Cryptographic Theory Meets Practice: Efficient and Privacy-Preserving Payments for Public Transport

Andy Rupp[1], Foteini Baldimtsi[2], and Gesine Hinterwälder[3] Christof Paar[4]

[1] Karlsruhe Institute of Technology, Germany <andy.rupp@rub.de>
[2] Brown University, USA <foteini@cs.brown.edu>
[3] University of Massachusetts Amherst, USA <hinterwalder@ecs.umass.edu>
[4] Ruhr-University Bochum, Germany <christof.paar@rub.de>

Abstract. We propose a new lightweight cryptographic payment scheme for transit systems, called P4R (Privacy-Preserving Pre-Payments with Refunds), which is suitable for low-cost user devices with limited capabilities for performing public-key operations. In the P4R system users deposit money to obtain a bundle of one-show credentials, where each credential allows to make an *arbitrary* ride in the system. The actual fare of a trip is determined on-the-fly at the end. If the deposit for the credential exceeds this fare, the user obtains a refund. The system allows to aggregate a user's refund values collected over several trips in a single token thereby saving memory and increasing privacy.

Our solution builds on Brands' e-cash scheme to realize the pre-payment system and on Boneh-Lynn-Shacham (BLS) signatures to implement the refund capabilities. Compared to a Brands-only solution as transportation payment system, P4R allows to minimize the number of coins a user needs to pay for his rides and, thus, it minimizes the number of expensive withdrawal transactions as well as the storage requirements for the fairly large coins. Moreover, P4R enables flexible pricing as it allows for exact payments of arbitrary amounts (within a certain range) using a *single* fast spending (and refund) transaction. Fortunately, the mechanisms enabling these features only induce very little computational overhead.

Choosing contemporary security parameters, we implemented P4R on a prototyping payment device, and show its suitability for future transit payment systems. Estimation results demonstrate that the data required for 20 rides consumes less than 10 KB of memory and the payment and refund transactions during a ride take less than half a second.

As for security, we can show that a group of malicious users is not able to cheat the system by receiving a total refund, which exceed the overall deposit minus the overall fare, and without being identified during the double-spending checks. At the same time, the system protects the privacy of honest users in the sense that transactions are anonymous (except for deposits) and trips are unlinkable.

Keywords. E-cash, refunds, privacy, lightweight payments, transit system.

1 Introduction

Deploying electronic payment systems in transportation as opposed to sticking with traditional systems (like cash or ticket-based systems) offers important benefits: significantly reduced revenue collection costs, enhanced customer satisfaction as well as improved services and operational efficiency. Moreover, such systems enable some advanced services and features like dynamic pricing (i.e., reduced fares in overcrowded subways). Hence, Electronic Payment Systems for Transportation (EPST) already are and will become an even more important component of the critical infrastructure "transportation" that deserves careful protection from a wide range of adversaries.

Projects like the Massachusetts Bay Transportation Authority (MBTA) "Charlie Card" [30] or the E-ZPass [19] show the potential of electronic payment systems as a reasonable, fair,

and efficient method for revenue collection. However, at the same time they are examples demonstrating the serious shortcomings of today's EPST. Currently deployed systems lack sufficient mechanisms protecting their security and especially the privacy of their users: One problem that EPST seem to share with many other commercial systems implementing security functions is the deployment of cryptographically weak proprietary primitives as, e.g., demonstrated for the Charlie Card or Oyster Card [35]. Besides security issues, concerns about the location privacy of EPST users are raised frequently, i.e., the un-/traceability of users within the transportation system. For instance, the fact that E-ZPass and Fast Lane toll plaza records have been used by lawyers to prove that their clients' cheating husbands were not where they pretended to be at a certain date and time [46], shows (i) that this system does not protect location privacy and (ii) this might be exploited in a questionable way. However, in order to enable a large-scale deployment and broad acceptance of an EPST, adequate security and privacy mechanisms are an essential requirement [40].

While currently deployed EPST suffer from serious privacy and security flaws, there is a wealth of cryptographic payment schemes offering strong security and privacy properties. However, the unique requirements of the transportation domain, especially, engineering constraints and functional requirements, prevent the use of well-established cryptographic protocols like e-cash schemes.

In this paper we restrict to the consideration of a *transit* payment scenario such as payment systems for subways. Here, payment devices can be fairly low-cost platforms such as RFID transponders, contactless or hybrid smart cards, which are provided by the Transportation Authority (TA). Given such a device, a user can charge it at a vending machine to pay for rides in the subway system. The entry and exit points are typically physically secured by turnstiles, which are equipped with readers. These readers are responsible for fare calculation and conducting payment transactions with user devices. To avoid congestion in front of turnstiles, transactions have to be *fast*: payments should be executable within a few hundred milliseconds. Transactions at the vending machine are less time-critical but should also not take longer than several seconds.

The resource constraints of the user devices together with the realtime requirements are one of the main obstacles preventing the application of e-cash. User devices are typically equipped with only a few tens of kilobytes of memory and an 8- or 16-bit microcontroller running at not more than 16 MHz. On a widely used microcontroller, a modular exponentiation in 1024-bit RSA requires about 5 s at 16 MHz, while a point multiplication on a 160-bit elliptic curve takes around 400 ms [23]. Thus, it is clearly prohibitive to do much more than a single full public-key operation on such a device during a payment transaction. However, almost all e-cash schemes make excessive use of exponentiations or ECC point multiplications in the spending protocol. Fortunately, by employing an ECC coprocessor as accelerator, the runtime of a point multiplication can be improved by roughly one order of magnitude, (e.g., a factor 12 for the coprocessor in [37]). Note that, due to power constraints we may only assume the usage of such a coprocessor when the payment device is in contact mode, e.g., when interacting with the vending machine, which fits our transit scenario.

1.1 Related Work

E-Cash. An e-cash scheme typically consists of a bank, users, merchants and the following protocols: (1) a withdrawal protocol where a user obtains e-coins from the bank; (2) a spending

protocol where the user sends coins to a merchant; (3) a deposit protocol where a merchant deposits coins obtained from a user to his bank account; and (4) other protocols for identifying malicious behavior. In his seminal paper [15] Chaum introduced anonymous electronic cash (e-cash) that allows anonymous and unlinkable payments, while at the same time it ensures unforgeability of e-coins. Since then, e-cash protocols have been extensively studied, e.g., see [4,8,10,11,12,13], to cite only a few. We restrict to consider two of the most important state-of-the-art schemes.

Brands [8] proposed one of the first and most famous offline anonymous e-cash schemes with the most efficient spending protocol (only two modular *multiplications* per coin are required). However, Brands' coins occupy a fairly large amount of memory and coin withdrawal is relatively expensive: For a wallet containing N coins, one needs to store $5N$ elements from a group G as well as $2N$ elements from $\mathbb{Z}_{|G|}$. Moreover, the withdrawal requires about $12N$ exponentiations on the user's side.

Camenisch et al. [10] proposed so-called compact e-cash in which storage grows logarithmically with N, rather than linearly as above, but at the cost of a less efficient spending protocol (18 exponentiations per coin). Although, N coins can be withdrawn in one cheap transaction, they still need to be spent one-by-one.

Payment Systems Dedicated to Transportation. Heydt-Benjamin et al. [24] were the first to propose an informal cryptographic framework for transit payment systems. Sadeghi et al. [41] present an RFID-based e-ticket scheme for transit applications. Their system does not expect the user's payment device to carry out too many expensive computations, however, the existence of external trusted devices, so called anonymizers (such as a user's cell phone), is assumed for the costly operations and their system only protects a user's privacy with respect to outsiders and not the transportation authority. Blass et al. [5] proposed another offline "privacy-preserving" payment system for transit applications that solely relies on a 128-bit hash function and lots of precomputed data on the back-end's side. Again, a user's privacy in their system is not protected from the TA. A payment system for public transportation that protects user privacy even against the TA was recently proposed by Kerschbaum et al. [28]. However, their scheme realizes a billing system which works quite differently from the more traditional pay-upfront ticketing systems that we study and moreover, induces a high computational overhead.

In the last few years, a whole series of cryptographic systems for privacy-preserving toll collection have been proposed, e.g., [38], [1], [32], [17], a scenario which is related to but significantly different from transit payments which is the focus of this paper.

1.2 Our Approach and Contribution

Due to their strong security and privacy guarantees, it would be highly desirable to build a transit payment system based on e-cash. A good candidate for this purpose is Brands' scheme because of its exceptionally efficient spending protocol. On the downside, Brands' coins are large and their withdrawal is expensive. Hence, it would be ideal having to spend only a single coin per trip. However, this conflicts with the necessity of allowing flexible and dynamic prices, i.e., fares should not be flat but arbitrary monetary amounts: Setting the denomination of a coin to be 1 cent certainly allows for flexible pricing, but users would need plenty coins to pay for a trip. Setting the face value to $2 reduces the number of required coins per trip, but severely restricts the system of fares. To do a tradeoff by using different

denominations, one would need to deal with overpayments and change in a privacy-preserving way. This is especially difficult in EPST where bank and merchants are the same entity.

Our proposal of a transit payment system addresses the issues discussed above. The idea is to "let a single coin be worth exactly the (variable) cost of an arbitrary trip in the system". Our payment system is not a typical e-cash scheme but follows the concept of pre-payments with refunds: a user has to make a deposit to get a coin worth an arbitrary ride and gets a refund, if the actual fare is less than his deposit. We build our pre-payment system using Brands' scheme. However, our approach also works for other schemes like [2], where coins can be made 2-showable without revealing the ID of a user. The refund system is realized by using BLS signatures in a novel way that allows to aggregate refunds.

The way we employ BLS signatures to construct our refund system may be best described by the term "history-free sequential aggregate blind signature". While history-free sequential aggregate signatures based on BLS [21] as well as blind signatures based on BLS [6] are known, our requirements and techniques are quite different. In an aggregate signature scheme, multiple signatures from different senders may be combined into one short signature. History-free sequential aggregation means that aggregation is done sequentially by the signers and each signer derives the next aggregate considering only the message to be signed, the signature key, and the previous aggregate (but not the previous messages and public keys) [21]. In contrast to this, in our setting, we have a single signer and moreover, part of the (actual) message to be signed as well as the previous aggregates need to be hidden from this signer. Furthermore, the verification of the aggregate signature is done given an aggregate of the messages as opposed to the full message vector. Also, we do not make use of hash chains as in [21] to realize these capabilities. Finally, as opposed to [6], blinding is done in the exponent (rather than by multiplying a random group element) which offers certain benefits such as avoiding a costly unblinding operation after every aggregation.

More precisely, in our setting we securely combine signatures for messages of the form (s, w_i), where s is hidden and w_i is public, into a signature for the message $(s, \sum w_i)$. To this end, the signer is only given w_i and the blinded previous aggregate a_{i-1}, where a_0 is a blinded version of s. The next aggregate value is then computed as $a_{i-1}^{d^{w_i}}$ using the BLS signature key d. Blinding an aggregate is done by raising it to some random exponent. Instead of removing the blinding factor after every aggregation, blindings are also aggregated (multiplicatively) to save computations. Security for our construction essentially means that a signature for $(s, \sum w_i)$ can only be constructed by sequentially querying the signer. While we believe that this new technique can also be beneficial in other contexts, we restrict to prove security in the context of our refund system construction (cf. Lemma 3).

In Section 3, we introduce a general framework and security/privacy models for pre-payment with refund schemes. Section 4 describes our realization of such a system. As for security, we show in Section 5 at a semi-formal level that it is infeasible for malicious users to abuse the system. This includes users who try to dodge the fare or receive higher refunds than they are eligible to. Such users will be identified by double-spending checks. Regarding privacy, we show in Section 6 that the transportation authority cannot identify a user behind a sequence of transactions (including trips, collecting and redeeming refunds, etc.) or decide whether a sequence of trips involves a single or multiple users. For our proofs, we need a generalization of the Incremental Diffie-Hellman assumption [20], besides requiring

that Schnorr's identification scheme is secure and Brands' scheme is secure and (2-show) unlinkable.

Furthermore, we implemented P4R using 160-bit elliptic curves on the Moo RFID tag [47] housing a 16-bit MSP430 microcontroller. The results presented in Section 7 show that the scheme is fairly efficient even on this, not for our purposes optimized, device. Assuming a clock rate of 16 MHz for a fielded version of the device, the computations required for entering the system can be executed in 2 ms, exiting takes 360 ms, and redeeming the refund token 340 ms. Withdrawal (5.22 s) is also not far from meeting real-world requirements and could strongly be improved by making use of dedicated hardware (e.g., an ECC coprocessor in contact mode).

Finally, our "pre-payment with refunds" approach leaves space for many design options and allows to make interesting tradeoffs between security, privacy, and efficiency. Some of these tradeoffs are discussed in Section 8.

