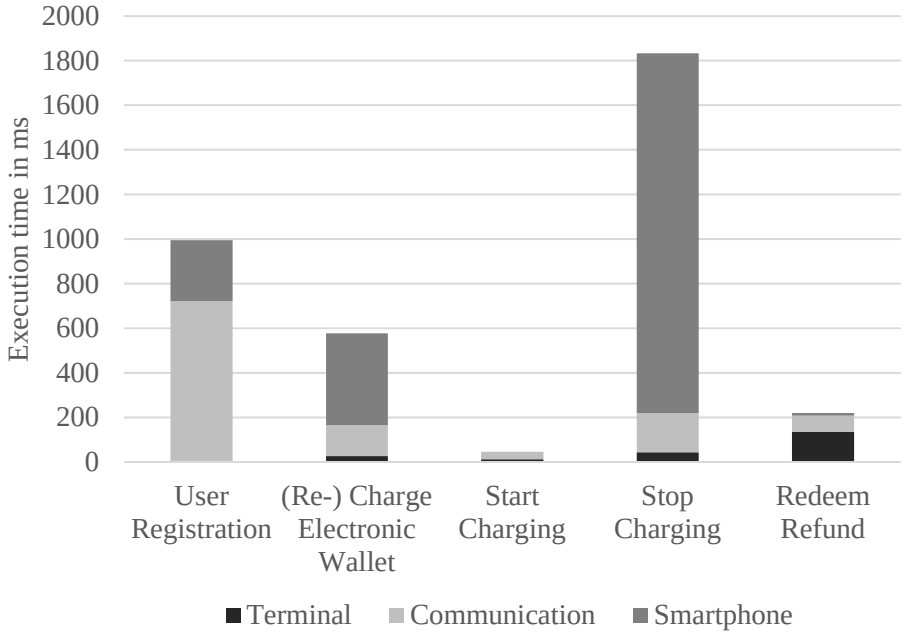


Figure 6.2. Illustration of Table 6.7, presenting the execution times of the various protocols of our payment scheme, based on a Google Nexus 5 NFC-smartphone, using host-based card emulation. The timings are averaged for refund values of all possible hamming weights.

to her refund token. Generating and validating a receipt is quite efficient, as it only requires computations in G_1 in our scheme. If the user finds out that the payment service provider has added a wrong value to his refund token or has generated an invalid refund token, the user could use this receipt in front of a judge.

This solution leads to further system variations. In a real-world scenario security-critical operations of the payment application would have to be executed on secure hardware as for example the secure element of a smartphone rather than its main CPU. This hardware can be assumed to be equally powerful to that of a smartcard. It is further highly desirable to allow different types of payment devices, as it cannot be expected that every user is in possession of a powerful NFC smartphone. In addition to using one's smartphone a device such as a smartcard should be offered to users at low cost. We would like to emphasize that all user side's computation, apart from checking whether a correct refund was added to the refund token, requires computations in G_1 only, which in our case is a 160-bit elliptic curve. Those computations could be executed on a smartcard-like device (cf. Chapter 5). Note that the pairing equation checks, which are executed in order to verify that a refund was encoded into the refund token correctly, only involve computations using public keys. Those could thus be executed on the smartphone's main processor or a terminal such as a user's personal computer, which a smartcard could be connected to.

We note that the communication takes quite a significant amount of time during the execution of the respective protocols. This partially originates from the fact that we had to split up communicated data into multiple APDUs, as one APDU can carry up to a maximum of 255 bytes of data only. This is because one byte in an APDU is dedicated to indicate the length of the sent data. Extended APDUs have been introduced in which this length is indicated with a value stored in two bytes, thus allowing a maximum of 65535 bytes of data to be sent at once. While this was not supported by our set up, it could noticeably reduce execution times.

