Efficient E-cash in Practice: NFC-based Payments for Public Transportation Systems

Gesine Hinterwälder[1], Christian T. Zenger [1,2], Foteini Baldimtsi[3], Anna Lysyanskaya[3], Christof Paar[1,2], and Wayne P. Burleson[1]

[1] Department of Electrical and Computer Engineering, University of Massachusetts Amherst
{hinterwalder,burleson}@ecs.umass.edu
[2] Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
{christian.zenger,christof.paar}@rub.de
[3] Computer Science Department, Brown University
{foteini,anna}@cs.brown.edu

Abstract. Near field communication (NFC) is a recent popular technology that will facilitate many aspects of payments with mobile tokens. In the domain of public transportation payment systems electronic payments have many benefits, including improved throughput, new capabilities (congestion-based pricing etc.) and user convenience. A common concern when using electronic payments is that a user's privacy is sacrificed. However, cryptographic e-cash schemes provide provable guarantees for both security and user privacy. Even though e-cash protocols have been proposed three decades ago, there are relatively few actual implementations, since their computation complexity makes an execution on lightweight devices rather difficult. This paper presents an efficient implementation of Brands [11] and ACL [4] e-cash schemes on an NFC smartphone: the BlackBerry Bold 9900. Due to their efficiency during the spending phase, when compared to other schemes, and the fact that payments can be verified offline, these schemes are especially suited for, but not limited to, use in public transport. Additionally, the encoding of validated attributes (e.g. a user's age range, zip code etc.) is possible in the coins being withdrawn, which allows for additional features such as variable pricing (e.g. reduced fare for senior customers) and privacy-preserving data collection. 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.

1 Introduction

In the year 2011 52% of the world population lived in urban areas. It is anticipated that the urbanization trend is going to continue, leading to an expected 67% of the population living in urban areas by the year 2050 [35]. This trend calls for well functioning public transportation systems, to ensure the mobility of people and limit air pollution in cities [32].

This work is supported by the NSF under CNS-0964641 and CNS-0964379. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

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. But, the payment process for trips should not impact the smooth operation of a transportation system. Hence, a payment needs to be executed quickly. This favors the use of electronic payment systems, which can lead to a better throughput and greater user convenience while at the same time allowing for congestion-based pricing. For the 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 system. However, currently employed electronic public transportation payment systems have suffered security attacks [20, 3], and they do not incorporate means to protect the user's (locational) privacy. For example it is reported that "in the period from August 2004 to March 2006 alone, the Oyster system was queried 409 times"[31], which shows that location data about customers is collected and stored and later used by other agencies. "Anonymous" cards are offered in some systems, but even with those cards user privacy can be sacrificed. The reason is that there is a unique identifier assigned in each card which makes payments made using the same card linkable. So a natural question is how to get the best of both worlds: all the benefits of electronic payments in public transportation systems but without sacrificing user privacy?

Cryptographic techniques make this possible. 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 describes the interaction between three types of entities: the bank, users, and shops. Monetary value is represented by electronic coins, which are pieces of data blindly signed by the bank. The bank is the only entity able to generate coins. It issues coins to a user, who utilizes them to pay at a shop. In an electronic coin its serial number as well as the identity of its possessing user is encoded in a blinded fashion. During spending this serial number and the user's identity are not revealed. At a later point in time the shop deposits the coins that he received from users to his bank account. During or after the deposit process the bank checks, whether a deposited coin had been deposited before. If so the bank can also check, whether a shop deposited the same coin twice, or whether a user double-spent a coin, in which case his identity will be revealed. An important property of certain e-cash schemes is that they support the encoding of users' attributes into coins (i.e. user's age or zip code). This is very convenient for two reasons: (1) it allows the collection of meaningful user data in order to analyze and improve the system in a privacy preserving way (2) it allows to implement additional features in the system such as variable pricing (e.g. reduced fares for senior customers).

This paper presents an implementation of several e-cash schemes that suit the transportation setting on a smartphone which can be used as a potential payment device. 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 smart cards 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 rollout.

Several attacks on NFC enabled mobile phones have been shown, yet most of them target the use of passive tags [36, 28], while we make use of the card emulation mode whose security greatly depends on its implementation (in our case the Blackberry Java API). The main threats introduced by the NFC communication link include: eavesdropping, data corruption/modification or insertion, as well as denial-of-service attacks [22]. Eavesdropping is not so harmful for our system 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 assumed not to reveal private information and (2) if an attacker knew the representation of a user's coin, he could not use it to pay for a trip, as knowing the user's secret key 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 these denial-of-service attacks is very hard as known from other contactless communication links. Relay attacks are also considered a threat for NFC [36]. However, in the transportation domain it is hard to realize them in practice since it would require an attacker to bring one device in close proximity to the payment machine and another one in close proximity to the user device, while additionally having access to the e-cash application on the user device. The user interface and the NFC properties can prevent this attack easily by deactivating the NFC functionality, which is a further benefit of relying on an NFC smartphone rather than on contactless smart cards.

1.1 Related Work

E-cash In 1982, Chaum [15] was the first to propose the idea of an e-cash system that allows anonymous, unlikable payments, which are secure against double spending in the offline setting. Following Chaum's paradigm many schemes were proposed [11, 16, 7, 12, 14]. The one due to Brands [11] is known for its efficiency during spending, however, a formal proof of security has never been given for it and it has been recently shown that it cannot be proven secure in the Random Oracle model using currently known techniques [5]. In 2001 Masayuki Abe proposed a three-move blind signature scheme [2] that can be extended to e-cash, is only slightly less efficient compared to Brands' and has a proof of security in the RO model for concurrent composition. However, he later found together with Ohkubo that his proof suffered some restrictions, since it was only valid for an adversary with overwhelming success probability, and they gave a new proof in the generic model [29]. Very recently, Baldimtsi and Lysyanskaya proposed Anonymous Credentials Light (ACL) [4], which extends Abe's scheme to allow the encoding of attributes into coins while preserving its efficiency. In contrast to Abe's proof, their proof of security goes through since they limit their attention to sequential composition only.