1.3 High-Level System Description

P4R is composed of three subsystems: the Trip Authorization Token (TAT) aka ticket system, the Refund Calculation Token (RCT) aka stamped ticket system, and the Refund Token (RT) aka piggy bank system. The TAT system is offline. Here vending machines play the role of the "bank" issuing TATs and (offline) readers at the entry turnstiles play the role of a "merchant" where tokens can be spent. The RT system is online. Here roles are reversed compared to the TAT system, i.e., readers at the exit turnstiles issue refunds and the (online) vending machines redeem these tokens.

A TAT (aka ticket) is a credential that authorizes a user to make a single trip. A user initially makes a deposit to obtain a number of TATs where the cost of a TAT equals the price of the most expensive trip in the system. Of course, to reduce this deposit it is also possible to have different types of TATs for different zones of the transportation system. The withdrawal of a TAT is done in a blind fashion such that a later use cannot be linked. The ID of a user is encoded in each TAT to prevent a repeated use for entering the system (double-spending detection). At the beginning of a ride, a user presents an unused TAT to the reader at the entry turnstile. If it is valid and the user can show (using a zero-knowledge proof) that he knows the ID encoded in the TAT, access is granted.

When leaving the system, the actual fare is determined at the exit turnstile. This is done as follows: on entering, the user also received an RCT (aka stamped ticket), which contains a MAC on the TAT, the date/time, and the ID of the reader. When he leaves, he presents this token to the exit turnstile which calculates the trip cost based on this information. He also provides a blinded version of his RT (aka piggy bank), where blank RT tokens are available from the vending machines, to which the reader adds the difference between the deposit for the TAT and the actual fare. To prevent the re-use of an RCT and thus claiming the same refund several times, the idea is to bind an RCT to the TAT which has been used on entering, and force a user to again prove the knowledge of the ID encoded into this TAT when he leaves. Note that an RT is used to add up several refunds instead of having a separate RT per refund. When a user decides to cash the collected refund, he presents his RT to the vending machine, which redeems it only if is not already marked as cashed.

2 Preliminaries

2.1 Negligible and Overwhelming Functions

Definition 1. *A function $\nu(k) : \mathbb{N} \mapsto [0,1]$ is called* negligible *if for every polynomial P, there exists some $c \geq 1$ such that for all $k > c$, $\nu(k) < |1/P(k)|$. We call a function $f : \mathbb{N} \mapsto [0,1]$* overwhelming *if $f(k) \geq 1 - \nu(k)$, where ν is negligible.*

Definition 1 can be easily generalized to functions in multiple security parameters.

2.2 Required Complexity Assumptions

Let us briefly review some complexity assumptions required for the security of the components of our scheme.

Definition 2 (Discrete Logarithm Assumption). *Let $k \in \mathbb{N}$ be a security parameter. Let G be a cyclic group of order q (k-bit prime) with generator g. Then, for every polynomial time algorithm $\mathcal{A}$, it holds that*

$$\mathsf{Pr}[h \leftarrow G; x \leftarrow \mathcal{A}(1^k, G, q, g, h) : x = \log_g h] \leq \nu(k)$$

where $\nu(k)$ is a negligible function.

The DL assumption is required for the security of Schnorr's identification scheme as well as Brands' e-cash scheme, both being building blocks of our construction.

For proving the security of our refund system, we additionally need a new assumption which we call the n-Σ-incremental DH assumption. This new number-theoretic assumption generalizes the Incremental Diffie-Hellman (IDH) assumption used to prove a loyalty scheme to be secure in [20]. Roughly, the IDH problem asks to compute $x^{d^v} \in G$ for a given pair $(x, h^d) \in G^2$ and with access to an exponentiation oracle $(\cdot)^d$, which can be queried less than v times. Currently, the best method to solve this problem is to compute the discrete logarithm d.

Definition 3 (n-Σ-Incremental DH-Assumption). *Let $k \in \mathbb{N}$ be a security parameter. Let $n \in \mathbb{N}$ and $(G, h, p, d, (h^{d^i})_{i\in[n]}) \leftarrow \mathsf{Gen}(1^k)$ be given, where G is a group of order p (k-bit prime) with generator h and $d \leftarrow \mathbb{Z}_p^*$. Furthermore, let $\mathcal{O}_G$ be a challenge oracle, which when queried returns a random $x_i \leftarrow G$, and $\mathcal{O}_d$ be an oracle which raises a given element to the d-th power. Consider the output $(c_1, v_1, \ldots, c_m, v_m) \leftarrow \mathcal{A}^{\mathcal{O}_d, \mathcal{O}_G}(1^k, G, p, h, (h^{d^i})_{i\in[n]})$ of a polynomial-time algorithm $\mathcal{A}$. Then the probability that $c_i = x_i^{d^{v_i}}$ with $v_i < p - 1$ for all $i \in [m]$, where $x_1, \ldots, x_m$ are the challenges generated by $\mathcal{O}_G$ in the run, and $\mathcal{A}$ has made less than $v_1 + \cdots + v_m$ oracle queries to $\mathcal{O}_d$, is negligible.*

Note that the DL assumption over G is implied by the n-Σ-IDH assumption over G. For small n and small exponents v_i (i.e., of size polynomially bounded in k) we can prove the hardness of the n-Σ-Incremental DH problem in the generic group model (GGM) [44] with a pairing. Moreover, the assumption can also be reduced to a variant of the DL assumption over the target group of the pairing in the semi-generic group model (SGGM) [27]. Please refer to Appendix B for a proof in the SGGM. We would like to note that this proof also implies hardness in the GGM.

2.3 Pairings and the BLS Signature Scheme

Let G, G_T be groups of prime order p. We will use multiplicative notation for both groups. A mapping $e : G \times G \to G_T$ is called a *cryptographic bilinear map* or *pairing* if it satisfies the following properties:

- *Non-degeneracy:* if g is a generator of G, then $e(g, g) \neq 1$.
- *Bilinearity:* $e(a^x, b^y) = e(a, b)^{xy}$ for any $a, b \in G$ and $x, y \in \mathbb{Z}_p$.
- *Computability:* there exists an efficient algorithm to compute $e(a, b)$ for any $a, b \in G$.

The signature scheme due to Boneh, Lynn and Shacham (BLS) [7] is secure against existential forgery under a chosen message attack (in the random oracle model) assuming that the Computational Diffie-Hellman problem is hard. Let G, G_T, g, e be given as described above. BLS consists of the following algorithms:

- *KeyGen* picks a random $d \leftarrow \mathbb{Z}_p^*$. The public key is $h = g^d$ and the private key is d.
- *Sign* signs a message $m \in G$ by simply computing $\sigma = m^d$.
- *Verify* verifies a message-signature pair (m, σ) by checking whether the equation $e(g, \sigma) = e(h, m)$ holds.

2.4 Zero Knowledge Proofs of Knowledge and Schnorr's Protocol

Our construction makes use of zero-knowledge proofs of knowledge (ZKPoK). Informally, a ZKPoK is a two-party protocol between a prover P and a probabilistic polynomial-time verifier V, where P convinces V that he knows a witness d to some "hard" relation (e.g., d is the discrete logarithm of h with respect to some generator g) without the verifier learning anything beyond the fact. In particular, a ZKPoK protocol satisfies the following properties (cf. [22] for formal definitions):

- *Completeness:* For honest V, P the protocol succeeds with overwhelming probability.
- *Soundness:* There exists some (expected poly-time) algorithm K (called knowledge extractor), which (by rewinding) extracts a witness from every (potentially dishonest) prover P^* succeeding in the protocol with non-negligible probability.
- *Zero-Knowledge:* For each (potentially dishonest) polynomial-time verifier V^*, there exists a polynomial-time algorithm S (called simulator), which, without knowing a witness, is able to generate a transcript that is indistinguishable from a real transcript of a proof between V^* and P.

In particular, we make use of Schnorr's protocol for proving knowledge of a discrete logarithm [42]. Let g be a generator of some group G of prime order p and let $h = g^d \in G$. To prove knowledge of the discrete logarithm d of h, P picks $r \leftarrow \mathbb{Z}_p$ and sends $a = g^r$ to V. The verifier then chooses a challenge element $c \leftarrow \mathbb{Z}_p$ and sends it to the prover. Finally, P computes the response $z = r + cd$ and sends it to V who checks whether the equation $g^z = ah^c$ is satisfied. Note that Schnorr's protocol is an honest-verifier zero-knowledge protocol. The proof of knowledge of a representation of an element $h = g_1^{d_1} g_2^{d_2}$ with respect to generators g_1 and g_2 used in Brands' scheme is a generalization of Schnorr's protocol: Here, P sends $a = g_1^{r_1} g_2^{r_2}$ and $z_1 = r_1 + cd_1$, $z_2 = r_2 + cd_2$ to V, who verifies $g_1^{z_1} g_2^{z_2} = ah^c$.

2.5 Brands' E-Cash System

In the following we sketch Brands' e-cash scheme [9]. During setup, the bank $\mathcal{B}$ chooses a cyclic group G of prime order q, generators (g, g_1, g_2), a number $x \in \mathbb{Z}_q$ and a collision-resistant hash function $H : G^5 \to \mathbb{Z}_q$. The bank's public key is $(g, g_1, g_2, g^x, g_1^x, g_2^x, H)$, and its secret key is x. A user's secret key in this scheme is some element $id \in \mathbb{Z}_q$ and his public key is g_1^{id}. The user's secret key is encoded in each coin the user withdraws. More precisely, a coin is a tuple $(A, B, \mathsf{sig}(A, B))$, where $A = g_1^{id \cdot s} g_2^s$, $B = g_1^{x_1} g_2^{x_2}$, and $\mathsf{sig}(A, B)$ is a signature on A and B. The value s, which is a random element from $\mathbb{Z}_q$, is the serial number of a coin. B is a commitment to random values $x_1, x_2 \in \mathbb{Z}_q$, which are later used during the spending protocol. Note that the values s, x_1, x_2 are only known to the user who withdraws the coin. Also, the bank does not know A, B, or $\mathsf{sig}(A, B)$ after an execution of the withdrawal protocol. The details of the withdrawal protocol are shown in Figure 1.

$\mathcal{U}$		$\mathcal{B}$
Input: $id, g, g_1, g_2, g^x, g_1^x, g_2^x, H$		Input: g, g_1, g_2, g_1^{id}
	$\xleftarrow{a, b}$	$w \leftarrow \mathbb{Z}_q$, $a = g^w$, $b = (g_1^{id} g_2)^w$
$s \leftarrow \mathbb{Z}_q$		
$A = (g_1^{id} g_2)^s$		
$y = (g_1^x)^{id} g_2^x$, $y' = y^s$		
$x_1, x_2, u, v \leftarrow \mathbb{Z}_q$		
$B = g_1^{x_1} g_2^{x_2}$		
$a' = a^u g^v$, $b' = b^{su} A^v$		
$c' = H(A, B, y', a', b')$		
$c = c' u^{-1} \bmod q$	$\xrightarrow{c}$	
$g^z \stackrel{?}{=} (g^x)^c a$	$\xleftarrow{z}$	$z = w + cx \bmod q$
$(g_1^{id} g_2)^z \stackrel{?}{=} y^c b$		
$z' = v + zu \bmod q$		

Fig. 1. Brands' e-cash: withdrawal of a coin $(A, B, \mathsf{sig}(A, B))$

In the spending protocol, the user presents $A, B, \mathsf{sig}(A, B)$ to the merchant and proves ownership of the coin by showing that he knows a representation of A with respect to g_1 and g_2 (i.e., id and s). To hide id and s, the random numbers x_1 and x_2, the user had committed to during the withdrawal, are used. If the user tries to double-spend a coin and is thus forced to re-use these values, id and s can easily be revealed.

Unfortunately, the security of Brands' scheme has never been fully formalized and proven rigorously. Nevertheless, to the best of our knowledge, up to date, i.e., during the last 20 years, the security properties claimed about Brands' scheme have not been refuted, although the scheme has been very popular. (The attack claimed in [14] could be easily fixed and seems to be caused by a lapse in [9]). In fact, even recently, Microsoft built its digital credential identity management system, U-Prove [36], based on the unproven blind signatures which are at the heart of Brands' scheme. The security of our construction relies on the following properties of Brands' e-cash:

– *Restrictiveness:* Briefly, the idea of a *restrictive* blind signature is to allow a user to get a signature on a (blinded) message, not known to the signer, only if the message satisfies a certain property. In Brands' withdrawal protocol, this property is basically that the secret key of the user must be embedded in the message.

More precisely, Brands defines restrictiveness in the following way (cf. Definition 5 in [9]): Let $m \in G$ ($m = g_1^{id} g_2$) be a message such that the receiver knows a representation (a_1, a_2) of m with respect to a generator-tuple (g_1, g_2) at the start of a blind signature protocol. Let (b_1, b_2) be the representation the receiver knows of the blinded number m' ($m' = g_1^{ids} g_2^s$) of m after the protocol has finished. If there exist two (blinding-invariant) functions I_1 and I_2 such that $I_1(a_1, a_2) = I_2(b_1, b_2)$, regardless of m and the blinding transformations applied by the receiver, then the protocol is called a restrictive blind signature protocol. Assumption 1 in [9] states that Brands' withdrawal protocol is a restrictive blind signature with $I_1(a_1, a_2) = a_1/a_2$ and $I_2(b_1, b_2) = b_1/b_2$.