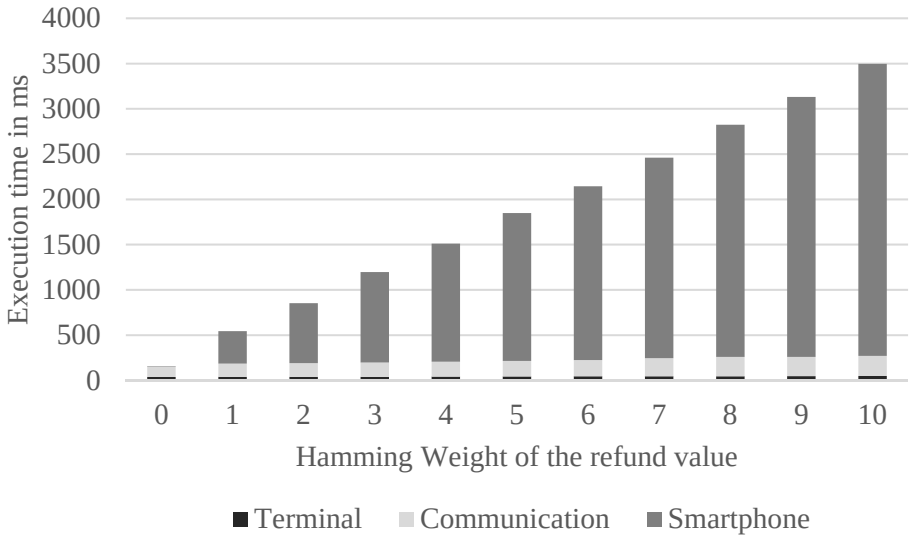


Figure 6.3. Execution times of the *End Charging* protocol for different refund values, incrementing the hamming weight of the refund value in each column.

6.8 Variations

We rely on a double-spending detection mechanism to punish fraudulent behavior, as this spreads the load of the back-end of the payment service provider. If fraud should be prevented one would instead rely on an online payment system. This would require a determination, whether a coin had been spent before, in real time, thus requiring real-time access to the payment token database of the payment service provider. At the same time this would reduce the complexity of the scheme, as no double-spending detection mechanism that reveals the identity of a cheating party would be required.

In our scheme an active adversary could interrupt the communication in the End Charging protocol and present his refund token instead of the user's refund token. This way the refund would be added to the adversary's refund token. An extension would be desirable in which the presented refund token in the End Charging protocol would be bound to the presented payment token, thus preventing this type of active attack.

It is desirable that a user can feed back electric energy to the grid, thus allowing to store electric energy in electric mobiles whenever a surplus is produced and feeding it back whenever there is a lack of electric energy. This can be achieved with our system, giving out payment tokens of zero value, which a user can use to authenticate to a charging station twice. The value of the energy that she fed back into the grid would then be credited to her refund token.

6.9 Discussion

We proposed how to adapt the lightweight privacy-preserving pre-payments with refunds scheme (P4R), which targets public transportation payment systems, to the e-mobility domain. Relying on the same payment scheme to be used for public transportation systems and in the e-mobility domain greatly increases user convenience. We allow the encoding of attributes into payment tokens and their selective disclosure. We further presented a full implementation of the proposed scheme on a Google Nexus 5 NFC-smartphone together with a thorough analysis of its performance. Apart from checking whether a refund was added correctly to a user's refund token, the time critical start charging and stop charging processes can be executed in 46 ms and 223 ms respectively. Including the refund check increases the execution time of the stop charging process to 1.8 seconds, which can be a limiting factor. We presented solutions to this, and various ideas, how the scheme could be modified or extended.

CHAPTER 7

CONCLUSION AND OUTLOOK

The ever-increasing processing power and hence the possibility to analyze large data sets increases the importance of privacy protection. It is thus of utter importance to design systems that by design protect the privacy of their users. This is particularly important in the domain of electronic payments. While electronic payments have many advantages for payers and payees, currently employed electronic payment systems usually reveal a payer's identity during a payment. This allows to generate fine grain patterns of users' behavior. In this dissertation we focused on a very concrete case, namely the analysis of privacy-preserving transportation payment systems. The unique characteristics of payments in the transportation domain are the high number of payments that have to be executed every day and the facts that the paid value is usually low and that payments have to be executed quickly in order to avoid congestion at the entrance points of a transportation system.

We analyzed the use of e-cash for the transportation setting and presented advantages and disadvantages of implementations of various payment system configurations resulting in designs that maintain or increase high standards of user convenience, while at the same time ensuring security and allowing the protection of users' privacy. We targeted various aspects of the implementation of the considered e-cash schemes, ranging from high-level implementations on NFC-enabled smartphones to low-level implementations for extremely constrained passively powered RFID-tags.