Privacy-preserving Payment Schemes Sadeghi et al. [33] and Blass et al. [9] proposed RFID-based, privacy-preserving e-ticket schemes for transit applications which are limited to only protect the users' privacy against outsiders and not against the transportation authority. Pirker et al. described how to make use of certain hardware capabilities of some mobile phones to build a secure, NFC-based, privacy-preserving, prepaid payment system [30], but the system requires the devices accepting a payment to be online at all times. Heydt-Benjamin et al. [23] proposed the use of recent advances in anonymous credentials and e-cash to design offline privacy-preserving public transportation payment systems. They consider a hybrid system with two kinds of tickets: passive RFID transponders and embedded systems such as cell phones.

E-cash Implementations Implementations of anonymous credentials on Java cards have been presented in [8] and [6]. Hinterwälder et al. presented an implementation of Brands' e-cash scheme for a computational RFID tag [24]. It is shown that it is feasible to execute the spending part efficiently on the tag, while the execution of the withdrawal part is still problematic. Derler et al. [19] implemented an NFC-based mobile ticketing system, which is based on Brands' private credential scheme. Execution times of several seconds are achieved, which can be a limiting factor in some application settings. Similarly, [18] presented a PDA implementation of an offline e-cash scheme that is based on Brands', which achieves an execution time of several seconds for the withdrawal phase.

1.2 Contributions

The work at hand presents a detailed description of how to encode attributes in Brands' e-cash scheme [11] and how to use ACL [4] for e-cash. Although the construction of Brands' e-cash was given in detail, it was never explicitly described how to use it combined with attributes. We also explain how ACL [4] can be used as an e-cash scheme, although it is pretty straightforward. The main contribution of this paper is that we present NFC-smartphone implementations of: Brands' scheme (with and without attributes), Abe's e-cash scheme (which does not support the encoding of attributes) and ACL and we compare and evaluate the performance of those four different schemes (Section 5). Our results are very promising: first of all we have fully realized the *first efficient and practical implementation of e-cash in smartphones*, and moreover, we show that a provable secure e-cash scheme like ACL is actually practical and has running time comparable to those of schemes without rigorous security proofs.

Our implementation is based on Elliptic Curve Cryptography (ECC), which is the most efficient established public-key primitive. The device used is 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 have developed a subtle technique that enables us to use the *ECDHKeyAgreement* class for calculating the scalar multiplication on an elliptic curve, which is present in this and other Java APIs. 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 the x-coordinate is needed for the ECDH key agreement. Our technique shows, how to efficiently recover the y-coordinate for those cases. Using these techniques, we achieve execution times that meet real-world requirements. Interestingly, our results show, that the computation necessary to execute the different schemes can easily be handled by a device as the BlackBerry Bold, whereas the component limiting the execution time is the NFC communication link.

Yet, this work is not limited to demonstrating and evaluating implementation results of a theoretical concept. We also present several ideas of why e-cash fulfills the unique requirements of transportation payment systems, and hence show how to make use of efficient e-cash in practice.

2 Payment System Requirements of the Transportation Setting

Several features of the considered e-cash schemes are especially useful for the design of a payment system that fulfills the requirements of transportation payments.

Verifying a Payment Offline We envision a scenario that is based on currently employed transportation payment systems, where a user buys fares at a vending machine and pays at an entrance point of the transportation system. For example in the case of buses it cannot be assumed that the device granting access to users is permanently connected to the back-end system of the transportation authority. Yet, we assume temporary connection to transfer the data of collected payments. Consequently, *verifying a payment needs to be possible in an offline fashion*. This holds for the chosen schemes, where a verification of the payment does not require access to the database. However, in this case fraud cannot be detected at the time of the payment, as it requires comparing the received data with the database. Alternatively the chosen e-cash schemes reveal a crime after the fact, and allow a user to be penalized, when misusing the system.

Modular Payment System In case of multiple transportation authorities the e-cash concept offers great convenience advantages for users, as it allows for multiple banks and shops. A user does not need multiple payment devices to use different transportation systems. Rather he can withdraw 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.

Different Denominations A transportation authority needs to offer different fare prices. This can be accomplished by assigning a low monetary value to coins and letting the user spend many coins to pay for a fare. Yet, spending many coins increases the execution time of a payment which should remain low. In the e-cash sending we can conveniently allow for different denominations of coins and thus reduce the number of coins that need to be spent. One way to accomplish this is to have the bank possess multiple public keys, one key for each possible denomination of coins.

Encoding Attributes The collection of user data is important to analyze and improve the system in order to adapt it to customer needs[1]. However, the collection of data

[1] For example the number of elevators that should be provided at a station could be planned better, when knowing, how many people require wheelchair accessibility.

should be done in a way that wouldn't sacrifice a user's privacy. With the use of e-cash this can be achieved by encoding attributes into coins. Those attributes allow the user to reveal some information, as for example his zip code, while keeping further information hidden, and hence hiding his identity. Apart from private data collection, encoding attributes into coins allows for private variable pricing. The system could require the users to encode specific information like their age and then, when the coin is spent, compute the right fare according to the presented attributes.

3 E-cash with Attributes

In this section we describe how e-cash with attributes works and we provide instantiations based on Brands' e-cash scheme [11] and ACL [4]. The descriptions are tailored to match the transportation setting but can be easily adopted for other settings as well. To our knowledge this is the first time that an explicit description of how Brands and ACL schemes can be used as e-cash schemes with attributes is given. Although at the end we will compare the implementation results of Brands with attributes, Brands without attributes, ACL and Abe's e-cash (which is essentially ACL without attributes), here we will only provide the detailed construction of the attribute supporting schemes. In our descriptions we will point out the differences between the attribute and the corresponding non-attribute version.

A transportation payment system based on e-cash is described as an interaction between three kinds of players: the transportation authority TA (with vending machines that are connected to its database), the users of the system U and the payment machines M that are placed at the entrance points and, after receiving a valid payment, grant a user access to the transportation system.

Setup. During the setup phase the transportation authority (or another trusted authority) generates the public parameters for the system together with the TA's secret key.

Account opening. Initially, users need to register with the TA and open an account. To do so, a user U, would have to present some form of identification (e.g. a passport) to the TA and provide a cryptographic commitment C for a set of attributes $(L_1, \ldots, L_n)$ that are required for the system and for his public key $pk_U = g^{sk_U}$. The attribute types and their order are determined by the TA during the setup of the system. A possible setting is to use attribute L_1 for the user's secret key ($sk_U = L_1$), L_2 for his age, L_3 for his zip code etc. For the transportation setting we assume that revealing attributes in clear is good enough and does not violate user's privacy [2].