- *Unforgeability:* Informally speaking, an e-cash system is called (one-more) unforgeable if it is infeasible for an adversary who engaged in ℓ successful executions of the withdrawal protocol to output $\ell + 1$ different and valid coins. Proposition 7 in [9] states that Brands' scheme is unforgeable under the assumption that Schnorr signatures are unforgeable. Note that one-more unforgeability was never proven rigorously for Brands' e-coins. In fact, there exists an impossibility result on proving the unforgeability of Brands' blind signatures in the Random Oracle Model using the *standard* replay reductions [3]. Nevertheless, this does not mean that Brands' scheme is insecure; its one-more unforgeability remains an open problem.
- *Unlinkability:* Intuitively, unlinkability in an e-cash scheme means that an adversary cannot tell whether two different coins came from the same user if he has access to the views of the withdrawal and spending operations. More precisely, consider a polynomial-time adversary $\mathcal{A}$, who impersonates the bank and the merchant (where e-coins are spent). Let $\mathcal{U}_0$ and $\mathcal{U}_1$ be two honest users. The adversary $\mathcal{A}$ may first ask both users to withdraw and spend a couple of coins. Then, during the challenge phase, we flip a bit $b \leftarrow \{0, 1\}$ and set the left user $\mathcal{U}_L = \mathcal{U}_b$ and the right user $\mathcal{U}_R = \mathcal{U}_{1-b}$. Now $\mathcal{A}$ may ask $\mathcal{U}_L$ and $\mathcal{U}_R$ to spend some coins (where we assume that a sufficient number of coins have been withdrawn before). In the end, $\mathcal{A}$ outputs a bit b' and wins if $b' = b$. We call an e-cash scheme unlinkable if $\mathcal{A}$'s chance to win is at most $\frac{1}{2} + \nu(k)$ where ν is a negligible function. The unlinkability of Brands scheme is based on the following stronger information-theoretic property which essentially says that any spending may correspond to any user and any withdrawal:

  Proposition 1 (Proposition 11 in [9]). *For any User, for any possible view of the Bank in an execution of the withdrawal protocol in which User accepts, and for any possible view of Merchant in an execution of the spending protocol in which the payer followed the protocol, there is exactly one set of random choices that the User could have made in the execution of the withdrawal protocol such that the views of the Bank and the Merchant correspond to the withdrawal and spending of the same coin.*

- *2-Showable Coins:* For our construction, we will need an e-cash scheme that allows to show a coin twice, while different coins remain unlinkable. It is straightforward to extend the basic Brands' scheme to a 2-showable one by adding an extra value $C = g_1^{x'_1} g_2^{x'_2}$ similar to B, where $x'_1, x'_2 \leftarrow \mathbb{Z}_q$. Then a user may first prove knowledge of a representation of A using x_1, x_2 and a second time using x'_1, x'_2.
  The restrictiveness and unforgeability properties of Brands' scheme naturally extend to this 2-showable coin version. Concerning unlinkability, it is acceptable that two spendings of the same coin can be linked, while it should still be infeasible to link transactions

involving different coins. Proposition 1 can be easily extended and proven for 2-showable coins.

3 A Framework and Security Model for Transit Payments with Refunds

In a transit payment scheme with refunds we consider four different parties: the users $\mathcal{U}$, the Transportation Authority (TA) $\mathcal{T}$, the online vending machines $\mathcal{V}$, and the offline readers $\mathcal{R}$ (at the turnstiles). $\mathcal{V}$ and $\mathcal{R}$ are usually owned and fully controlled by $\mathcal{T}$. In the following we define such a system as a composition of three subsystems, the TAT, the RCT, and the RT subsystem comprising various algorithms and protocols which are executed by the above parties. For the sake of simplicity, we only present a high-level system interface at this point not listing every single input which may be required by the parties to execute the protocols.

TAT Subsystem. This subsystem includes algorithms and protocols allowing users to register to the system and use TAT tokens, specifically:

- $\mathsf{KGenTAT}(1^{k_1})$ is a probabilistic algorithm executed by $\mathcal{T}$ to setup the TAT subsystem. On input of 1^{k_1}, it generates $\mathcal{T}$'s public and private key $(pk^{\mathsf{TAT}}, sk^{\mathsf{TAT}})$.
- $\mathsf{Register}(\mathcal{U}(ID_{\mathcal{U}}, pk^{\mathsf{TAT}}), \mathcal{T}(\cdot))$ is an interactive protocol executed between a user with identity $ID_{\mathcal{U}}$ and $\mathcal{T}$. As output $\mathcal{U}$ obtains a public/private key pair $(pk^{\mathcal{U}}, sk^{\mathcal{U}})$ and the TA registers $\mathcal{U}$'s public key and identity information $ID_{\mathcal{U}}$ to a user database $\mathcal{DB}_{\mathcal{U}}$ (to be used to identify a user who double-spends a TAT).
- $\mathsf{BuyTAT}(\mathcal{U}(pk^{\mathsf{TAT}}, pk^{\mathcal{U}}, sk^{\mathcal{U}}), \mathcal{V}(sk^{\mathsf{TAT}}))$ is an interactive protocol between a user with public key $pk^{\mathcal{U}}$ and a vending machine who knows the secret key sk^{TAT}. If the protocol is successful, $\mathcal{U}$ obtains a TAT token TAT_i issued by $\mathcal{V}$, which "encodes" $\mathcal{U}$'s secret key in a blind fashion.
- $\mathsf{ShowTAT}(\mathcal{U}(\mathsf{TAT}_i, sk^{\mathcal{U}}), \mathcal{R}(pk^{\mathsf{TAT}}))$ is executed between a user and a reader to spend/show TAT_i. After a successful interaction, $\mathcal{R}$ stores TAT_i together with the transcript of the protocol in a TAT database $\mathcal{DB}_{\mathsf{TAT}}$.
- $\mathsf{IdentTAT}(\mathcal{DB}_{\mathsf{TAT}})$ is an algorithm executed by $\mathcal{T}$ to check if there exist two different transcripts for the same TAT_i in the TAT database. If this is the case, it outputs the identity $ID_{\mathcal{U}}$ of the cheating user along with his secret key as a proof.

RCT Subsystem. This subsystem comprises algorithms and protocols between users and readers required to handle refund calculation:

- $\mathsf{KGenRCT}(1^{k_2})$ is executed by $\mathcal{T}$ and returns a k_2-bit RCT key K.
- $\mathsf{GetRCT}(\mathcal{U}(\mathsf{TAT}_i), \mathcal{R}(id_{\mathcal{R}}, K))$ is an interactive protocol between a user and a reader that is executed immediately after $\mathsf{ShowTAT}$. The execution ends with the user obtaining a refund calculation token RCT_i for the TAT token TAT_i just spent.
- $\mathsf{ShowRCT}(\mathcal{U}(\mathsf{RCT}_i, sk^{\mathcal{U}}), \mathcal{R}(K, pk^{\mathsf{TAT}}))$ is a protocol executed by a user to enable the reader to calculate his refund when he exits the system. To this end, $\mathcal{R}$ checks the validity, freshness, and ownership of RCT_i sent by $\mathcal{U}$. Then it calculates and outputs the refund value w based on the information in RCT_i and stores the transcript of the protocol to an RCT database $\mathcal{DB}_{\mathsf{RCT}}$ together with RCT_i.
- $\mathsf{IdentRCT}(\mathcal{DB}_{\mathsf{RCT}})$ is executed by $\mathcal{T}$ to check if there exist two different transcripts for the same RCT_i in the RCT database. In that case it outputs the identity $ID_{\mathcal{U}}$ of the cheating user along with his secret key as a proof.

RT Subsystem. This subsystem, handling the refund collection and redeem process, comprises the following algorithms and protocols:

- $\mathsf{KGenRT}(1^{k_3})$ is a probabilistic algorithm executed by $\mathcal{T}$ that outputs a public and secret key pair $(pk^{\mathsf{RT}}, sk^{\mathsf{RT}})$.
- $\mathsf{GetRT}(\mathcal{U}(pk^{\mathcal{U}}, sk^{\mathcal{U}}), \mathcal{V}(\cdot))$ is an interactive protocol between a user and a vending machine to create a refund token RT for the user. $\mathcal{V}$ adds RT to a database $\mathcal{DB}_{\mathsf{RT}}$.
- $\mathsf{GetRefund}(\mathcal{U}(\mathsf{RT}), \mathcal{R}(w, sk^{\mathsf{RT}}))$ is executed immediately after $\mathsf{ShowRCT}$ between the user and the reader to add the calculated refund w for a single trip to the user's (blinded) RT RT.
- $\mathsf{RedeemRT}(\mathcal{U}(\mathsf{RT}, pk^{\mathcal{U}}, sk^{\mathcal{U}}), \mathcal{V}(pk^{\mathsf{RT}}))$ is an interactive protocol between $\mathcal{U}$ and $\mathcal{V}$ in order to redeem the user's refund token RT. If the token is valid, and not marked as "redeemed" in $\mathcal{DB}_{\mathsf{RT}}$, $\mathcal{U}$ receives the corresponding refund amount. After that, RT is marked as "redeemed" in $\mathcal{DB}_{\mathsf{RT}}$.

3.1 Informal Discussion of Security and Privacy Aspects

In the following, we will discuss which kind of adversaries and attacks are covered by our security and privacy models, such that readers who prefer to skip the formal models and proofs do not miss important details.

3.1.1 Security Let us take a quick look at the type of adversaries captured by our security definition given in Section 3.2.1. We consider adversaries who want to pay less than the real fare and, for this, they are in full control of a group of malicious colluding users and may also eavesdrop on honest users. Hence, the adversary may manipulate the messages (e.g., TAT, RCT, RT tokens, or other values) sent during protocol runs of malicious users but only perform passive attacks on honest users. In particular, the latter means that he may not jam or modify the (wireless) communication between honest users and authorities or perform relay attacks. Note that mounting such attacks on a large scale, however, seems to be hard.

Although, the adversary may act maliciously with respect to the protocol interactions of colluding users, all users are assumed to always execute the corresponding protocols in the right sequence. In particular, we do not consider physical attacks to evade the fare. For instance, a user may not leave the system by jumping over the exit turnstile, not executing $\mathsf{ShowRCT}$, and is thus still in possession of an unused RCT or can call GetRCT a second time at the entry turnstile. More generally, we assume that for every call of GetRCT a user does, he also has to execute $\mathsf{ShowRCT}$ and there are never two calls of GetRCT without an intermediate call of $\mathsf{ShowRCT}$. More illustratively, this means that when starting a ride, after the entry turnstile has sent an RCT, the user must always (be forced to) properly terminate this ride by leaving through an exit turnstile and presenting the received RCT there. This ensures that a user physically inside the transportation system is always in possession of exactly one unused RCT.

Moreover, colluding users are assumed not to share their secret IDs, i.e., $sk^{\mathcal{U}}$. Otherwise, two users in our scheme could exchange their RCTs while traveling and obtain higher refunds than they are eligible to.[5] While this might seem like a strong assumption, note that in our

[5] Note that we do not run into this problem in case of a single user registering with two different IDs. Our restriction with respect to the protocol sequence ensures that when this user is inside the transportation system, he is in possession of only one unused RCT (issued using one or the other of his IDs).

system, sharing ID information also entails a high risk for a user: Others might spend a user's RCT twice and thus the user's ID is revealed to the TA or the ID is used to buy TATs in the user's name which might then be misused. So there is also a strong incentive for users to behave honest in this way. Users may also be forced not to share by using tamper resistant payment devices making it very hard to extract $sk^{\mathcal{U}}$ from the device.

Although, our adversarial model is restricted in some ways as described above, it still covers a broad class of attacks. In particular, adversaries may try to forge, double-spend, and use eavesdropped TATs, RCTs and RTs or manipulate other protocol values in order to ride for free or receive higher refunds than they are eligible to.

3.1.2 Privacy Let us now discuss what type of adversaries and attacks are covered by our privacy model given in Section 3.2.2. In this model the adversary may act as the TA with respect to honest users. Additionally, he may try to learn from training phases where users are not anonymous in interactions and receive orders from the adversary. Essentially, the adversary's goal is to identify the user behind a sequence of certain events (including trips, the redemption of an RT, etc.) or to decide whether a sequence of events involves one and the same or multiple users. In particular, our model guarantees that except for user registration and buying TATs, any other event, including rides (i.e., showing TATs, obtaining and showing RCTs, collecting refunds) as well as obtaining and redeeming RTs, is perfectly anonymous. That means, for a given user identity, any sequence of trips is equally likely (with respect to an adversary only considering protocol transcripts). Furthermore, even if the user ID could be linked with the redemption of an RT token (by external means) and thus the total refund amount is leaked for this user, our model still ensures that the user's sequence of trips cannot be distinguished from any other one leading to the same refund amount. This also includes sequences, where the individual trips involve different users.