By practical evaluation, we demonstrated that e-cash presents a promising solution for privacy-preserving payments in public transportation systems, if the number

of coins that have to be spent per payment can be kept low. While the computation required for the time-critical spending of a single coin can be executed in 13 ms even on the passively powered computational RFID tag Moo the less time-critical withdrawal requires several seconds, which however could be sped up relying on hardware accelerators. We further showed that the limitation of having to keep the number of spent coins low can be alleviated by using the privacy-preserving pre-payments with refunds scheme (P4R). P4R adds a flexible privacy-preserving refund system to e-cash. Together with the fact that in P4R a fare price can be determined at an exit point of the transportation system, this allows for a very flexible and dynamic pricing system where prices can be adapted to customer attributes (as for example their age) and transport conditions. We further analyzed how to use specific features of e-cash protocols to target the unique requirements of transportation payment systems, one of them being the encoding of user attributes into coins, which we showed can be done at only a moderate performance loss. Using off-the-shelf smartcards the payment of an electronic coin, which encodes two user attributes, which are not revealed, can be executed in under 800 ms. Moreover, we presented a privacy-preserving payment scheme targeting the unique requirements of e-mobility, where it is desired that a user pays for the received charge at the end of a charging process, when the amount of charge can be determined, and at the same time it has to be ensured that a user cannot finalize the charging process avoiding to pay for received charge. Here the charging process can be started within 46 ms, while finalizing the charging process requires 1.8 s.

A general limitation of e-cash based payment systems, which is a limitation of all electronic payment schemes in which a user has to load an electronic wallet, is the fact that a user always gives up a certain level of privacy by loading his payment device. By analyzing the overall amount of money a user spends on fares, the transportation authority knows, whether a user tends to take shorter or longer trips, though it cannot

differentiate between those users that take only few long trips versus those that take many short trips.

While this thesis focused on the implementation aspects and practicability aspects of e-cash, several interesting new e-cash protocols have been proposed recently. One of them is transferable e-cash [6], which does not require that a shop deposits a received coin to his bank account right away, but allows to move coins back and forth between users and shops multiple times, reaching a user experience similar to physical cash. We propose the practical analysis of advanced e-cash schemes as a promising future research direction. As such, many advanced e-cash protocols rely on cryptographic pairings, while the implementation of lightweight pairings has not yet been investigated thoroughly. As a further future research direction for the implementation of e-cash schemes we hence propose the analysis of light-weight pairing implementations in hard- and software.

In conclusion, one can say that this thesis demonstrates that e-cash is not only interesting from a theoretical perspective. Efficient and practical realization of e-cash schemes are feasible and present a promising alternative for classic electronic payments, especially in the transportation domain.

BIBLIOGRAPHY

[1] Near Field Communication Forum. http://www.nfc-forum.org/, 2008.

[2] M. Abe. A secure three-move blind signature scheme for polynomially many signatures. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 136–151. Springer Berlin Heidelberg, 2001.

[3] R. Abelson and M. Goldstein. Millions of anthem customers targeted in cyberattack. `http://www.nytimes.com/2015/02/05/business/hackers-breached-data-of-millions-insurer-says.html?_r=1`, 2015.

[4] O. Acıiçmez, B. Brumley, and P. Grabher. New results on instruction cache attacks. In S. Mangard and F.-X. Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 110–124. Springer Berlin Heidelberg, 2010.

[5] N. Asokan, P. A. Janson, M. Steiner, and M. Waidner. The state of the art in electronic payment systems. *IEEE Computer*, 30(9):28–35, 1997.

[6] F. Baldimtsi, M. Chase, G. Fuchsbauer, and M. Kohlweiss. Anonymous transferable e-cash. In J. Katz, editor, *Public-Key Cryptography – PKC 2015*, volume 9020 of *Lecture Notes in Computer Science*, pages 101–124. Springer Berlin Heidelberg, 2015.

[7] F. Baldimtsi and A. Lysyanskaya. Anonymous credentials light. In *Proceedings of the 2013 ACM SIGSAC conference on Computer communications security*, CCS '13, pages 1087–1098, New York, NY, USA, 2013. ACM.

[8] P. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer Berlin Heidelberg, 2006.