Withdrawal. Whenever U wants to withdraw coins from a vending machine, he first needs to prove ownership of his account and then he runs the withdrawal protocol. In order to prove ownership of his account, we require U to form a digital signature (i.e. a Schnorr signature [34]) on a message that describes the number of coins he wishes to withdraw and include some kind of timestamp. This is useful for two reasons: (1) the TA

[2] If there is need for extra privacy we could require for all attributes that are revealed during the spending phase to be binary attributes i.e. L_2 equals 0 if the user is more than 18 and less than 65 years old and 1 otherwise (used for age discounts). This way we avoid *range proofs* which are rather costly. Besides, for the transportation setting binary attributes would work rather well, given that we need the attributes mainly for variable pricing or private data collection.

can easily identify the user by checking whether the signature is valid under the user's public key and (2) it provides an extra level of security against a man-in-the-middle who may intercept the communication and try to withdraw more coins. Note that depending on the protocol the adversary may not be able to actually spend these coins (since he needs the user's secret key to execute the payment protocol). However, he can still hurt U by reducing his account balance. Alternatively, we could require the execution of the identification phase for every single coin withdrawn but this is obviously less efficient than identifying U once for all the coins he wishes to withdraw.

Spending. In order for a user U to spend a coin at a payment machine, U runs the spending protocol as described in the corresponding e-cash system. Moreover, he might be asked (by the system) to reveal some of his attributes. When a User reveals an attribute L_j he can either send the attribute value in clear or provide a proof of knowledge of that attribute. Note that in both cases he needs to prove that the attribute he reveals is the same one he committed to during the account opening phase.

Deposit. The payment machines present the spending transcript to the TA. Here, an extra mechanism, called *double spending detection*, is required to protect the TA against cheating users. The double spending detection is executed off-line in order to detect and penalize users, who spent the same coin more than once. In order for double spending detection to be possible, the TA needs to preserve a special database where all the deposited coins are stored. Obviously we cannot assume that all the coins are stored there forever, thus, we need to introduce some kind of *coin expiration date* after which the coin will be deleted from the database. This could be done either by having the expiration date encoded as an attribute in the coins (so the user would have to prove that the coin is still valid during spending) or by changing the TA's public key (i.e. once a year) and setting older coins invalid.

3.1 Brands' E-cash with Attributes

We describe how Brands' e-cash scheme [11] can be modified to support the encoding of users' attributes. The main differences between the attribute and the non-attribute version can be found in the account opening and the spending phases (when attributes are revealed). The actual withdrawal protocol is essentially the same.

Setup. The TA (or another trusted authority) picks a group G of prime order q, generators $h, g, g_1, \ldots, g_n$, where n is the maximum number of attributes needed for the system, and a hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. The public key of the TA is $y = g^x$, where $x \in \mathbb{Z}_q$ is its secret key.

Account Opening. The account opening procedure of Brands with attributes is presented in Table 1. When a user U wants to open an account with the TA, he first presents an identification document to the TA and then encodes his attributes (L_1, L_2, $\ldots$, L_n $\in \mathbb{Z}_q$) into a commitment I[3]. The value L_1 (which is not revealed to the TA) serves as the user's secret key. His public key is $pk_U = g_1^{L_1}$. Then, U provides a proof of knowledge π that he knows the opening of the commitment I and that the same value

[3] Obtaining these attributes can be done similar to a credential system. A user would either have to reveal his attributes to the TA when committing to them or obtain them from some other trusted authority and then prove knowledge of them to the TA.

L_1 has been used to generate I and pk_U. The proof π can be computed as a standard Schnorr *AND proof* of knowledge of several discrete logarithms [34, 25]. Upon receiving π, I, pk_U, the TA checks the validity of the proof, stores the user's information and computes $z = (Ih)^x$ which is send to U. The corresponding protocol for Brands without attributes only requires the user to prove knowledge of his secret key.

Table 1: Brands' with Attributes Account Opening Protocol

TA($y = g^x$)		U($pk_U = g_1^{L_1}$)
		$I = g_1^{L_1} \dots g_n^{L_n}$ $\pi = PK\{(\Lambda_1, \dots, \Lambda_n) :$ $I/pk_U = g_2^{\Lambda_2} \dots g_n^{\Lambda_n} \wedge pk_U = g_1^{\Lambda_1}\}$
Verify π	$\xleftarrow{I, pk_U, \pi}$	If $Ih \neq 1$
Store identifying information of U together with I		
$z = (Ih)^x$	$\xrightarrow{z}$	Store z

Withdrawal. To withdraw k coins, U first has to identify himself to the TA by proving knowledge of his secret key and then claiming how many coins he wants to withdraw. These two actions can be achieved simultaneously by computing a Schnorr signature [34] $\sigma(m)$ on a message m of the form: $m =$"# of coins + time/date". The TA will authorize the user to perform the withdrawal, if the signature validates under the user's public key and there are sufficient funds in his account. Then the user runs the withdrawal protocol k times, once for each coin he withdraws (Table 2).

Table 2: Brands' with Attributes Withdrawal Protocol

TA($y = g^x$)		U($pk_U = g_1^{L_1}$)
$w \in_{\mathcal{R}} \mathbb{Z}_q$, $a = g^w$, $b = (Ih)^w$	$\xrightarrow{a,b}$	$s \in_{\mathcal{R}} \mathbb{Z}_q^*$, $A = (Ih)^s$, $z' = z^s$ $x_0, x_1, \dots, x_n, u, v \in_{\mathcal{R}} \mathbb{Z}_q$ $B = A^{x_0} g_1^{x_1} \dots g_n^{x_n}$ $a' = a^u g^v,\ b' = b^{su} A^v$ $c' = \mathcal{H}(A, B, z', a', b')$
	$\xleftarrow{c}$	$c = c'/u \mod q$
$r = cx + w \mod q$	$\xrightarrow{r}$	$g^r \stackrel{?}{=} y^c a$, $(Ih)^r \stackrel{?}{=} z^c b$ $r' = ru + v \mod q$

For each coin, U needs to store the values: $A, B, \text{sign}(A, B) \hat{=} A, B, z', a', b', r'$ together with s and the values $x_0, x_1, \dots, x_n$, where $\text{sign}(A, b)$ is essentially a blind Chaum-Pedersen signature [17]. In order to verify the signature one needs to check whether: $g^{r'} = y^{c'} a'$ and $A^{r'} = z'^{c'} b'$. Note that the basic difference of Brands withdrawal when using attributes is found in the computation of B since you need to pick as many random values x_i as the number of attributes in the scheme and compute $B = A^{x_0} g_1^{x_1} \dots g_n^{x_n}$. Those random values x_i will be used later, during the spending phase in order for the user to prove knowledge of his attributes. For Brands withdrawal without attributes the value B is computed as $B = g_1^{x_1} g_2^{x_2}$: only two random values x_1 and x_2 are required and it will be used for proving knowledge of user's secret key in the spending phase.