Note that we consider adversaries who only make use of the information provided by protocol runs, but not any kind of background knowledge (like user profiles, the likelihood of certain trip combinations, etc.). We additionally need to make the restriction that the adversary, when acting as the TA, honestly follows the GetRefund protocol. This is needed as, for efficiency reasons, users (owning constrained devices) in our system will not verify the amount added to their RT. Thus, by adding some bogus refund to the RT of one out of two users, instead of the claimed amount, or by using separate signature keys for the two users, the adversary could distinguish them. As a practical countermeasure, however, one could introduce spot-checks where users with more powerful devices like smartphones actually verify issued refunds.

In the following, we will argue that in theory (again ignoring any background knowledge and side information) the refund amount revealed by RedeemRT should not be of much help in restricting the set of possible sequences of trips an (anonymous) user redeeming a refund of v cent could have taken. This set, now containing all combinations of trips (in the records) leading to a refund of v cents, should usually be pretty large for the following reasons: Note that the number of decompositions of v into single-trip refunds equals the number of *integer partitions* of v. Hence, we are interested in the number $p_S(v)$ of partitions of integers v where parts are restricted to come from a certain set S (the set of single-trip refunds). Unfortunately, there are no explicit formulas for this number for arbitrary S, but only for S satisfying certain

conditions. For instance, in the case $S = \{w_1, \ldots, w_k\}$ with $w_i < w_{i+1}$ and $\gcd(S) = 1$, Schur shows [43] that $p_S(v) \approx v^{k-1}((k-1)!\, w_1 \cdots w_k)^{-1}$, where $p_S(v) > 0$ for v large enough.

Fortunately, in our application scenario the parameters w_i and k are usually pretty small and should be publicly known. Hence, we can compute $p_S(v)$ in a naive way and do not rely on such explicit formulas.

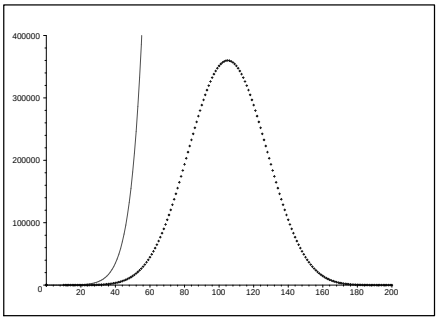

Fig. 2. Number of partitions (solid line) and 10-partitions (dots) of integers in the range $[0, 200]$ with parts from the range $[1, 20]$

For instance, let us consider a toy example of a distance-based EPST, where the refund unit is dimes, a user has to pay one dime per traveled station, and the longest trip consists of 20 stations which translates to a deposit of 20 dimes for a TAT. Thus, we are interested in the number $p_S(v)$ of combinations of single-trip refund values $w_i \in S := \{1, \ldots, 20\}$ leading to a fixed sum v. The graph on the left-hand side (solid line) of Figure 2 depicts these numbers (in the range $0 \leq v \leq 200$). One can see that $p_S(v)$ grows super-polynomially for these parameters and we exceed a number of 1000 partitions already for $v = 22$. Moreover, the graph on the right-hand side (dots) of Figure 2 shows the numbers $p_S(v, 10)$ of possible partitions consisting of 10 parts. This reflects the situation, where the size of the partition would also be known, which is the case for a variation of our basic system described in Section 8. Clearly, $p_S(v, 10)$ is a Gaussian function and a number of 1000 partitions is exceeded between $v = 34$ and $v = 176$. By means of this toy example, we would like to demonstrate that already for a small total refund amount, the number of possible trip sequences can be pretty large which, however, depends on the parameters.

Now, note that for each particular partition $v = w_{i_1} + \cdots + w_{i_m}$ there typically should not only be a single, but numerous sequences of trips in the records satisfying this partition. Thus, in the extreme (but pointless) case where all trips lead to the same refund w, there is only a single partition for each particular v, but there naturally should be plenty of different sequences leading to v.

Finally, we would like to stress that the actual level of location privacy of a user in a real implementation will certainly depend on numerous factors including the background knowledge of an adversary, his actual goals (tracing users, meeting disclosure, etc.), user behavior (e.g., when are RTs redeemed), transportation system characteristics (e.g., popularity of particular trips), etc. For instance, in rare cases it could happen that since the issue date of a refund token nobody but the owner of this token did trips that could have resulted in a particular refund amount v (though, in theory many sequences are possible). Also, the price system might be set up in a way that allows to decide whether a user redeeming an RT has taken one particular trip or not (e.g., if all trips lead to refunds of 50 cents and only one particular trip leads to a refund of 51 cents). Nevertheless, we believe that for real-world transportation systems this is not a serious issue. In fact, the TA should publish the set of possible trips and corresponding refund values which would allow to (partly) check the

existence of such a problem. An analysis of the actual degree of location privacy provided by P4R, taking all these factors into account, would certainly be a very interesting contribution on its own but is out of scope of this paper.

3.2 Security and Privacy Model

We define the desired properties by means of experiments using a probabilistic polynomial time adversary $\mathcal{A}$. The considered adversary is in full control of a (dynamically growing) set of malicious colluding users and may also eavesdrop on honest users. To this end, $\mathcal{A}$ can adaptively query a set of interactive oracles that are defined below and basically mirror the system interface. The protocols ShowTAT/GetRCT and ShowRCT/GetRefund that are executed consecutively are combined. We assume that the system parameters and keys have already been determined and omit them from the oracle descriptions below for simplicity. Of course, $\mathcal{A}$ is assumed to receive all the messages exchanged in the scope of the executed protocols. In addition, $\mathcal{A}$ may manipulate the messages (e.g., TAT, RCT, RT tokens, or other values) sent during protocol runs of malicious users (but only act as passive adversary with respect to honest users).

- Register$^*(ID_{\mathcal{U}})$ is used to register a user with identity $ID_{\mathcal{U}}$ by running protocol Register. We assume that the user's secret key $sk^{\mathcal{U}}$, generated in the scope of the registration, is randomly chosen, globally unique, verifiably linked to the user, and unknown to the adversary.
- Corrupt$(ID_{\mathcal{U}})$ is used by the adversary to corrupt some previously honest registered user $ID_{\mathcal{U}}$. The adversary receives all internal data of the party except for its secret key $sk^{\mathcal{U}}$ and, from then on, the adversary acts on behalf of this party. In particular, the adversary may try to use forged data or data from a different user in interactions involving this corrupt user. Note that for honest users the adversary is not able to control which data is used during interactions, but may only decide which protocols the user executes (provided the order of executions does not violate our policy below). We collect identities of corrupt users in an initially empty set $\mathcal{C}$.
- BuyTAT$^*(ID_{\mathcal{U}})$ can be used to make an honest or corrupt user $ID_{\mathcal{U}}$ interact with a vending machine to retrieve a TAT (running BuyTAT).
- GetRT$^*(ID_{\mathcal{U}})$ makes a user interact with a vending machine to obtain a new RT.
- Enter$(ID_{\mathcal{U}}, id_{\mathcal{R}}, ts)$ makes the party $ID_{\mathcal{U}}$ interact with the reader $id_{\mathcal{R}}$ at time ts to show a TAT (executing ShowTAT) and to get an RCT afterwards (executing GetRCT). If both protocol executions are successful, Enter outputs "granted", otherwise it returns "denied". Before Enter can be executed again on behalf of the same user, Exit must be called first for this user and terminate with "granted". Honest users may not be asked to double-spend a TAT.
- Exit$(ID_{\mathcal{U}}, id'_{\mathcal{R}}, ts')$ makes the party $ID_{\mathcal{U}}$ interact with reader $id'_{\mathcal{R}}$ at time ts' to get a refund, running ShowRCT and GetRefund. If both protocol executions are successful, Exit outputs "granted", otherwise it returns "denied". Before Exit can be called (again after a successful execution) for a user, this user must have been involved in a successful execution of Enter. Honest users may not be asked to show an RCT twice (or an RCT other than the one they received before using Enter). The refund of a user is calculated from the presented RCT and denoted by w. The authentic trip costs denoted by u are determined

by $id'_{\mathcal{R}}, ts'$ and the data $id_{\mathcal{R}}, ts$ from the last call of Enter for this user. Note that in case the user is able to cheat here, w may be higher than the deposit for a TAT minus u.
– $\mathsf{RedeemRT}^*(ID_{\mathcal{U}})$ makes user $ID_{\mathcal{U}}$ redeem an RT at the vending machine. Honest users may not be asked to redeem an RT twice.

3.2.1 TA Security To put it simply, a transportation payment system with refunds should primarily guarantee that the TA does not lose any money. In other words, users should not be able to receive reimbursements which exceed the overall deposit for TATs minus the overall fare of their trips without being identified by the TA.

To model attacks, we define an adversary $\mathcal{A}$, who acts as a group of malicious colluding users, who also eavesdrop on honest users. The TA including vending machines and turnstiles are assumed not to be corrupted by this adversary and behave according to the specified protocols. The key material (in particular $sk^{\mathsf{TAT}}, sk^{\mathsf{RT}}, K$) of these entities is not known to $\mathcal{A}$. We capture the security notion stated above by means of following game-based definition using the oracles introduced in Section 3.2:

Definition 4 (TA Security). *We call a transportation payment scheme with refunds TA-secure if, for any PPT adversary $\mathcal{A}$ the probability of outputting 1 in the experiment below is negligible.*

Experiment TA Security. Let $(sk^{\mathsf{TAT}}, pk^{\mathsf{TAT}}) \leftarrow \mathsf{KGenTAT}(1^{k_1})$, $K \leftarrow \mathsf{KGenRCT}(1^{k_2})$, *and* $(sk^{\mathsf{RT}}, pk^{\mathsf{RT}}) \leftarrow \mathsf{KGenRT}(1^{k_3})$. *Furthermore, let* $pk = (pk^{\mathsf{TAT}}, pk^{\mathsf{RT}})$ *be the public parameters and keys. Then run* $\mathcal{A}^{\mathsf{Register}^*,\mathsf{Corrupt},\mathsf{BuyTAT}^*,\mathsf{GetRT}^*,\mathsf{Enter},\mathsf{Exit},\mathsf{RedeemRT}^*}(pk)$ *and let*

– $\ell\mathfrak{U}$ *be the total deposit for TATs made during executions of* BuyTAT^*,
– $\mathfrak{F}_1, \ldots, \mathfrak{F}_m$ *be the authentic trip costs incurred in interactions with* $\mathsf{Enter}/\mathsf{Exit}$,
– $\mathfrak{R}_1, \ldots, \mathfrak{R}_k$ *be the reimbursements paid by $\mathcal{V}$ in interactions with* $\mathsf{RedeemRT}^*$,

involving users eventually in $\mathcal{C}$. The experiment outputs 1 if $\sum_{i=1}^{k} \mathfrak{R}_i > \ell\mathfrak{U} - \sum_{j=1}^{m} \mathfrak{F}_j$ *or* $m > \ell$ *and when running the double-spending identification algorithms* $\mathsf{IdentTAT}$ *and* $\mathsf{IdentRCT}$, *$\mathcal{T}$ cannot output any identity $ID_{\mathcal{U}}$ and secret key $sk^{\mathcal{U}}$ belonging to one of the previously registered users in $\mathcal{C}$.*

Remark 1. Note that $sk^{\mathcal{U}}$ is randomly chosen and unknown to $\mathcal{T}$ when a user registers. By just guessing $sk^{\mathcal{U}}$ at the end of the experiment, $\mathcal{T}$ has a negligible chance to identify a double spender (or even a valid user). So the experiment is well-defined in this respect.

Furthermore, note that although the $\mathsf{Corrupt}$ oracle does not reveal the user's secret key to the adversary, it is still possible that $\mathcal{A}$ may try to spend a user's TAT (or RCT) twice to obtain $sk^{\mathcal{U}}$. With this knowledge, $\mathcal{A}$ can use the user's RCT with a different user and get a higher refund. However, we would like to stress that in this way also the TA detects this double-spending, so the adversary does not break security in the sense of Definition 4. In practice, the TA would also blacklist the ID of the cheating user such that no TATs can be purchased anymore encoding this ID.

It is not hard to see that the first inequality in Definition 4 essentially covers all attack scenarios a TA may want to be protected from as they result in losing money. The second inequality $m > \ell$ is only meant to prevent scenarios where the number of trips exceeds the number of purchased TATs which does not necessarily result in the TA losing money:

Consider an adversary who buys one TAT worth $\$k$, but is able to do k rides worth \$1. For some reason he does not (or is not able to) claim any refunds. Such an attack does not violate the first inequality, as the adversary did trips worth a total of $\$k$ for which he pre-paid $\$k$ and did not get any refund.