[9] L. Batina, J.-H. Hoepman, B. Jacobs, W. Mostowski, and P. Vullers. Developing efficient blinded attribute certificates on smart cards via pairings. In D. Gollmann, J.-L. Lanet, and J. Iguchi-Cartigny, editors, *Smart Card Research and Advanced Application*, volume 6035 of *Lecture Notes in Computer Science*, pages 209–222. Springer Berlin Heidelberg, 2010.

[10] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. Compact e-cash and simulatable VRFs revisited. In H. Shacham and B. Waters, editors, *Pairing-Based Cryptography – Pairing 2009*, volume 5671 of *Lecture Notes in Computer Science*, pages 114–131. Springer Berlin Heidelberg, 2009.

[11] D. Bernstein. Batch binary edwards. In S. Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 317–336. Springer Berlin Heidelberg, 2009.

[12] D. Bernstein and P. Schwabe. NEON crypto. In E. Prouff and P. Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 320–339. Springer Berlin Heidelberg, 2012.

[13] D. J. Bernstein. Curve25519: new Diffie-Hellman speed records. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *Public Key Cryptography – PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer-Verlag Berlin Heidelberg, 2006. `http://cr.yp.to/papers.html#curve25519`.

[14] D. J. Bernstein. Cryptography in NaCl. `http://cr.yp.to/highspeed/naclcrypto-20090310.pdf`, 2009.

[15] D. J. Bernstein, B. van Gastel, W. Janssen, T. Lange, P. Schwabe, and S. Smetsers. TweetNaCl: A crypto library in 100 tweets, to appear. Document ID: c74b5bbf605ba02ad8d9e49f04aca9a2, `http://cryptojedi.org/papers/#tweetnacl`.

[16] P. Bichsel, J. Camenisch, T. Groß, and V. Shoup. Anonymous credentials on a standard java card. In *Proceedings of the 16th ACM Conference on Computer and Communications Security – ACM CCS 2009*, CCS '09, pages 600–610, New York, NY, USA, 2009. ACM.

[17] E.-O. Blass, A. Kurmus, R. Molva, and T. Strufe. PSP: Private and secure payment with rfid. In *Proceedings of the 8th ACM Workshop on Privacy in the Electronic Society*, WPES '09, pages 51–60, New York, NY, USA, 2009. ACM.

[18] A. J. Blumberg. On locational privacy, and how to avoid losing it forever. 2009.

[19] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[20] 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. *IACR Cryptology ePrint Archive*, 2009:389, 2009.

[21] S. Brands. Untraceable off-line cash in wallets with observers (extended abstract). In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '93, pages 302–318, London, UK, UK, 1994. Springer-Verlag.

[22] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321. Springer Berlin Heidelberg, 2005.

[23] A. Chan, Y. Frankel, and Y. Tsiounis. Easy come — easy go divisible cash. In K. Nyberg, editor, *Advances in Cryptology — EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 561–575. Springer Berlin Heidelberg, 1998.

[24] D. Chaum. Blind signatures for untraceable payments. In D. Chaum, R. Rivest, and A. Sherman, editors, *Advances in Cryptology*, pages 199–203. Springer US, 1983.

[25] D. Chaum. Achieving electronic privacy. *Scientific American*, pages 96–101, August 1992.

[26] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In S. Goldwasser, editor, *Advances in Cryptology — CRYPTO' 88*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer New York, 1990.

[27] E. Clemente-Cuervo, F. Rodríguez-Henríquez, D. O. Arroyo, and L. Ertaul. A PDA implementation of an off-line e-cash protocol. In *Proceedings of the 2007 International Conference on Security & Management, SAM 2007, Las Vegas, Nevada, USA, June 25-28, 2007*, pages 452–458, 2007.

[28] B. Cohen. AES-hash. `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/aes-hash/aeshash.pdf`, 2001.

[29] N. Costigan and P. Schwabe. Fast elliptic-curve cryptography on the cell broadband engine. In B. Preneel, editor, *Progress in Cryptology – AFRICACRYPT 2009*, volume 5580 of *Lecture Notes in Computer Science*, pages 368–385. Springer Berlin Heidelberg, 2009.

[30] J. Daemen and V. Rijmen. The rijndael block cipher. `http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf`, 1999.

[31] Department of Economic and Social Affairs - United Nations New York. *World Urbanization Prospects - The 2011 Revision*, 2012.

[32] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, Sept. 2006.