Spending. When U spends a coin to M (with identifying information I_{M}) the spending protocol is executed. In Table 3 we present Brands spending protocol when supporting the encoding of but not revealing any attributes. During this phase the user presents the coin: $A, B, \text{sign}(A, B)$ to M and also proves knowledge of the representation of A (i.e. U proves knowledge of his attributes). After the transaction and if the signature verifies (i.e. the coin is valid), M saves the payment transcript consisting of $A, B, \text{sign}(A, B)$, $(R, r_1, \ldots, r_n)$ and the time stamp date/time.

This protocol is significantly less efficient compared to Brands e-cash without attributes, since the user needs to prove knowledge of the representation of A, which now includes $n + 1$ exponents. For Brands without attributes the user only needs to compute r_1 and r_2 to prove knowledge of his secret key.

Table 3: Brands' with Attributes Spending Protocol when not Revealing Attributes

U$(pk_U = g_1^{L_1})$		M
	$\xrightarrow{A,B,\text{sign}(A,B)}$	$A \overset{?}{\neq} 1$
$r_1 = -dL_1 + x_1 \mod q$	$\xleftarrow{d}$	$d = \mathcal{H}_0(A, B, I_{\mathsf{M}}, \text{date/time})$
$\ldots$		
$r_n = -dL_n + x_n \mod q$		
$R = d/s + x_0 \mod q$	$\xrightarrow{(R,r_1,\ldots,r_n)}$	$g_1^{r_1} \ldots g_n^{r_n} h^{-d} \overset{?}{=} A^{-R}B$
		Verify sign(A, B)

What if U additionally wants to reveal an attribute, say L_j, during the spending protocol (remember that we assume that attributes are revealed in clear)? In order to reveal L_j, U does not need to compute r_j, when receiving the value d from M. Instead, he sends the attribute value L_j together with its blinding value x_j in his response to M, which is shown in Table 4. Thus, revealing a larger number of attributes reduces the user-side's computation, but increases the amount of data that U sends to the payment machine M.

Deposit. In order to deposit a coin, M submits the payment transcript to the TA. The TA first checks the validity of the coin (i.e. it verifies sign(A, B) and checks whether $g_1^{r_1} \ldots g_n^{r_n} h^{-d} \overset{?}{=} A^{-R}B$) and then queries the 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 TA 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 TA will store $(A, d, R, r_1, \ldots, r_n)$ in her database. However, if the coin had been recorded, it means that it was spent twice by U (we assume that the payment machines in our system are trusted and will not try to submit the same coin twice: yet, this could easily be checked by storing the payment machine's identification I_{M} together with each coin that is stored). If a coin had been double spent the identity of the cheating user can be revealed by computing:

Table 4: Brands' with Attributes Spending Protocol when Revealing the Attribute L_j

$\mathsf{U}(pk_U = g_1^{L_1})$		M
	$\xrightarrow{A,B,\text{sign}(A,B)}$	$A \overset{?}{\neq} 1$
$r_1 = -dL_1 + x_1 \mod q$	$\xleftarrow{d}$	$d = \mathcal{H}_0(A, B, I_{\mathsf{M}}, \text{date/time})$
$\dots$		
$r_{j-1} = -dL_{j-1} + x_{j-1} \mod q$		
$r_{j+1} = -dL_{j+1} + x_{j+1} \mod q$		
$\dots$		
$r_n = -dL_n + x_n \mod q$		
$R = d/s + x_0 \mod q$	$\xrightarrow{(R,r_1,\dots r_{j-1},r_{j+1},\dots,r_n)(L_j,x_j)}$	$g_1^{r_1} \dots g_j^{-dL_j+x_j} \dots g_n^{r_n} h^{-d}$ $\overset{?}{=} A^{-R}B$ Verify sign(A, B)

$I = g_1^{(r_1-r_1')/(R-R')} \dots g_n^{(r_n-r_n')/(R-R')}$ which was stored together with some identifying information of U during the account registration phase.

3.2 ACL E-cash with Attributes

The ACL scheme [4] is a recent, very efficient "linkable" [4] anonymous credential system, which was constructed on top of Abe's blind signature scheme [2]. It is straightforward to use a "linkable" anonymous credential scheme in order to describe e-cash with attributes since coins are essentially "single-use" credentials. In this section we explicitly describe how the ACL scheme can be used as e-cash with attributes. Note that Abe's blind signature scheme itself is immediately an e-cash scheme (without attributes though) and has been described in the past [2]. Thus, as mentioned above, we will not provide a detailed description of Abe's scheme; instead we will just point out the differences while describing ACL.

Setup. The setup phase of ACL e-cash is similar to the original ACL scheme. The TA chooses a group G of order q, a generator g and a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$. It also picks $z, h, h_0, h_1, h_2, \dots h_n \in_{\mathcal{R}} G$, where n is the maximum number of attributes. The secret key of the transportation authority is $x \in_{\mathcal{R}} \mathbb{Z}_q$, the public key is: $y = g^x$ and z is the "tag public key".

Account opening. When a user U with attributes $(L_2, \dots, L_n)$, secret key $L_1 \in_{\mathcal{R}} \mathbb{Z}_q$ and public key $pk_U = h_1^{L_1}$ wants to open an account at the TA he presents a valid identification document and commits to his attributes and public key, as shown in Table 5. For each User the TA stores: $pk_U, C/pk_U$ and a copy of his identification document. U also needs to store the randomness R that corresponds to his commitment C.