3.2.2 User Privacy Our model should cover adversaries who try to identify the user behind a sequence of protocol runs or try to link certain protocol runs.

A natural attempt to define such a model is to follow the definition of blindness in blind signature schemes. That is, consider an honest-but-curious adversary who impersonates the transportation authority (vending machines and readers) and runs executions with two honest users $\mathcal{U}_0$ and $\mathcal{U}_1$. The adversary first asks both users to buy TATs. Then, the two users are assigned as left and right user in random order according to a secret random bit b and the adversary may instruct them to get an RT and ride for a predetermined amount. Finally both users cash-in their RTs. An adversary should not be able to identify the order bit b better than by guessing. Note that similarly to blind signature schemes, if some executions abort, it may be easy to distinguish the users. We thus declare the adversary to lose if this happens, provided formally by outputting a random guess for b.

The above approach, however, guarantees less than we would like to achieve. In particular, it does not necessarily provide unlinkability of rides. The definition does not rule out adversaries knowing which trips belong together, it merely prevents them from identifying the user behind each of the two sequences. We ensure unlinkability by allowing the adversary to ask specific users to take trips (for a known amount) before and after entering a challenge phase in which the order of the users is random. Some caution has to be taken with regard to RTs though: an RT issued during the (non-anonymous) pre-challenge phase may not be redeemed during the challenge phase and an RT issued during the challenge phase may not be redeemed during the (non-anonymous) post-challenge phase. Moreover, we need to stipulate that the sums of refunds collected by both users during the challenge phase with an RT obtained in the pre-challenge phase need to be identical. Note that our more elaborated approach still covers the basic requirement that one cannot identify the user behind a trip sequence, because we let the adversary decide if and how often users take trips in the non-anonymous pre-challenge and post-challenge phase. In particular, the adversary could ask both users to immediately enter the challenge phase.

For our definition below, we make use of the same oracle-based attack model Register*, BuyTAT*, GetRT*, Enter, Exit, RedeemRT* as before, except that the adversary now takes over the roles of $\mathcal{T}$, $\mathcal{R}$, and $\mathcal{V}$. All users are assumed to be honest.

Definition 5 (User Privacy). *A pre-payment system with refunds is called private, if the probability for any PPT adversary $\mathcal{A}$ in predicting b in the experiment below is negligibly close to $\frac{1}{2}$. The scheme is called semi-honest private if the above holds under the condition that the adversary honestly follows the* GetRefund *protocol.*

Experiment User Privacy. Let us assume that $\mathcal{A}$ already created the system's public data and two identities ID_0, ID_1. Let two honest users $\mathcal{U}_0$ and $\mathcal{U}_1$ register to the system by means of Register* *using ID_0 and ID_1. The remaining experiment consists of the following three phases which are run sequentially. We assume that* Enter *and the corresponding call to* Exit *for a trip are always both executed in the same phase. Additionally, an RT issued during the*

pre-challenge phase may not be redeemed during the challenge phase and an RT issued during the challenge phase may not be redeemed during the post-challenge phase.

- Pre-Challenge Phase*: $\mathcal{A}$ may ask both users to run* BuyTAT*, GetRT*, Enter, Exit, RedeemRT*, multiple times and concurrently. $\mathcal{A}$ ensures that both users withdraw sufficiently many TATs for doing trips during the upcoming challenge phase.*
- Challenge Phase*: We pick a random and secret bit $b \leftarrow \{0,1\}$ and assign user $\mathcal{U}_b$ as the left user $\mathcal{U}_L$ and user $\mathcal{U}_{1-b}$ as the right user $\mathcal{U}_R$. We assume that both users use at most one refund token from the pre-challenge phase to collect some refunds in the challenge phase. $\mathcal{A}$ may execute the following steps repeatedly:*
  - *instruct $\mathcal{U}_L$ or $\mathcal{U}_R$ to run* GetRT* *to issue a new* RT *token*
  - *instruct $\mathcal{U}_L$ or $\mathcal{U}_R$ to run* RedeemRT* *to redeem an* RT *previously issued during the challenge phase*
  - *instruct $\mathcal{U}_L$ or $\mathcal{U}_R$ to run* Enter *for the next fresh TAT and a trip of value c*
  - *instruct $\mathcal{U}_L$ or $\mathcal{U}_R$ to run* Exit *for a trip previously started during the challenge phase, where the user may be asked to use an RT from the pre-challenge or the challenge phase to collect the refund*

  Let v_L, v_R be the sum of refunds of the users $\mathcal{U}_L$ and $\mathcal{U}_R$ collected using their refund tokens from the pre-challenge phase.
- Post-Challenge Phase*: $\mathcal{A}$ may ask both users to run* BuyTAT*, GetRT*, Enter, Exit, RedeemRT* *multiple times and concurrently.*

$\mathcal{A}$ outputs a guess b' for b. If one of the executions with the honest users has aborted, or if $v_L \neq v_R$, then output a random b' instead.

Remark 2. Note that in the experiment above, the adversary does not have access to the Corrupt oracle since we only care about the privacy of honest users. In fact, if $\mathcal{A}$ could corrupt those users, he would always win in the experiment as, e.g., he could count how many unused TATs each user owns before and after the challenge phase. Additionally, as the pre- and post-challenge phases are identifying by definition, whereas the challenge phase is not, we need to ensure that linkable transactions (e.g., obtaining and redeeming an RT) can only be executed in phases of the same "kind".

4 P4R: A Privacy-Preserving Transit Payment System

Our new payment system dedicated to transit applications follows the general framework given in Section 3. In our basic scheme, the TAT and RCT subsystems are loosely coupled and realized based on an extension of Brands' protocol. Our RT system is realized by using BLS signatures in a new way.

The essential parts of our basic scheme are shown in Figure 3. Note that for efficiency reasons we assume that users (with constrained devices) do not check signatures on RTs or MACs on RCTs. In other words, users trust the TA to issue correct refunds.

4.1 TAT Subsystem

To realize this subsystem, we build on Brands' scheme due to its efficient spending protocol. We slightly extend his scheme such that the ownership of a TAT can be shown twice without leaking a user's ID (which is needed for the RCT system).

- $\mathsf{KGenTAT}(1^{k_1})$ chooses a cyclic (elliptic curve) group $\mathbb{G}$ of prime order $q = \Theta(2^{k_1})$, group generators $g, g_1, g_2 \in \mathbb{G}$, a random number $x \in \mathbb{Z}_q^*$, and a collision-resistant hash function $H : \mathbb{G}^5 \to \mathbb{Z}_q$. The public key of $\mathcal{T}$ is $pk^{\mathsf{TAT}} = (\mathbb{G}, q, g, g_1, g_2, g^x, g_1^x, g_2^x, H)$ and its secret key is $sk^{\mathsf{TAT}} = x$.
- $\mathsf{Register}(\mathcal{U}(ID_{\mathcal{U}}, pk^{\mathsf{TAT}}), \mathcal{T}(.))$ is a protocol similar to the account setup in Brands' scheme. A secure and authenticated channel between $\mathcal{U}$ and $\mathcal{T}$ is assumed ($\mathcal{T}$ may also be represented by $\mathcal{V}$). $\mathcal{U}$ first presents some identity information $ID_{\mathcal{U}}$ to $\mathcal{T}$ like a credit card number, a bank account, a passport or a social security number. $\mathcal{T}$ first verifies the ID information and then $\mathcal{U}$ samples a random $id_{\mathcal{U}} \in \mathbb{Z}_q$, verifies that $g_1^{id_{\mathcal{U}}} g_2 \neq 1$, and sends $pk^{\mathcal{U}} = g_1^{id_{\mathcal{U}}}$ to $\mathcal{T}$. $\mathcal{U}$ proves his knowledge of the discrete logarithm of $pk^{\mathcal{U}}$ with respect to g_1 and stores $(pk^{\mathcal{U}}, sk^{\mathcal{U}}) = (g_1^{id_{\mathcal{U}}}, id_{\mathcal{U}})$, since he will later need this information when buying TATs. $\mathcal{T}$ checks whether $g_1^{id_{\mathcal{U}}}$ is already contained in the user database $\mathcal{DB}_{\mathcal{U}}$ and if not it is stored along with $ID_{\mathcal{U}}$.
- $\mathsf{BuyTAT}(\mathcal{U}(pk^{\mathsf{TAT}}, pk^{\mathcal{U}}, sk^{\mathcal{U}}), \mathcal{V}(sk^{\mathsf{TAT}}))$ is used by registered users to buy TATs. To this end, $\mathcal{U}$ sends $pk^{\mathcal{U}}$ to $\mathcal{V}$ who checks whether there exists an entry for the user in the database. After that, $\mathcal{U}$ proves knowledge of his secret key $sk^{\mathcal{U}}$. Then a TAT is computed interactively as an extended coin in Brands' scheme: $\mathsf{TAT}_i = (A_i, B_i, C_i, \mathsf{sig}_{sk^{\mathsf{TAT}}}(A_i, B_i, C_i))$, where $A_i = (pk^{\mathcal{U}})^{s_i} g_2^{s_i} = g_1^{id_{\mathcal{U}} s_i} g_2^{s_i}$, $B_i = g_1^{x_i} g_2^{y_i}$, and $C_i = g_1^{x_i'} g_2^{y_i'}$. Notice the additional commitment C_i in comparison to a coin in Brands' scheme. This value is later used in the RCT system to show ownership of TAT_i a second time. Multiple TATs can be withdrawn without repeating the identification in the beginning.
- $\mathsf{ShowTAT}(\mathcal{U}(\mathsf{TAT}_i, sk^{\mathcal{U}}, s_i, x_i, y_i), \mathcal{R}(pk^{\mathsf{TAT}}))$ corresponds to spending a coin in Brands' scheme. $\mathcal{U}$ sends TAT_i to $\mathcal{R}$ who verifies the signature and $\mathcal{U}$ proves ownership of the TAT using the secrets $sk^{\mathcal{U}}$, s_i and the randomness x_i and y_i he has committed to by means of B_i. It is important to note that the order in which B_i and C_i appear in TAT_i is fixed due to the signature. This prevents $\mathcal{U}$ from interchanging B_i and C_i or using C_i for a second trip with that TAT. TAT_i is stored along with the response (z_1, z_2) of the user in the TAT database $\mathcal{DB}_{\mathsf{TAT}}$.
- $\mathsf{IdentTAT}(\mathcal{DB}_{\mathsf{TAT}})$ is run by $\mathcal{T}$ to check for double-spending. If there exists two different response tuples (z_1, z_2) and (z_1^*, z_2^*) for the same TAT (with respect to B_i) in the TAT database, it recovers the user's secret key $id_{\mathcal{U}} = (z_1 - z_1^*)/(z_2 - z_2^*) \mod q$. By looking up $g_1^{id_{\mathcal{U}}}$ in the user database the cheater's identity $ID_{\mathcal{U}}$ can be revealed.

4.2 RCT Subsystem

By presenting an RCT to a reader at an exit turnstile, a user shows that he is eligible to receive a certain refund, which depends on the origin of his trip. Naturally, the TA wants to prevent a user from re-using an RCT, e.g., to demand the same refund several times, or exchanging his RCT with one from another user in order to fake the origin of his trip and obtain a higher refund. In order to discourage this kind of misbehavior, the idea is to bind a user's ID to the RCT tokens. Certainly, this cannot be done in a straightforward manner since it would violate anonymity. So instead, we bind an RCT to the TAT which just has been used to enter the system and force a user to prove ownership of this TAT when again he is leaving.