[33] M. Düll, B. Haase, G. Hinterwälder, M. Hutter, C. Paar, A. H. Sánchez, and P. Schwabe. High-speed Curve25519 on 8-bit, 16-bit and 32-bit microcontrollers, 2015. Submitted to Designs Codes and Cryptography Journal.

[34] S. Döbelt, B. Kämpfe, and J. F. Krems. Smart grid, smart charging, smart privacy? an empirical investigation of consumers' willingness to provide smart charging information. 2014.

[35] T. for London. Oyster. `http://www.tfl.gov.uk/fares-and-payments/oyster/what-is-oyster` [Accessed: 2015-03-08], 2015.

[36] T. Frosch, S. Schäge, M. Goll, and T. Holz. Improving location privacy for the electric vehicle masses. 2013. [pdf].

[37] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113 – 3121, 2008. Applications of Algebra to Cryptography.

[38] F. Garcia, G. de Koning Gans, R. Muijrers, P. van Rossum, R. Verdult, R. Schreur, and B. Jacobs. Dismantling MIFARE classic. In S. Jajodia and J. Lopez, editors, *Computer Security - ESORICS 2008*, volume 5283 of *Lecture Notes in Computer Science*, pages 97–114. Springer Berlin Heidelberg, 2008.

[39] R. Goundar, M. Joye, and A. Miyaji. Co-z addition formulæ and binary ladders on elliptic curves. In S. Mangard and F.-X. Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 65–79. Springer Berlin Heidelberg, 2010.

[40] C. Gouvêa and J. López. Software implementation of pairing-based cryptography on sensor networks using the MSP430 microcontroller. In B. Roy and N. Sendrier, editors, *Progress in Cryptology - INDOCRYPT 2009*, volume 5922 of *Lecture Notes in Computer Science*, pages 248–262. Springer Berlin Heidelberg, 2009.

[41] C. Gouvêa, L. B. Oliveira, and J. López. Efficient software implementation of public-key cryptography on sensor networks using the MSP430X microcontroller. *Journal of Cryptographic Engineering*, 2(1):19–29, 2012.

[42] C. P. L. Gouvêa. Personal communication, Oct. 2014.

[43] G. Greenwald. Why privacy matters. TED Talk `http://www.ted.com/talks/glenn_greenwald_why_privacy_matters?language=de#t-258248`, 2014.

[44] J. Guajardo, R. Blümel, U. Krieger, and C. Paar. Efficient implementation of elliptic curve cryptosystems on the ti MSP430x33x family of microcontrollers. In K. Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 365–382. Springer Berlin Heidelberg, 2001.

[45] N. Gura, A. Patel, A. Wander, H. Eberle, and S. Shantz. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 119–132. Springer Berlin Heidelberg, 2004.

[46] D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[47] E. Haselsteiner and K. Breitfuß. Security in Near Field Communication (NFC) - Strengths and Weaknesses. http://events.iaik.tugraz.at/RFIDSec06/Program/papers/0022006.

[48] T. S. Heydt-Benjamin, H.-J. Chae, B. Defend, and K. Fu. Privacy for public transportation. In *Proceedings of the 6th International Conference on Privacy Enhancing Technologies*, PET'06, pages 1–19, Berlin, Heidelberg, 2006. Springer-Verlag.

[49] G. Hinterwälder, A. Moradi, M. Hutter, P. Schwabe, and C. Paar. Full-size high-security ECC implementation on MSP430 microcontrollers. In *Third International Conference on Cryptology and Information Security in Latin America - LATINCRYPT 2014, Florianopolis, Brazil, September 17-19, 2014. Proceedings.* Springer, 2014. in press.

[50] G. Hinterwälder, C. Paar, and W. Burleson. Privacy preserving payments on computational RFID devices with application in intelligent transportation systems. In J.-H. Hoepman and I. Verbauwhede, editors, *Radio Frequency Identification. Security and Privacy Issues*, volume 7739 of *Lecture Notes in Computer Science*, pages 109–122. Springer Berlin Heidelberg, 2013.

[51] G. Hinterwälder, F. Riek, and C. Paar. Efficient full-size e-cash on MULTOS smartcards, 2015. Submitted to 11-th Workshop on RFID Security – RFIDsec 2015.