Withdrawal. To withdraw k coins from his account U first identifies himself to the transportation authority (similar to Section 3.1). Then U runs the ACL blind signature protocol k times (Table 6) once for every coin.

After each execution of the withdrawal protocol, U obtains a $coin = (\zeta, \zeta_1, \rho, \omega, \rho_1', \rho_2', \omega')$, which he stores together with rnd, τ and γ. Note that he omits to compute and store the

[4] In a "linkable" anonymous credential system a user can only use a credential once if he does not want his transactions to be linkable.

Table 5: ACL with Attributes Account Opening Protocol

TA$(y = g^x, z)$		U$(pk_U = h_1^{L_1})$
		$R \in_{\mathcal{R}} \mathbb{Z}_q, C = h_0^R h_1^{L_1} \prod_{i=2}^n h_i^{L_i}$ $\pi = PK\{(P, \Lambda_1, \ldots, \Lambda_n) :$
Check π Store identifying information of U together with C	$\xleftarrow{C/pk_U, pk_U, \pi}$	$C/pk_U = h_0^P h_2^{\Lambda_2} \ldots h_n^{\Lambda_n} \wedge pk_U = h_1^{\Lambda_1}\}$

Table 6: ACL with Attributes Withdrawal Protocol

TA$(y = g^x, z)$		U$(pk_U = h_1^{L_1})$
$rnd \in_{\mathcal{R}} \mathbb{Z}_q, z_1 = Cg^{rnd}, z_2 = z/z_1$ $u, c', r'_1, r'_2 \in_{\mathcal{R}} \mathbb{Z}_q$		
$a = g^u, a'_1 = g^{r'_1} z_1^{c'}, a'_2 = h^{r'_2} z_2^{c'}$	$\xrightarrow{rnd, a, a'_1, a'_2}$	$z_1 = Cg^{rnd}, \gamma \in_{\mathcal{R}} \mathbb{Z}_q^*$ $\zeta = z^\gamma, \zeta_1 = z_1^\gamma, \zeta_2 = \zeta/\zeta_1$ $\tau \in_{\mathcal{R}} \mathbb{Z}_q, \eta = z^\tau$ Check whether $a, a'_1, a'_2 \in G$ $t_1, t_2, t_3, t_4, t_5 \in_{\mathcal{R}} \mathbb{Z}_q$ $\alpha = ag^{t_1} y^{t_2}, \alpha'_1 = a_1'^{\gamma} g^{t_3} \zeta_1^{t_4}$ $\alpha'_2 = a_2'^{\gamma} h^{t_5} \zeta_2^{t_4}$ $\varepsilon = \mathcal{H}(\zeta, \zeta_1, \alpha, \alpha'_1, \alpha'_2, \eta)$
$c = e - c' \mod q$	$\xleftarrow{e}$	$e = (\varepsilon - t_2 - t_4) \mod q$
$r = u - cx \mod q$	$\xrightarrow{c, r, c', r'_1, r'_2}$	$\rho = r + t_1 \mod q$ $\omega = c + t_2 \mod q$ $\rho'_1 = \gamma r'_1 + t_3 \mod q$ $\rho'_2 = \gamma r'_2 + t_5 \mod q$ $\omega' = c' + t_4 \mod q$

value μ of the original ACL scheme. Instead, during the spending phase (Table 7), he computes μ_p to "bind" the coin to a specific transaction. For each withdrawn coin, the TA stores z_1 together with the public key of U that this coin corresponds to (z_1 is going to be used to reveal the identity of the user in case of double-spending). The withdrawal of Abe's e-cash is essentially the same.

Spending. U releases the coin to the merchant together with ϵ_p and μ_p. The spending protocol for ACL e-cash and Abe's e-cash is identical and presented in Table 7.

In order for a user U to also reveal an attribute L_j during spending, he will have to execute the Revealing Attribute L_j protocol (Table 8) additionally to the Spending protocol. In this protocol he needs to prove that the attributes on his initial commitment correspond to the attributes encoded in the withdrawn coin (i.e. in value ζ_1). In other words, the user needs to provide a proof of equality of committed values under different commitment keys and bases [25], for a new commitment C' that the user computes for the attribute L_j and ζ_1 in which the attributes are encoded.

Table 7: ACL with Attributes Spending Protocol

$\mathsf{U}(pk_U = h_1^{L_1})$		M
$\epsilon_p = \mathcal{H}(z^\tau, \text{coin}, \text{desc})$	$\xleftarrow{\text{desc}}$	$desc = \mathcal{H}(I_{\mathsf{M}}, \text{date/time})$
$\mu_p = \tau - \epsilon_p\gamma \mod q$	$\xrightarrow{\varepsilon_p,\mu_p,\text{coin}}$	$\zeta \overset{?}{\neq} 1$
		$\epsilon_p \overset{?}{=} \mathcal{H}_4(z^{\mu_p}\zeta^{\epsilon_p},\text{coin,desc}) \mod q$
		$\omega + \omega'$
		$\overset{?}{=} \mathcal{H}(\zeta, \zeta_1, g^\rho y^\omega, g^{\rho'_1}\zeta_1^{\omega'}, h^{\rho'_2}\zeta_2^{\omega'}, z^{\mu_p}\zeta^{\epsilon_p}) \mod q$