$\mathcal{U}$		$\mathcal{V}$ (online)
(1) User buys TATs		
$\mathsf{TAT}_i = (A_i, B_i, C_i, \mathsf{sig}(A_i, B_i, C_i))$ where $A_i = g_1^{id_{\mathcal{U}} s_i} g_2^{s_i}$, $B_i = g_1^{x_i} g_2^{y_i}$, $C_i = g_1^{x'_i} g_2^{y'_i}$	PoK of $id_{\mathcal{U}}$ $\longleftrightarrow$ blind sigs on (A_i, B_i, C_i) $\longleftrightarrow$	
(2) User receives fresh RT		
$\mathsf{RT} = \mathsf{SN}_{\mathsf{RT}}, R = 1, v = 0$	$\mathsf{SN}_{\mathsf{RT}}$ $\longleftarrow$	$\mathsf{SN}_{\mathsf{RT}} \leftarrow G$ add $\mathsf{SN}_{\mathsf{RT}}$ to $\mathcal{DB}_{\mathsf{RT}}$
(5) User redeems RT		
$r \leftarrow \mathbb{Z}_p^*$, $\mathsf{RT}' = \mathsf{RT}^r$, $R = Rr \bmod p$	$\mathsf{SN}_{\mathsf{RT}}, \mathsf{RT}', v, R$ $\longrightarrow$	check validity of $\mathsf{SN}_{\mathsf{RT}}$ $v \stackrel{?}{<} p - 1$ $e(\mathsf{SN}_{\mathsf{RT}}^R, h^{d^v}) \stackrel{?}{=} e(\mathsf{RT}', h)$ mark $\mathsf{SN}_{\mathsf{RT}}$ in $\mathcal{DB}_{\mathsf{RT}}$

$\mathcal{U}$		$\mathcal{R}$ (offline)
(3) User shows TAT and receives RCT when entering system		
	TAT_i $\longrightarrow$	check validity of TAT_i
$z_1 = x_i + c(id_{\mathcal{U}} s_i) \bmod q$	c $\longleftarrow$	$c \leftarrow \mathbb{Z}_q$
$z_2 = y_i + cs_i \bmod q$	z_1, z_2 $\longrightarrow$	$g_1^{z_1} g_2^{z_2} \stackrel{?}{=} A_i^c B_i$ add $(\mathsf{TAT}_i, z_1, z_2, c)$ to $\mathcal{DB}_{\mathsf{TAT}}$
	RCT_i $\longleftarrow$	$\mathsf{RCT}_i = (\mathsf{TAT}_i, \mathrm{ts}, id_{\mathcal{R}},$ $\mathsf{MAC}_K(\mathsf{TAT}_i, \mathrm{ts}, id_{\mathcal{R}}))$
(4) User shows RCT and collects refund on his RT when leaving system		
	RCT_i $\longrightarrow$	check validity of $\mathsf{RCT}_i, \mathsf{TAT}_i$
$z'_1 = x'_i + c'(id_{\mathcal{U}} s_i) \bmod q$	c' $\longleftarrow$	$c' \leftarrow \mathbb{Z}_q$
$z'_2 = y'_i + c' s_i \bmod q$	z'_1, z'_2 $\longrightarrow$	$g_1^{z'_1} g_2^{z'_2} \stackrel{?}{=} A_i^{c'} C_i$ add $(\mathsf{TAT}_i, z'_1, z'_2, c')$ to $\mathcal{DB}_{\mathsf{RCT}}$
$r \leftarrow \mathbb{Z}_p^*$	w $\longleftarrow$	determine refund $w \in \mathbb{Z}_{p-1}$
$\mathsf{RT}' = \mathsf{RT}^r$	RT' $\longrightarrow$	
$v = v + w$, $R = Rr \bmod p$, $\mathsf{RT} = \mathsf{RT}''$	RT'' $\longleftarrow$	$\mathsf{RT}'' = \mathsf{RT}'^{d^w}$

Fig. 3. Main parts of our a lightweight pre-payment system with refunds

- $\mathsf{KGenRCT}(1^{k_2})$ is executed by $\mathcal{T}$ and returns a random k_2-bit key K.
- $\mathsf{GetRCT}(\mathcal{U}(\mathsf{TAT}_i), \mathcal{R}(id_{\mathcal{R}}, K))$ is executed immediately after $\mathsf{ShowTAT}$. $\mathcal{R}$ sends $\mathsf{RCT}_i = (\mathsf{TAT}_i, \mathrm{ts}, id_{\mathcal{R}}, \mathsf{MAC}_K(\mathsf{TAT}_i, \mathrm{ts}, id_{\mathcal{R}}))$ to $\mathcal{U}$, where ts is a timestamp and $id_{\mathcal{R}}$ is the reader's ID. Note that TAT_i does not have to be sent back to $\mathcal{U}$.
- $\mathsf{ShowRCT}(\mathcal{U}(\mathsf{RCT}_i, sk^{\mathcal{U}}, s_i, x'_i, y'_i), \mathcal{R}(K, pk^{\mathsf{TAT}}))$ is executed between a user and a reader at the exit turnstile. $\mathcal{U}$ sends RCT_i to $\mathcal{R}$ who checks the MAC and the signature. If both are valid, $\mathcal{U}$ proves ownership of TAT_i contained in RCT_i using the randomness x'_i and y'_i committed to by C_i. If the proof succeeds, $\mathcal{R}$ determines the refund for user $\mathcal{U}$ based on

the timestamp and $id_{\mathcal{R}}$ contained in RCT_i. TAT_i and a transcript of the proof are stored in $\mathcal{DB}_{\mathsf{RCT}}$ for RCT re-use detection.
– $\mathsf{IdentRCT}(\mathcal{DB}_{\mathsf{RCT}})$ works as $\mathsf{IdentTAT}$ with respect to C_i.

4.3 RT Subsystem

Our realization of the RT system employs BLS signatures over bilinear groups in a new way to allow for a secure and privacy-preserving aggregation of refunds. The basic ideas are (i) to represent a refund value w as a w-times BLS signature $\mathsf{SN}_{\mathsf{RT}}^{d^w}$, where $\mathsf{SN}_{\mathsf{RT}}$ is a unique random serial number chosen by the TA and d is the TA's secret BLS signature key and (ii) let $\mathcal{R}$ do the sequential aggregation.

– $\mathsf{KGenRT}(1^{k_3})$ is a probabilistic algorithm run by $\mathcal{T}$ to choose cyclic groups G, G_T of prime order $p = \Theta(2^{k_3})$ with an efficiently computable non-degenerated pairing $e : G \times G \to G_T$. It also chooses a random generator $h \in G$ and an exponent $d \in \mathbb{Z}_p^*$. Let W be the set of all possible single-trip refunds. Then, the public key is $pk^{\mathsf{RT}} = (G, G_T, p, e, h, (h^{d^w})_{w \in W})$ and the secret key is $sk^{\mathsf{RT}} = d$. Note that pk^{RT} does not need to be stored on constrained user devices assuming users do not verify RTs.
– $\mathsf{GetRT}(\mathcal{U}(\cdot), \mathcal{V}(\cdot))$ is run by a user to receive a fresh blank refund token: $\mathcal{V}$ randomly samples an element $\mathsf{SN}_{\mathsf{RT}} \in G$ which is not already contained in the RT database, $\mathcal{DB}_{\mathsf{RT}}$, a central database accessible to all vending machines, sends it to the user, and adds it to this database. $\mathcal{U}$ sets $\mathsf{RT} = \mathsf{SN}_{\mathsf{RT}}$ and the corresponding refund amount v stored on the token to zero. Additionally, some variable R used to aggregate blinding factors is set to 1.
– $\mathsf{GetRefund}(\mathcal{U}(R, v, \mathsf{RT}, G, p), \mathcal{R}(w, sk^{\mathsf{RT}}))$ is executed immediately after $\mathsf{ShowRCT}$ to add a refund for a single trip to the user's RT. First, $\mathcal{R}$ sends the value w to $\mathcal{U}$, where $w \in \mathbb{Z}_{p-1}$ represents the refund amount in some unit (e.g., dimes). Then $\mathcal{U}$ computes a blinded version of his refund token RT as $\mathsf{RT}' = \mathsf{RT}^r$, where $r \leftarrow \mathbb{Z}_p^*$, and sends it to $\mathcal{R}$. The reader signs RT' by raising it to d^w. Upon receiving $\mathsf{RT}'' = \mathsf{RT}'^{d^w}$, $\mathcal{U}$ does not remove the blinding factor using $r^{-1} \bmod p$ but instead blindings are aggregated. That means, the blinding variable R is updated as $R = Rr \bmod p$ and the refund token as $\mathsf{RT} = \mathsf{RT}''$ by $\mathcal{U}$. In this way, we can save one exponentiation per refund. Moreover, v is set to $v = v + w$.
– $\mathsf{RedeemRT}(\mathcal{U}(R, v, \mathsf{RT}, \mathsf{SN}_{\mathsf{RT}}, G, p, \mathcal{V}(sk^{\mathsf{RT}}))$ is run by $\mathcal{U}$ to redeem his refund token: $\mathcal{U}$ computes a blinded version $\mathsf{RT}' = \mathsf{RT}^r$ of his RT, where $r \leftarrow \mathbb{Z}_p^*$, and updates $R = Rr \bmod p$. The refund token RT', the serial number $\mathsf{SN}_{\mathsf{RT}}$, the collected amount v, and the aggregated blinding factor R are sent to $\mathcal{V}$. $\mathcal{V}$ checks that $\mathsf{SN}_{\mathsf{RT}}$ is valid and has not already been redeemed by looking up $\mathsf{SN}_{\mathsf{RT}}$ in $\mathcal{DB}_{\mathsf{RT}}$. Then, $\mathcal{V}$ verifies whether the refund amount is within the allowed range $[0, p-1]$ and if the signature is valid by checking

$$e(\mathsf{SN}_{\mathsf{RT}}^R, h^{d^v}) = e(\mathsf{SN}_{\mathsf{RT}}, h)^{Rd^v} = e(\mathsf{SN}_{\mathsf{RT}}^{Rd^v}, h) = e(\mathsf{RT}', h)$$

involving the pairing e. Finally, $\mathsf{SN}_{\mathsf{RT}}$ is marked as redeemed in $\mathcal{DB}_{\mathsf{RT}}$.

5 Security of P4R

As discussed in Section 3.2, TA-security demands that an adversary cannot cheat the system by receiving reimbursements, which exceed the overall deposit minus the overall fare, without being identified during the double-spending checks. In order to show that P4R satisfies

Definition 4 we prove that certain security properties hold for the individual subsystems and then argue why this is sufficient for TA-security.

Security in the TAT subsystem. The TAT subsystem needs to guarantee that an adversary cannot do more trips than the number of TATs he bought. In order to do this, he would have to create a valid TAT by himself (at least one more than he legally withdrew), use a foreign (eavesdropped) TAT, or use a TAT issued to him more than once. As the TAT system is realized by a minor modification of Brands' scheme, all these possibilities are ruled out by the security of this e-cash scheme.

Lemma 1 (TAT Security). *Consider an adversary $\mathcal{A}$ as in Definition 4 and let ℓ and m denote the number of times* $\mathsf{BuyTAT}^*()$ *and* $\mathsf{Enter}()$ *was run successfully for users in $\mathcal{C}$, respectively. Let us assume that the DL assumption holds and Brands' blind signature protocol is restrictive and unforgeable. Then, with overwhelming probability it holds that $m \leq \ell$ unless* $\mathsf{IdentTAT}()$ *detects a double-spending by a user in $\mathcal{C}$.*

Proof Sketch. Ignoring the slight change we applied to the signature protocol in Figure 1, the Lemma immediately follows from the assumption that Brands' e-cash is secure, as $m > \ell$ corresponds to the case that more coins can be spent than have been legally withdrawn.

More precisely, due to the unforgeability of Brands' scheme, $\mathcal{A}$ cannot have obtained more than ℓ valid TATs. Assuming the DL problem is hard and using the soundness of Schnorr's identification protocol (which is executed in the scope of $\mathsf{BuyTAT}^*()$) as well as the soundness of the TAT registration protocol, each user (in $\mathcal{C}$) can only buy TATs in his name. That means, we can be sure that the public key $g_1^{id_\mathcal{U}}$ can only be used by a user who registered with secret key $id_\mathcal{U}$. Furthermore, restrictiveness essentially guarantees that the value A_i contained in each of these TATs is indeed of the form $A_i = g_1^{id_\mathcal{U} s_i} g_2^{s_i}$, where $id_\mathcal{U}$ is the secret key of the corresponding user and s_i is a serial number chosen by the user. The soundness of Brands' spending protocol ensures that if $\mathsf{Enter}()$ terminates successfully, the user must have presented a valid TAT, known a representation of A_i in terms of g_1 and g_2, and used a representation of B_i as blinding terms in the proof of knowledge. In particular, the representation of A_i can be extracted in case we see two protocol runs involving two different challenges but the same blinding terms. Now the DL assumption implies that it is infeasible, given a foreign TAT (for which $id_\mathcal{U}$ is not given), to find any representation of A_i, and thus to run $\mathsf{Enter}()$ successfully with this TAT. Furthermore, it implies that it is infeasible given a TAT for which one representation of A_i and B_i, respectively, is known to come up with a second, different representation for A_i or B_i (cf. Corollary 8 in [8]). Thus, each time $\mathsf{Enter}()$ is run for a particular TAT issued to a user, this user must prove knowledge of $id_\mathcal{U}$ and s_i and use the same fixed blinding terms x_i and y_i. Hence, in case the same TAT is used more than once, $id_\mathcal{U}$ can be extracted by means of $\mathsf{IdentTAT}()$ with overwhelming probability.

Finally, it is easy to see that the additional parameter C_i we added to Brands' basic scheme does not change anything. In particular, C_i cannot be used as a replacement for B_i in the scope of a $\mathsf{ShowTAT}$ protocol run (with the goal to use a TAT twice) as the order of the components in a TAT tuple is fixed due to the signature. □

Security in the RCT subsystem. This system ensures that the user gets charged exactly as much as his trip costs. In particular, it prevents an adversary from creating RCTs himself, using the same RCT twice, or using a "foreign" RCT.