[52] G. Hinterwälder, C. T. Zenger, F. Baldimtsi, A. Lysyanskaya, C. Paar, and W. P. Burleson. Efficient e-cash in practice: NFC-based payments for public transportation systems. In E. De Cristofaro and M. Wright, editors, *Privacy Enhancing Technologies*, volume 7981 of *Lecture Notes in Computer Science*, pages 40–59. Springer Berlin Heidelberg, 2013.

[53] M. Hutter and P. Schwabe. NaCl on 8-bit AVR microcontrollers. In A. Youssef, A. Nitaj, and A. Hassanien, editors, *Progress in Cryptology – AFRICACRYPT 2013*, volume 7918 of *Lecture Notes in Computer Science*, pages 156–172. Springer Berlin Heidelberg, 2013.

[54] M. Hutter and P. Schwabe. Multiprecision multiplication on AVR revisited, 2014. `https://cryptojedi.org/papers/avrmul-20140715.pdf`.

[55] M. Hutter and E. Wenger. Fast multi-precision multiplication for public-key cryptography on embedded microprocessors. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 459–474. Springer Berlin Heidelberg, 2011.

[56] T. I. Incorporated. MSP430x2xx family user's guide. `http://www.ti.com/lit/ug/slau144j/slau144j.pdf`, 2004.

[57] T. I. Incorporated. Enabling secure portable medical devices with TI's MSP430 MCU and wireless technologies. `http://www.ti.com/lit/wp/slay027/slay027.pdf`, 2012.

[58] T. I. Incorporated. MSP430FR58xx, MSP430FR59xx, MSP430FR68xx, and MSP430FR69xx family user's guide. `http://www.ti.com/lit/ug/slau367e/slau367e.pdf`, 2012.

[59] T. I. Incorporated. MSP43F261x, MSP43F241x mixed signal microcontroller. 2012.

[60] T. I. Incorporated. MSP-EXP430FR5969 LaunchPad Development Kit user's guide. `http://www.ti.com/lit/ug/slau535a/slau535a.pdf`, 2014.

[61] W. Janssen. Curve25519 in 18 tweets. Bachelor's thesis, Radboud University Nijmegen, 2014. `http://www.cs.ru.nl/bachelorscripties/2014/Wesley_Janssen___4037332___Curve25519_in_18_tweets.pdf`.

[62] A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7:595–596, 1963. Translated from Doklady Akademii Nauk SSSR, Vol. 145, No. 2, pp. 293–294, July 1962.

[63] N. Lawson. Highway to hell: Hacking toll systems. `http://www.root.org/talks/BH2008_HackingTollSystems.pdf`, 2008.

[64] J. K. Liu, M. H. Au, W. Susilo, and J. Zhou. Enhancing location privacy for electric vehicles (at the right time). In S. Foresti, M. Yung, and F. Martinelli, editors, *Computer Security – ESORICS 2012*, volume 7459 of *Lecture Notes in Computer Science*, pages 397–414. Springer Berlin Heidelberg, 2012.

[65] D. M. Lochter. ECC brainpool standard curves and curve generation v. 1.0. `http://www.ecc-brainpool.org/download/Domain-parameters.pdf`, 2005.

[66] MAOSCO Limited. MULTOS developer's guide. `http://www.multos.com/uploads/MDG.pdf`, 2014.

[67] M. B. T. A. (MBTA). MBTA ScoreCard 2013 March [Feb'13 Data]. http://www.mbta.com/uploadedfiles/About_the_T/Score_Card/ScoreCard2013.

[68] N. Meloni. New point addition formulae for ECC applications. In C. Carlet and B. Sunar, editors, *Arithmetic of Finite Fields*, volume 4547 of *Lecture Notes in Computer Science*, pages 189–201. Springer Berlin Heidelberg, 2007.

[69] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[70] S. Michael McFarland. Why we care about privacy. `http://www.scu.edu/ethics/practicing/focusareas/technology/internet/privacy/why-care-about-privacy.html`, 2012.

[71] P. L. Montgomery. Speeding the pollard and elliptic curve methods of factorization. In *Mathematics of Computation*, pages 243–264, 1987.

[72] W. Mostowski and P. Vullers. Efficient u-prove implementation for anonymous credentials on smart cards. In M. Rajarajan, F. Piper, H. Wang, and G. Kesidis, editors, *Security and Privacy in Communication Networks*, volume 96 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 243–260. Springer Berlin Heidelberg, 2012.