Table 8: ACL with Attributes Revealing the Attribute L_j Protocol

$\mathsf{U}(pk_U = h_1^{L_1})$		M
recall $\zeta_1 = h_0^{R\gamma}h_1^{L_1\gamma}\ldots h_n^{L_n\gamma}g^{rnd\gamma}$		
$rnd' \in_R \mathbb{Z}_q$		
$C' = h_j^{L_j\gamma}g^{rnd'\gamma}$		
$= h_j^{L_j\gamma}1^{R\gamma}1^{L_1\gamma}\ldots 1^{L_{j-1}\gamma}1^{L_{j+1}\gamma}\ldots 1^{L_n\gamma}g^{rnd'\gamma}$		
$r, r', r_0, \ldots, r_n \in_R \mathbb{Z}_q$		
$\widetilde{\zeta_1} = h_0^{r_0}\ldots h_n^{r_n}g^r$		
$\widetilde{C'} = 1^{r_0}1^{r_1}\ldots h_j^{r_j}\ldots 1^{r_n}g^{r'}$		
$c = \mathcal{H}(\zeta_1, \widetilde{\zeta_1}, C', \widetilde{C'}, \text{date/time})$		
$s_0 = r_0 + cR\gamma$		
$s_1 = r_1 + cL_1\gamma$		
$\ldots$		
$s_n = r_n + cL_n\gamma$		
$s = r + c\,rnd\,\gamma$		
$s' = r' + c\,rnd'\,\gamma$	$\xrightarrow{L_j,\zeta_1,\widetilde{\zeta_1},C',\widetilde{C'},s_0,\ldots,s_n,s,s'}$	$c = \mathcal{H}(\zeta_1, \widetilde{\zeta_1}, C', \widetilde{C'}, \text{date/time})$
		$\widetilde{\zeta_1}\zeta_1^c \overset{?}{=} h_0^{s_0}\ldots h_n^{s_n}g^s$
		$\widetilde{C'}C'^c \overset{?}{=} 1^{s_0}1^{s_1}\ldots h_j^{s_j}\ldots 1^{s_n}g^{s'}$

From a theoretical point of view the user side of the spending protocol of ACL (Table 7), when supporting the encoding but not revealing attributes, is more efficient than the one of Brands' scheme (Table 3), since, following the trick that was first suggested by Abe [2], the user only needs to compute the updated μ_p value for the coin verification instead of proving knowledge of all his attributes. In practice this depends on the relative cost for executing modular arithmetic in $\mathbb{Z}_q$ compared to the hash function and the communication cost.

Deposit. During deposit, the payment machine M sends the coin as well as ε_p, μ_p, $desc$ and the date and time of the transaction to the TA which verifies that both the coin is valid and $desc$ correctly encodes date/time and I_M. Double spending can be checked later in an off-line fashion. In order to do that the TA needs to check whether a coin has been deposited with two different $desc$ and $desc'$. In this case we will have (ε_p, μ_p) and (ε'_p, μ'_p) and the TA can calculate: $\gamma = (\mu'_p - \mu_p)/(\epsilon_p - \epsilon'_p)$ and $z_1 = \zeta_1^{1/\gamma}$, the TA

can find the user to whom z_1 was given and “punish” him. A useful observation is that in the ACL scheme, when a user is found cheating, his identity can be revealed without the TA learning his secret key and attribute values. From a privacy point of view this is much better, since the user does not need to form a new secret key and commit to his attributes all over again, after getting “punished” by the TA .

4 Framework Implementation

We will now describe important aspects of our implementation. In our measurement setup the terminal, which represents the vending as well as the payment machines, 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[5] provided by RIM.

4.1 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 which the NFC-smartphone emulates a standard-conform smart card. Building the payment system on standard appliances, makes it conform to already installed payment infrastructure and hence facilitates deployment. The underlying standard of the 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 smart card using so called Application Protocol Data Units (APDUs). The reader initializes the communication by sending a command APDU to the smart card. The smart card executes this command and replies with a response APDU. This communication procedure is specified in standard ISO/IEC 7816-4. Note that the size of an APDU is limited to 256 bytes, which impacts the execution time of the protocols, as will be discussed further in Section 5.

4.2 Cryptographic Framework

We base the schemes on Elliptic Curve Cryptography, and deduce from [10] that a 160-bit elliptic curve presents sufficient security for a micro-payment system. We chose the standardized curve *secp160r1* from [13]. The underlying prime field of this curve $\mathbb{Z}_p$ is based on a generalized Mersenne prime $p = 2^{160} - 2^{31} - 1$, which allows for an efficient implementation of the curve arithmetic. On the terminal side we use the Bouncy Castle Crypto Library version 1.5[6]. 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

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

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

execution of the payment schemes and is realistic in the transportation setting. However, our investigations focus on the execution of 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 curve *secp160r1*. As such, 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 the 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].

Efficient Execution of EC Scalar Multiplication Using the ECDH Key Agreement An implementation of the scalar multiplication $\boldsymbol{Q_k} = k{\cdot}\boldsymbol{P}$ 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 the ECDHKeyAgreement class. This class offers the method *generateSharedSecret*, which executes a scalar multiplication of the input point $\boldsymbol{P} = (x_P, y_P)$ with the input scalar k. This method executes in 1 ms, but only returns the x-coordinate x_{Q_k} of the resulting point $\boldsymbol{Q_k}$. In the protocols of the considered payment schemes multiple point multiplications have to be executed. Hence, knowledge of the y-coordinate y_{Q_k} of $\boldsymbol{Q_k}$ is essential for further computations.

This drawback can be overcome. Going from the short Weierstrass equation ($y^2 = x^3 + ax + b$), on which the chosen elliptic curve is based, the magnitude of y_{Q_k} can be calculated as $y_{Q_k} = \sqrt{x_{Q_k}^3 + ax_{Q_k} + b} \mod p$.[7] This results in two options for the resulting point $\boldsymbol{Q_k}$:

$$\boldsymbol{Q_k} = (x_{Q_k}, \pm y_{Q_k}) \begin{cases} \boldsymbol{Q_k^{(+)}} & = (x_{Q_k}, +y_{Q_k}) \\ \boldsymbol{Q_k^{(-)}} & = (x_{Q_k}, -y_{Q_k}) \end{cases} \quad (1)$$

To choose the correct option ($\boldsymbol{Q_k^{(+)}}$ or $\boldsymbol{Q_k^{(-)}}$) for $\boldsymbol{Q_k}$ we verify y_{Q_k} over the coherence $\boldsymbol{Q_{k+1}} = \boldsymbol{Q_k}{+}\boldsymbol{P} = (k+1){\cdot}\boldsymbol{P}$. We pick the positive result $+y_{Q_k}$ and calculate the

[7] Algorithm 3.36 in [27] describes how to calculate the square root in $\mathbb{Z}_p$, if $p \equiv 3 \mod 4$, which holds for the chosen curve.

x-coordinate of $\boldsymbol{Q_{k+1}^{(+)}}$ by adding $\boldsymbol{Q_{k}^{(+)}}$ and $\boldsymbol{P}$, using the group law for point addition on an elliptic curve [21]