Lemma 2 (RCT Security). *Consider an adversary $\mathcal{A}$ as in Definition 4. Let m denote the number of times* Exit() *was run successfully for users in $\mathcal{C}$ and let $\mathfrak{F}_1, \ldots, \mathfrak{F}_m$ denote the authentic fares and $\mathfrak{F}'_1, \ldots, \mathfrak{F}'_m$ the calculated fares thereby incurred.*[6] *Assume that the DL assumption holds, Brands' blind signature protocol is restrictive, and the MAC used in the RCT system is unforgeable. Then, with overwhelming probability it holds that $\mathfrak{F}_i = \mathfrak{F}'_i$ unless* IdentRCT() *detects a double-spending by a user in $\mathcal{C}$.*

Proof Sketch. Note that a calculated fare, $\mathfrak{F}'_i$, may only differ from the authentic fare, $\mathfrak{F}_i$, for a trip if the RCT presented during Exit() contains a reader ID or a timestamp which differs from the real entrance point or time, i.e., the data given as input to Enter(). Due to the unforgeability of the employed MAC scheme, it is not possible for $\mathcal{A}$ to create fake RCTs or modify the reader or timestamp information of a given RCT. Thus, in order to run Exit() successfully, the RCT needs to be at least issued by a legitimate reader (via Enter()) to some user for some trip. Now there are two cases we have to consider: Either the RCT has not been issued to the particular user for which Exit() is executed (foreign RCT), or it is has been issued to this user but not for the current trip (i.e., not during the preceding call of Enter() for this user). Note that showing an RCT successfully also means to prove knowledge of $id_{\mathcal{U}}$ and s_i encoded in the TAT included in the RCT. Similar arguments as in the proof of Lemma 1 rule out the first case, using the soundness of Brands spending protocol and the assumptions that the DL problem is hard and Brands' signatures are restrictive. Furthermore, since in our model an RCT has to be presented at the end of each trip, the second case implies that the RCT has been used before by that user. Hence, by the security properties of Brands' scheme and the DL assumption, IdentRCT() will reveal the secret key of the double-spender in $\mathcal{C}$ with overwhelming probability. □

Security in the RT subsystem. The RT system guarantees that the sum of refund amounts cashed in never exceeds the sum of refund values issued by the readers. In particular, it prevents an adversary from forging RTs, redeeming an RT twice, or stealing refunds from honest users.

Lemma 3 (RT Security). *Consider an adversary $\mathcal{A}$ as in Definition 4. Let k' and k denote the number of times* Exit() *and* RedeemRT*() *was run successfully for users in $\mathcal{C}$, respectively. Let $\mathfrak{R}'_1, \ldots, \mathfrak{R}'_{k'}$ be the corresponding single-trip refund values issued to them and $\mathfrak{R}_1, \ldots, \mathfrak{R}_k$ be the total refund amounts redeemed by them. Let n denote an upper bound on the single-trip refunds and assume that the n-Σ-Incremental DH assumption holds. Then with overwhelming probability we have $\sum_{i=1}^{k} \mathfrak{R}_i \leq \sum_{i=1}^{k'} \mathfrak{R}'_i$.*

Proof Sketch. The adversary wins the RT security game if he manages to redeem a bigger refund amount than the one issued to the group of malicious users. To accomplish this, $\mathcal{A}$ could either (a) steal refunds from the group of honest users or (b) leverage the refunds issued to malicious users.

Our model considers a passive adversary with regard to the set of honest users. Hence, an RT owned by a user in $\mathcal{C}$ (i.e., originally issued to him), cannot be sneaked in into an execution of Exit() of an honest user not in $\mathcal{C}$ to obtain his refund surreptitiously. Thus, $\mathcal{A}$ is

[6] Note that in our model the numbers of authentic and calculated fares coincide: During each successful run of Exit() exactly one fare is calculated and as we demand that there is exactly one preceding successful execution of Enter(), there is also one authentic fare associated.

not able to simply collect single-trip refunds on behalf of honest users. Moreover, $\mathcal{A}$ cannot redeem an RT owned by a user who is not in $\mathcal{C}$, provided that the DL assumption holds (which is implied by n-Σ-Incremental DH).[7] Basically, the reason is that $\mathcal{A}$ would have to provide the correct (random) blinding factor R, which is used as an exponent, for the verifications performed in the scope of $\mathsf{RedeemRT}$ (cf. Appendix A for a formal proof). Thus, $\mathcal{A}$ cannot steal refunds from honest users.

Let us now discuss why $\mathcal{A}$ cannot receive a higher refund than the total one issued to malicious users. First, $\mathcal{A}$ cannot redeem an RT (of a user in $\mathcal{C}$) twice as the online double-spending check ensures that each redeemed refund $\mathfrak{R}_i$ must be associated with a different random serial number. Hence, it is not hard to see that the only remaining possibility for $\mathcal{A}$ to cheat the RT system is to claim a higher refund for an RT serial number. However, this is infeasible under the Σ-Incremental DH assumption: Due to the checks performed by the reader, in order to claim a refund $\mathfrak{R}_i$, a user needs to present the blinding factor R_i, the serial number of the refund token $\mathsf{SN}_{\mathsf{RT}_i}$, and the refund token RT'_i such that ${\mathsf{RT}'_i}^{R_i^{-1}}$ equals the $d^{\mathfrak{R}_i}$-th power of $\mathsf{SN}_{\mathsf{RT}_i}$, where $\mathsf{SN}_{\mathsf{RT}_i}$ is a random number chosen by the TA. To see this, note that the verification equation satisfies $e(\mathsf{SN}_{\mathsf{RT}_i}^{R_i}, h^{d^{\mathfrak{R}_i}}) = e(\mathsf{RT}', h)$ and $\mathfrak{R}_i < p-1$, and therefore $e(\mathsf{SN}_{\mathsf{RT}_i}, h^{d^{\mathfrak{R}_i}}) = e({\mathsf{RT}'}^{R_i^{-1}}, h)$. Thus, we are in the setting of the n-Σ-Incremental DH assumption where $\mathsf{SN}_{\mathsf{RT}_1}, \ldots, \mathsf{SN}_{\mathsf{RT}_k}$ are the challenges. Signing with $d^{\mathfrak{R}'_i}$ can be interpreted as applying the $\mathcal{O}_d$-oracle $\mathfrak{R}'_i$ times. So the oracle would be applied $\sum_{i=1}^{k'} \mathfrak{R}'_i$ times in total. The case $\sum \mathfrak{R}_i > \sum \mathfrak{R}'_i$ would violate the assumption. □

Security in the overall system. Putting all pieces together, we can show the following Theorem for P4R:

Theorem 1 (TA Security). *Let us assume that the n-Σ-Incremental DH Assumption holds, Brands' blind signature protocol is restrictive and unforgeable, and the MAC scheme used in the RCT system is unforgeable. Then P4R (with single-trip refunds bounded by n) is TA-secure in the sense of Definition 4.*

Proof Sketch. As in Definition 4, let $\mathfrak{U}$ be the cost of one TAT and let ℓ and m denote the number of times $\mathsf{BuyTAT}^*()$ and $\mathsf{Enter}()/\mathsf{Exit}()$ was run successfully for users in $\mathcal{C}$, i.e., ℓ is the number of bought TATs and m is the real number of trips. Furthermore, let $\mathfrak{F}_1, \ldots, \mathfrak{F}_m$ denote the authentic fares, $\mathfrak{F}'_1, \ldots, \mathfrak{F}'_m$ the calculated fares, and $\mathfrak{R}'_1 = \mathfrak{U} - \mathfrak{F}'_1, \ldots, \mathfrak{R}'_m = \mathfrak{U} - \mathfrak{F}'_m$ the corresponding refund values thereby incurred. Finally, by $\mathfrak{R}_1, \ldots, \mathfrak{R}_k$ we denote the reimbursements paid by $\mathcal{V}$ in successful executions of $\mathsf{RedeemRT}^*()$ to users in $\mathcal{C}$.

Then from Lemma 1, we know that $m \leq \ell$ unless a TAT double spending has been detected. Furthermore, we have

$$\sum_{i=1}^{k} \mathfrak{R}_i \overset{L.3}{\leq} \sum_{j=1}^{m} \mathfrak{R}'_j = \sum_{j=1}^{m} \mathfrak{U} - \mathfrak{F}'_j \overset{L.2}{=} \sum_{j=1}^{m} \mathfrak{U} - \mathfrak{F}_j \overset{L.1}{\leq} \ell\mathfrak{U} - \sum_{j=1}^{m} \mathfrak{F}_j,$$

unless a TAT or RCT double-spending has been detected (see also Appendix C). □

[7] Actually, $\mathcal{A}$ is able to redeem a blank RT (containing no refunds), preventing an honest user from later cashing in his collected refunds. While this does not violate our security definition, it may be undesirable as it constitutes a denial of service attack against honest users. Fortunately, they can easily be protected with little overhead, e.g., by forcing users to do a PoK of a secret y when redeeming, where y is randomly chosen by the anonymous user when obtaining an RT and h^y is associated with this RT by the TA.

6 User Privacy of P4R

We show that P4R is semi-honest private according to Definition 5. This property essentially follows from the fact that all values that would allow to link transactions (such as user IDs, refund tokens, etc.) are information-theoretically hidden.

Theorem 2 (Privacy). *P4R is semi-honest private following Definition 5.*

Proof Sketch. Let $\mathcal{A}$ be an honest-but-curious adversary as in Definition 5 and let $\mathcal{U}_0$ and $\mathcal{U}_1$ denote the two honest users of the privacy game. We show that $\mathcal{A}$ has a negligible advantage in winning the privacy game by considering a simulation game which is independent of the challenge bit b but perfectly indistinguishable from the real game. In this simulation game, the challenger behaves exactly like the real challenger in the pre-challenge and post-challenge phases but acts differently in the challenge phase: he assigns $\mathcal{U}_0$ as the left user and $\mathcal{U}_1$ as the right user in a *fixed* manner and independent of the bit b. Clearly, the probability to guess b in this game is $\frac{1}{2}$.

The challenge phase in the simulation game differs from the one in the real game if the challenger instructs a different user, e.g., $\mathcal{U}_0$ instead of $\mathcal{U}_1$ or vice versa, to do a trip or get/redeem a new refund token. The crucial observation is that the views generated by these transactions for $\mathcal{U}_0$ and $\mathcal{U}_1$ are perfectly indistinguishable.

This is obvious for GetRT^* and $\mathsf{RedeemRT}^*$ as these transactions do not contain any user-specific information whatsoever. Hence, $\mathcal{U}_0$ may perform these transactions on behalf of $\mathcal{U}_1$ (or vice versa) in the challenge phase without the adversary taking notice. As the action of obtaining and redeeming the same RT is linkable, our model forbids to do one of these transactions in a phase where the user is non-anonymous by definition, i.e., pre- or post-challenge phase, while performing the other in the challenge phase.

Also Enter/Exit views for $\mathcal{U}_0$ and $\mathcal{U}_1$ are perfectly indistinguishable: Let us ignore RCTs and RTs for a moment. Then Proposition 1 guarantees that a particular Enter/Exit transcript may belong to any user and any execution of BuyTAT^*.

It is easy to see that this also holds in the presence of RCTs which are computed over TATs and do not contain any additional information about users. They can only be used to link an Enter transcript with an Exit transcript which is intended and can be done anyway in a 2-showable coins scheme. For this reason, our privacy experiment does not allow to execute Enter in the pre-challenge phase and the corresponding call to Exit in the challenge phase or, similarly, execute Enter in the challenge phase and run Exit in the post-challenge phase, as the pre- and post-challenge phase is identifying.

Moreover, one needs to observe that also for the views generated by GetRefund an information-theoretic statement similar to Proposition 1 holds, i.e., there are unique choices such that the pre-/post-challenge phase view for a user and the challenge-phase view for, possibly, a different user "can be made correspondent". The main argument is that for any two observed (randomized) refund tokens RT'_1 and RT'_2 (not necessarily from the same user) there is a unique $r \in \mathbb{Z}_p^*$ such that $\mathsf{RT}'_2 = {\mathsf{RT}'_1}^r$.

Finally, as the adversary can link an RT issued during the pre-challenge and redeemed during the post-challenge phase to a specific user, we need that the sum of refunds (which is revealed to the adversary) collected during the challenge phase with this kind of RT to be the same for both users. This allows a challenge phase view, i.e., a series of Exit calls

for a particular (hidden) user involving a pre-challenge RT, to correspond to any of the two $\mathsf{RedeemRT}^*$ calls in the post-challenge phase.

Based on the arguments above, one can conclude that the real game and the simulation game are perfectly indistinguishable. □

7 Performance Evaluation

We evaluate P4R's performance by estimating the storage space required on the user device and the overall database requirements for the scheme. Furthermore, we present implementation results for a platform that can be seen as an approximation of future payment tokens. Our evaluation of the execution time is limited to the user device, since the vending machines and turnstiles can be equipped with powerful hardware, or connected to a back-end system and efficiently execute complex algorithms. However, as argued in Section 1 payment devices must be assumed to be low-cost and low-power devices, such as contactless smart cards or RFID tokens.