[73] C. Mulliner. Vulnerability analysis and attacks on NFC-enabled mobile phones. In *Availability, Reliability and Security, 2009. ARES '09. International Conference on*, pages 695–700, March 2009.

[74] Multos international Pte Ltd. Company profile & an introduction to MULTOS technology. `http://www.multosinternational.com/media/4992/multos_international.pdf`.

[75] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`.

[76] C. Paar and J. Pelzl. *Understanding Cryptography - A Textbook for Students and Practitioners*. Springer, 2010.

[77] C. Paar, A. Rupp, K. Schramm, A. Weimerskirch, and M. Wolf. Implementing data security and privacy in next-generation electric vehicle systems. 2010. [pdf].

[78] C. Pendl, M. Pelnar, and M. Hutter. Elliptic curve cryptography on the WISP UHF RFID tag. In A. Juels and C. Paar, editors, *RFID. Security and Privacy*, volume 7055 of *Lecture Notes in Computer Science*, pages 32–47. Springer Berlin Heidelberg, 2012.

[79] N. Perlroth. Target struck in the cat-and-mouse game of credit theft. `http://www.nytimes.com/2013/12/20/technology/target-stolen-shopper-data.html?pagewanted=all`, 2013.

[80] W. Rankl and W. Effing. *Smart Cards in Payment Systems*, pages 747–788. John Wiley & Sons, Ltd, 2010.

[81] W. Rankl and W. Effing. *Smart Cards in Transportation Systems*, pages 869–891. John Wiley & Sons, Ltd, 2010.

[82] C. Research. SEC 2: Recommended elliptic curve domain parameters, 2000. `http://www.secg.org/download/aid-386/sec2-final.pdf`.

[83] S. K. Ribeiro, S. Kobayashi, M. Beuthe, J. Gasca, D. Greene, D. S. Lee, Y. Muromachi, P. J. Newton, S. Plotkin, D. Sperling, R. Wit, and P. J. Zhou. Transport and its infrastructure. *Climate Change 2007: Mitigation. Contribution of Working Group III to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*, 2007.

[84] M. Rivain. Fast and regular algorithms for scalar multiplication over elliptic curves. Cryptology ePrint Archive, Report 2011/338, 2011. `https://eprint.iacr.org/2011/338.pdf`.

[85] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. Special publication 800 - 22 revision 1a – a statistical test suite for random and pseudorandom number generators for cryptographic applications. `http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22rev1a.pdf`, 2010.

[86] A. Rupp, F. Baldimtsi, G. Hinterwälder, and C. Paar. Cryptographic theory meets practice: Efficient and privacy-preserving payments for public transport. *ACM Trans. Inf. Syst. Secur.*, 17(3):10:1–10:31, Mar. 2015.

[87] A. Rupp, G. Hinterwälder, F. Baldimtsi, and C. Paar. P4R: Privacy-preserving pre-payments with refunds for transportation systems. In A.-R. Sadeghi, editor, *Financial Cryptography and Data Security*, volume 7859 of *Lecture Notes in Computer Science*, pages 205–212. Springer Berlin Heidelberg, 2013.

[88] R. Ryan, Z. Anderson, and A. Chiesa. The Anatomy of a Subway Hack: Breaking Crypto RFID's and Magstripes of Ticketing Systems. Presentation at DEFCON '08, 2008.

[89] A. Sadeghi, I. Visconti, and C. Wachsmann. User privacy in transport systems based on RFID e-tickets. In *Proceedings of the 1st International Workshop on Privacy in Location-Based Applications, Malaga, Spain, October 9, 2008*, 2008.

[90] A.-R. Sadeghi and M. Schneider. Electronic payment systems. In E. Becker, W. Buhse, D. Günnewig, and N. Rump, editors, *Digital Rights Management*, volume 2770 of *Lecture Notes in Computer Science*, pages 113–137. Springer Berlin Heidelberg, 2003.

[91] P. Sasdrich and T. Güneysu. Efficient elliptic-curve cryptography using Curve25519 on reconfigurable devices. In D. Goehringer, M. Santambrogio, J. Cardoso, and K. Bertels, editors, *Reconfigurable Computing: Architectures, Tools, and Applications*, volume 8405 of *Lecture Notes in Computer Science*, pages 25–36. Springer International Publishing, 2014.