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

Then we check whether the result is equal to the x-coordinate of $\boldsymbol{Q_{k+1}}$ returned when executing the *generateSharedSecret* function on $(k+1)$ and $\boldsymbol{P}$. While this algorithm, which is summarized as Algorithm 4.1, executes the generateSharedSecret method twice and calculates 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 4.1: Recovering the y-coordinate, when using the ECDH key agreement class for point multiplication

```
Data: input point $P$, input scalar $k$
Result: x- coordinate and y-coordinate of resulting point $Q = (x_Q, y_Q)$
begin
    $x_{Q_k} \leftarrow$ generateSharedSecret($k, P$)
    $x_{Q_{k+1}} \leftarrow$ generateSharedSecret($(k+1), P$)
    $\pm y_{Q_k} = \sqrt{x_{Q_k}^3 + a \cdot x_{Q_k} + b} \mod p$
    $x_{Q_{k+1}}^{(+)} = (\frac{+y_{Q_k} - y_P}{x_{Q_k} - x_P})^2 - x_P - x_{Q_k} \mod p$
    if $x_{Q_{k+1}} == x_{Q_{k+1}}^{(+)}$ then
        return $(x_{Q_k}, +y_{Q_k})$;
    else
        return $(x_{Q_k}, -y_{Q_k})$;
end
```

5 Implementation Results and Evaluation

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 once per year, when creating new public keys of the system each year). We limit the discussion of our results to those time critical parts.

The results for the execution of the withdrawal and the spending phase for all schemes are presented in the Tables 9 and 10 respectively. We present two cases: I) Brands' e-cash scheme, when not allowing, and Abe's scheme, which does not allow, the encoding of attributes, and II) Brands' scheme and ACL when allowing the encoding of two attributes and revealing both of them. Note that the private key of the user is not counted as an attribute, i.e. he 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 attributes if the transportation system requires so, but keep in mind that there is a trade-off between the number of attributes and the spending time.

Table 9: Execution time of withdrawal per coin for I) Brands and Abe not supporting attributes and II) Brands and ACL supporting the encoding of and revealing 2 attributes.

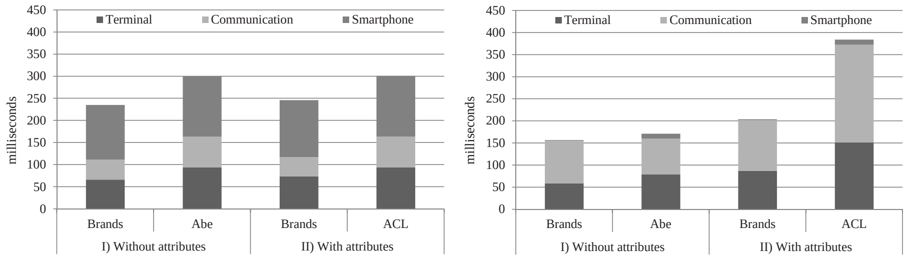
Scheme		Execution time in milliseconds			
		Terminal	Communication	Smartphone	Total
I) Without attributes	Brands	66.1	45.1	123.8	235
	Abe	93.6	69.6	137.5	301
II) With attributes	Brands	73.2	44.1	128.7	246
	ACL	93.6	69.9	137.5	301

Table 10: Execution time of spending per coin for I) Brands and Abe not supporting attributes and II) Brands and ACL supporting the encoding of and revealing 2 attributes.

Scheme		Execution time in milliseconds			
		Terminal	Communication	Smartphone	Total
I) Without attributes	Brands	58.8	96.8	1.4	157
	Abe	79.3	81.0	10.7	171
II) With attributes	Brands	87.3	114.8	2.0	204
	ACL	151.2	221.4	11.4	384

Figure 1 illustrates 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. For example could the TA in the withdrawal protocol of ACL send the number rnd to the user right after generating it. Then while the TA calculates a, a'_1 and a'_2 the user could at the same time calculate z_1, ζ, ζ_1 and ζ_2.

An advantage of the ACL scheme is that for the revealing attributes phase (Table 8) the values $C', \widetilde{\zeta_1}, \widetilde{C'}$ 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 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[31]. Hence, the implementation shows that it is feasible to spend a coin meeting the extreme time constraints of the



(a) Withdrawal protocol

(b) Spending protocol

Fig. 1: Execution times of the (a) withdrawal and (b) spending protocols for the cases I) not supporting attributes and II) supporting the encoding and revealing 2 attributes.

transportation setting. Spending several coins serially exceeds those time constraints. Yet, the execution time could be further reduced by batching the executions that are required for spending several coins.

While the terminal side is represented by a powerful computer 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 4 we focus on the execution time on the payment device and of the communication. The implementation results could be improved when not relying on a Java implementation for the terminal side, which is a realistic scenario, since the TA has full control over those devices.

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/sec, while the maximum bandwidth supported by the NFC standard is 424 kbits/sec. 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 (Figure 1). Moreover, the length of an APDU is limited to 256 bytes. For some protocol steps data had to be sent using two APDUs. Hence, the overall execution time could be improved when allowing longer APDUs.

A further observation is that the computational complexity of Brands' spending protocol, when allowing attributes, decreases the more of them are revealed. Yet, at the same time the data to be communicated increases. For our implementation the increase in data that needs to be communicated, dominates the change in execution 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 11 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 $C', \widetilde{\zeta_1}, \widetilde{C'}$ 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.

Table 11: Coin size (per coin data stored on user device) for the cases I) Brands and Abe not supporting attributes and II) Brands and ACL supporting 2 attributes.

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

In the following we will 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 [26]. In the case of Brands' e-cash with attributes scheme the TA 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 or average 178 MB per day that need to be stored in the database. In the case of ACL the TA has to have two databases. One stores the values z_1 together with the public key pk_U of a user for each withdrawn coin and another one that stores ζ_1, $desc, \varepsilon_p$ and μ_p for each spent coin (186 bytes per coin or 227 MB 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.

6 Conclusion

Brands' [11], Abe's [2] and ACL [4] e-cash are all 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 paper presented an explicit description of e-cash with attributes for both Brands' [11] and ACL [4] schemes. Further it 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 calculate the point multiplication, 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. Phones equipped with Android release, 4.0 (Ice Cream Sandwich, ICS) or higher can easily be extended with Spongy Castle, the repackage of Bouncy Castle for Android. It does support EC key generation, ECDH key exchange and ECDSA signatures. Yet, when using this class the execution of the generateSharedSecret function on the Samsung Galaxy S3 is much slower than the presented results on the BlackBerry Bold 9900. As part of future work use of the Android NDK can be investigated to reach the timing requirements of the transportation setting.

References

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, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 136–151. Springer, 2001.
3. Z. Anderson, R. Ryan, and A. Chiesa. The Anatomy of a Subway Hack: Breaking Crypto RFID's and Magstripes of Ticketing Systems, 2008.
4. F. Baldimtsi and A. Lysyanskaya. Anonymous Credentials Light. *IACR Cryptology ePrint Archive*, 2012:298, 2012.
5. F. Baldimtsi and A. Lysyanskaya. On The Security of One-Witness Blind Signature Schemes. *IACR Cryptology ePrint Archive*, 2012:197, 2012.
6. 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, *CARDIS*, volume 6035 of *Lecture Notes in Computer Science*, pages 209–222. Springer, 2010.
7. M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. Compact E-Cash and Simulatable VRFs Revisited. In H. Shacham and B. Waters, editors, *Pairing*, volume 5671 of *Lecture Notes in Computer Science*, pages 114–131. Springer, 2009.
8. P. Bichsel, J. Camenisch, T. Groß, and V. Shoup. Anonymous credentials on a standard java card. In E. Al-Shaer, S. Jha, and A. D. Keromytis, editors, *ACM Conference on Computer and Communications Security*, pages 600–610. ACM, 2009.
9. E.-O. Blass, A. Kurmus, R. Molva, and T. Strufe. PSP: private and secure payment with RFID. In E. Al-Shaer and S. Paraboschi, editors, *WPES*, pages 51–60. ACM, 2009.
10. 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.
11. S. Brands. Untraceable Off-line Cash in Wallets with Observers (Extended Abstract). In D. R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318. Springer, 1993.
12. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact E-Cash. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321. Springer, 2005.
13. Certicom Research. *Standads for Efficient Cryptography (SEC) 2: Recommended Elliptic Curve Domain Parameters*, version 1.0 edition, 2000.
14. A. H. Chan, Y. Frankel, and Y. Tsiounis. Easy Come - Easy Go Divisible Cash. In K. Nyberg, editor, *EUROCRYPT*, volume 1403 of *Lecture Notes in Computer Science*, pages 561–575. Springer, 1998.
15. D. Chaum. Blind Signatures for Untraceable Payments. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *CRYPTO*, pages 199–203. Plenum Press, New York, 1982.
16. D. Chaum, A. Fiat, and M. Naor. Untraceable Electronic Cash. In S. Goldwasser, editor, *CRYPTO*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer, 1988.
17. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '92, pages 89–105, London, UK, UK, 1993. Springer-Verlag.

18. E. Clemente-Cuervo, F. Rodríguez-Henríquez, D. O. Arroyo, and L. Ertaul. A PDA Implementation of an Off-line e-Cash Protocol. In S. Aissi and H. R. Arabnia, editors, *Security and Management*, pages 452–458. CSREA Press, 2007.
19. D. Derler, K. Potzmader, J. Winter, and K. Dietrich. Anonymous Ticketing for NFC-Enabled Mobile Phones. In L. Chen, M. Yung, and L. Zhu, editors, *INTRUST*, volume 7222 of *Lecture Notes in Computer Science*, pages 66–83. Springer, 2011.
20. F. D. Garcia, G. de Koning Gans, R. Muijrers, P. van Rossum, R. Verdult, R. W. Schreur, and B. Jacobs. Dismantling MIFARE Classic. In S. Jajodia and J. López, editors, *ESORICS*, volume 5283 of *Lecture Notes in Computer Science*, pages 97–114. Springer, 2008.
21. D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.
22. E. Haselsteiner and K. Breitfuß. Security in Near Field Communication (NFC) - Strengths and Weaknesses. http://events.iaik.tugraz.at/RFIDSec06/Program/papers/002n
23. T. S. Heydt-Benjamin, H.-J. Chae, B. Defend, and K. Fu. Privacy for Public Transportation. In G. Danezis and P. Golle, editors, *Privacy Enhancing Technologies*, volume 4258 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2006.
24. G. Hinterwälder, C. Paar, and W. P. Burleson. Privacy preserving payments on computational rfid devices with application in intelligent transportation systems. In J.-H. Hoepman and I. Verbauwhede, editors, *RFIDSec*, volume 7739 of *Lecture Notes in Computer Science*, pages 109–122. Springer, 2012.
25. A. Lysyanskaya. Signature schemes and applications to cryptographic protocol design. In *PhD Thesis*. Massachusetts Institute of Technology, 2002. AAI0804606.
26. M. B. T. A. (MBTA). MBTA ScoreCard 2013 March [Feb'13 Data]. http://www.mbta.com/uploadedfiles/About_the_T/Score_Card/ScoreCard-03.pdf, 2013.
27. A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
28. C. Mulliner. Vulnerability analysis and attacks on nfc-enabled mobile phones. In *ARES*, pages 695–700. IEEE Computer Society, 2009.
29. M. Ohkubo and M. Abe. Security of three-move blind signature schemes reconsidered. In *SCIS'03, Symposium on Cryptography and Information Security, Japan*, 2003.
30. M. Pirker and D. Slamanig. A Framework for Privacy-Preserving Mobile Payment on Security Enhanced ARM TrustZone Platforms. In G. Min, Y. Wu, L. C. Liu, X. Jin, S. A. Jarvis, and A. Y. Al-Dubai, editors, *TrustCom*, pages 1155–1160. IEEE Computer Society, 2012.
31. W. Rankl and W. Effing. *Smart Cards in Transportation Systems*, pages 869–891. John Wiley & Sons, Ltd, 2010.
32. 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.
33. A.-R. Sadeghi, I. Visconti, and C. Wachsmann. User Privacy in Transport Systems Based on RFID E-Tickets. In C. Bettini, S. Jajodia, P. Samarati, and X. S. Wang, editors, *PiLBA*, volume 397 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
34. C.-P. Schnorr. Efficient Identification and Signatures for Smart Cards. In G. Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.
35. United Nations New York. *World Urbanization Prospects - The 2011 Revision* , 2012.
36. 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, 2011.