[92] C. Schnorr. Efficient identification and signatures for smart cards. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology — EUROCRYPT '89*, volume 434 of *Lecture Notes in Computer Science*, pages 688–689. Springer Berlin Heidelberg, 1990.

[93] M. Scott and P. Szczechowiak. Optimizing multiprecision multiplication for public key cryptography. Cryptology ePrint Archive, Report 2007/299, 2007. `http://eprint.iacr.org/2007/299/`.

[94] D. J. Solove. "I've got nothing to hide" and other misunderstandings of privacy. 2007.

[95] M. Sterckx, B. Gierlichs, B. Preneel, and I. Verbauwhede. Efficient implementation of anonymous credentials on java card smart cards. In *1st International Workshop on Information Forensics and Security – WIFS 2009*, pages 106–110, Dec 2009.

[96] STMicroelectronics. ST23ZL48 data brief. `https://www.commoncriteriaportal.org/files/epfiles/ANSSI-CC-cible_2010-02en.pdf`, 2009.

[97] STMicroelectronics. SA23YR48B / SB23YR48B / SA23YR80B / SB23YR80B security target - public version - common criteria for it security evaluation. `http://www.st.com/st-web-ui/static/active/en/resource/technical/document/data_brief/CD00239124.pdf`, 2013.

[98] L. A. Sweeney. *Computational Disclosure Control: A Primer on Data Privacy Protection*. PhD thesis, 2001. AAI0803469.

[99] I. Systems. IAR C/C++ Compiler user guide for texas instruments' MSP430 microcontroller family. `http://ftp.iar.se/WWWfiles/msp430/webic/doc/EW430_CompilerReference.pdf`, 2015.

[100] P. Szczechowiak, A. Kargl, M. Scott, and M. Collier. On the application of pairing based cryptography to wireless sensor networks. In *Proceedings of the Second ACM Conference on Wireless Network Security*, WiSec '09, pages 1–12, New York, NY, USA, 2009. ACM.

[101] H. Tews and B. Jacobs. Performance issues of selective disclosure and blinded issuing protocols on java card. In O. Markowitch, A. Bilas, J.-H. Hoepman, C. Mitchell, and J.-J. Quisquater, editors, *Information Security Theory and Practice. Smart Devices, Pervasive Systems, and Ubiquitous Networks – WISTP 2009*, volume 5746 of *Lecture Notes in Computer Science*, pages 95–111. Springer Berlin Heidelberg, 2009.

[102] E. Tromer, D. Osvik, and A. Shamir. Efficient cache attacks on aes, and countermeasures. *Journal of Cryptology*, 23(1):37–71, 2010.

[103] R. Verdult and F. Kooman. Practical attacks on nfc enabled cell phones. In *Near Field Communication (NFC), 2011 3rd International Workshop on*, pages 77–82, Feb 2011.

[104] P. Vullers and G. Alpár. Efficient selective disclosure on smart cards using idemix. In S. Fischer-Hübner, E. de Leeuw, and C. Mitchell, editors, *Policies and Research in Identity Management*, volume 396 of *3rd IFIP WG 11.6 Working Conference on Advances in Information and Communication Technology – IDMAN 2013*, pages 53–67. Springer Berlin Heidelberg, 2013.

[105] E. Wenger, T. Unterluggauer, and M. Werner. 8/16/32 shades of elliptic curve cryptography on embedded processors. In G. Paul and S. Vaudenay, editors, *Progress in Cryptology – INDOCRYPT 2013*, volume 8250 of *Lecture Notes in Computer Science*, pages 244–261. Springer International Publishing, 2013.

[106] E. Wenger and M. Werner. Evaluating 16-bit processors for elliptic curve cryptography. In E. Prouff, editor, *Smart Card Research and Advanced Applications*, volume 7079 of *Lecture Notes in Computer Science*, pages 166–181. Springer Berlin Heidelberg, 2011.

[107] H. Zhang, J. Gummeson, B. Ransford, and K. Fu. Moo: A batteryless computational RFID and sensing platform. `https://web.cs.umass.edu/publication/docs/2011/UM-CS-2011-020.pdf`, 2011.