The UMass Moo [47], a computational RFID tag designed at the University of Massachusetts, approximates hardware that could be used in future contactless payment devices. It is passively powered, i.e., it harvests its energy from the RF-field generated by the reader and can communicate over a distance of up to several meters. Its core is an MSP430F2618 microcontroller, an ultra-low power MCU from Texas Instruments. This microcontroller has a 16-bit RISC processor, 8 KB of RAM and 116 KB of flash memory. Beneficial for the implementation of the elliptic curve arithmetic is its memory-mapped hardware multiplier supporting multiply-and-accumulate, i.e. taking 16-bit inputs and accumulating the 32-bit results. While, as a prototyping device, the Moo is costly, it can be assumed that if rolled out at a large quantity, its price would be in the range of a few dollars. That a high quantity roll-out is a realistic scenario for payment devices for public transport has been shown with the Octopus Card, where over 25 million cards are in circulation today. Furthermore, this example shows that paying a small amount ($6.5 for an Octopus Card) for a payment device is not a roadblock for its acceptance.

Our work optimizes and extends the implementation of Brands' e-cash scheme presented in [25].

7.1 Elliptic Curve Cryptography

The RT subsystem is based on BLS signatures over bilinear groups. In our setting only vending machines need to verify BLS signatures, thus, we provide them with the BLS secret key. Then, the complex pairing computation can be converted to a scalar multiplication on the elliptic curve: instead of checking the pairing equation during $\mathsf{RedeemRT}$, $\mathcal{V}$ only needs to check whether $\mathsf{RT}' \stackrel{?}{=} \mathsf{SN}_{\mathsf{RT}}^{Rd^v}$ holds.

We use the same curve for the group $\mathbb{G}$ underlying the TAT system and the group G underlying the RT system, which leads to $\mathbb{Z}_p \hat{=} \mathbb{Z}_q$. Assuming that an 80-bit security level presents sufficient security for a micro-payment system, we base the scheme on *secp160r1* [39], which is based on a generalized Mersenne prime.

The Montgomery Powering Ladder [34] was used to implement the point multiplication on the elliptic curve, thus making timing attacks more difficult. For the execution of the

point multiplication, the input points were converted to Jacobian coordinates. Meloni found that, when executing a point addition using Jacobian coordinates, the number of required modular multiplications can be reduced, if the input points share the same Z-coordinate [33]. Moreover, updating the Z-coordinate of the original point to make it the same as the Z-coordinate of the resulting point can be done at no extra cost. With our implementation, a point multiplication on the chosen elliptic curve can be executed in 5.6 million clock cycles on the MSP430F2618, i.e., 350 ms at an operating frequency of 16 MHz.

7.2 Hash Functions and Message Authentication Code

During the setup, $\mathcal{T}$ decides on hash functions and a message authentication code. Hash functions should accept as input elements in G_q and elements in $\mathbb{Z}_q$ and produce 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. We implemented AES-hash [16] and based the hash function on the 160-bit Rijndael block cipher *Rijndael160*. Furthermore, we re-use Rijndael160 to also implement the block cipher-based message authentication code CMAC [18], thereby saving code space.

7.3 Identification Protocol

As suggested in Section 4, we use Schnorr's protocol whenever the user needs to prove knowledge of his secret key. We implement non-interactive Schnorr signatures enabling a user on the one hand to identify himself and at the same time to specify the number of TATs he would like to withdraw in an authenticated fashion. The message m being signed specifies the desired number of TATs together with date/time information.

7.4 Storage Estimation

In our implementation, G and $\mathbb{G}$ are cyclic groups generated by the points on the curve *secp160r1*, spanned by the base point g of order $q = p$, where q is a 161-bit prime. We store points in affine coordinates, thus requiring 40 B of storage for an element in $G = \mathbb{G}$ and 21 B for a scalar in $\mathbb{Z}_q = \mathbb{Z}_p$. Based on these parameter choices, our storage space estimations for P4R are summarized in Table 1. 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 TAT key comprises of two elements, an element in $\mathbb{G}$ and one in $\mathbb{Z}_q$, which requires 61 B 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 $\mathbb{G}$ 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 $\mathbb{Z}_p$, 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, which leads to a more than sufficiently 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 $id_{\mathcal{R}}$ to require 10 B of memory each, while storing $\mathsf{MAC}_K(\mathsf{TAT}_i, \mathrm{ts}, id_{\mathcal{R}})$ requires 21 B, which adds up to additional 41 B of data.

Table 1. Storage space estimation for payment data

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

7.5 Database Estimation

Here we estimate the size of the four databases required for P4R. $\mathcal{DB}_{\mathcal{U}}$ stores identifying information for each user along with his public key and is a comparably small database. $\mathcal{DB}_{\mathsf{TAT}}$ and $\mathcal{DB}_{\mathsf{RCT}}$ 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, $\mathcal{T}$ uses a database $\mathcal{DB}_{\mathsf{RT}}$ 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 [31]. Hence, $\mathcal{DB}_{\mathsf{TAT}}$ and $\mathcal{DB}_{\mathsf{RCT}}$ 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 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.

7.6 Implementation Results

We implement our scheme in C using the IAR Embedded Workbench IDE for TI MSP430. Time-critical parts, i.e., multiplication and squaring in $\mathbb{Z}_p$ have been sped up using assembly language. Implementation results for the user side's computation on the Moo are presented in Table 2 for 4 MHz and 16 MHz. The maximum frequency supported by the MSP430F2618 is 16 MHz, but the microcontroller on the Moo is operated at only 4 MHz, due to power constraints that result from powering it passively.

The results show that the protocols for entering the system (ShowTAT & GetRCT) can be executed efficiently even on the chosen unoptimized prototyping device. 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 execution time for receiving a refund (ShowRCT & GetRefund) is a little 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

Table 2. Timing results for the user side's computation of buying a TAT (BuyTAT), receiving a refund token (GetRT), paying a fare (ShowTAT & GetRCT), getting a refund (ShowRCT & GetRefund) and redeeming an accumulated refund (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

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 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.

7.7 Performance Comparison of P4R with Brands' E-cash

In this section, we briefly compare the performance of P4R and a transportation payment system directly built from Brands' e-cash. As justified in Section 1.2, 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.

Table 3. 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	-

Table 3 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 the P4R databases $\mathcal{DB}_{\mathsf{TAT}}$, $\mathcal{DB}_{\mathsf{RCT}}$ and $\mathcal{DB}_{\mathsf{RT}}$.

8 Extensions and Variations of the Basic Scheme

Discouraging the Theft of Readers. It is important to note that in our RT system; readers act as money printing machines, where all readers share the same key to sign refunds.

Hence, it is vital to protect readers from becoming compromised or stolen, e.g., by using tamper-resistant hardware. We propose an additional measure to discourage the compromise of readers, namely, we ensure that the sum of all issued refunds never exceeds the total deposit for TATs. In this way, an adversary in possession of a stolen reader is able to get the full deposit for a purchased TAT and thus a ride for free but is not able to print additional money. This can be realized by binding the issuing of a fresh RT to the purchase of a bundle of TATs. More precisely, if k TATs are purchased, each for the price of t cents, then the serial number of the blank RT is associated with an upper bound of kt cents which is checked when the RT is redeemed. In this way, we can improve the TA security of our basic system. However, on the downside, a user also loses some privacy by this measure since redeeming refunds is not anonymous anymore and the TA knows the number of addends a user's total refund amount is composed of. The right-hand side of Figure 2 illustrates the implications of the latter by an example.

Reducing Information Leakage due to Disclosure of Refunds. In our basic scheme, there exists a *direct* link between total refunds and trips: if an anonymous user redeems a refund token worth \$5, then, his sequence of trips is within the set of all possible trips such that the issued refunds sum up to \$5. Combinations of trips which cannot lead to this amount are excluded, e.g, trips worth \$2 and \$3.10, respectively. Our goal now is to make this link a little more "fuzzy" and enlarge the set of possible trips. Instead of forcing a user to add the complete refund issued by a reader to a single refund token, we let the user split up the refund amount into fractions of his choice which are then added to several of his refund tokens.

For instance, a user could always have two RTs in parallel for collecting refunds. If he is eligible for a refund of w cents, he chooses a number r between 0 and w uniformly at random and tells the reader that he would like r cents to be added to the one RT and $w-r$ to the other RT (he needs to send blind versions of both RTs to the reader). Of course, this modification leads to some overhead regarding storage, computation, and communication compared to the basic scheme. In particular, the user device needs to blind multiple RTs to obtain a refund. We leave a more thorough and formal analysis of the "gain" in terms of privacy achieved by this approach as future work.

Encouraging the TA to Play Honest. For efficiency reasons, we assume in the basic scheme that low-cost user devices do not verify whether the correct refund was received. However, more powerful devices, e.g., NFC-enabled smart phones are able to store the large keys and do the pairing-based signature verification. So in order to further encourage the TA to play fair, we propose a hybrid system where parts of the users participate in the transportation payment system using smart phones or where at least some authority regularly verifies the correctness of issued refunds by means of more powerful devices.

In the following we describe some simple ideas which can be used to alleviate the storage requirements by avoiding the need to store a key h^{d^w} for each possible refund value w on the user devices. A first approach would be to let the TA sign all possible keys h^{d^w} using a separate signature scheme and store the public key of this scheme on each user device. Then a reader issuing a refund w could send h^{d^w} along with the corresponding certificate to the user. Of course, this results in an additional signature verification for the user. Alternatively or in addition readers may issue a refund of $w = \sum_{i=0}^{k} w_i B^i$ cents, where B is some basis (e.g., $B = 10$), $0 \leq w_i \leq B-1$, by sending the elements $a_0 = \mathsf{RT}'^{d^{w_0 B^0}}, a_1 = a_0^{d^{w_1 B^1}}, \ldots, a_k =$

$a_{k-1}^{d^{w_k B^k}}$ to the user. To verify the signatures on the a_i's the user only needs to be given the elements $h^{d^{wB^i}}$ for $1 \leq w \leq B-1$ and $0 \leq i \leq k$. For instance, if we want to allow refunds up to \$9.99 where the refund unit is cents, we could set $B = 10$ and $k = 2$. This results in 27 public key elements (instead of 999) and 3 signatures per refund that need to be verified.

Ensuring Security in Case of an Aborted Entrance. Our security model demands that every entry operation of a user is properly terminated by a corresponding exit operation. In an actual system implementation this has to be ensured, e.g., by physical means. Although, this is a reasonable requirement, many existing transportation infrastructures allow users to abort an entry operation, i.e., to show a ticket but not entering the system with this ticket. Thus, it would increase the practicality of P4R if the scheme could live without this requirement. Aborting an entry operation in our scheme after GetRCT has already been executed would leave the user with a spare RCT token. The essential problem is that the user can now enter the system at some point with a second TAT and will be in possession of two unused RCTs which *both* can be used to get (higher) refunds: the user can show the spare RCT to get a higher refund on his first trip, leaving him with the second (real) RCT as spare RCT, and the process can be repeated. Let us consider the following modification of the system: We allow a user to only have a single RT at a time and bind this RT (and the current balance) to an RCT at entrance. This can be realized by personalizing RTs (by means of a PoK of $sk^{\mathcal{U}}$) and including the current (blinded) RT in the MAC computation. The crucial observation is that this modification ensures that the spare and the real RCT will both be bound to the same RT balance. Hence, either the spare or the real RCT are of use *but not both* (more precisely, either the refund collected with the spare or with the real RCT can be redeemed but not both). As the user paid twice the full deposit to obtain the two RCTs but only does one trip (he aborted the other) and is only able to get one refund, he will eventually pay the maximal fare for this trip, even if he is able to get the maximal refund with the spare RCT. A more formal proof is left as future work. Note that this modification also has some privacy implications as obtaining and redeeming refunds is not anonymous anymore.

9 Conclusion

This paper considers privacy-preserving pre-payments with refunds for the transit domain. We present a formal framework including security and privacy models for this type of system as well as an efficient instantiation, called P4R, based on Brand's e-cash and BLS signatures. P4R improves the performance, reduces storage requirements, and allows for flexible pricing as compared to a naive e-cash based solution. Additionally, we show the flexibility of our approach by presenting several tradeoffs with respect to security, privacy, and efficiency. Last but not least, a proof-of-concept implementation as well as implementation results and estimates are presented.

While our implementation results suggest that the withdrawal protocol might be a bottleneck with respect to our unoptimized prototyping device, we believe that this is not an issue anymore for a dedicated payment device (featuring a crypto-coprocessor in contact mode) which we can expect to be available in a large transportation system. An interesting future work is also to improve the performance at the protocol level, e.g., by optimizing

Brands' withdrawal protocol or by replacing the whole pre-payment system with a more efficient instantiation than Brands' e-cash. An ad-hoc approach in this direction might be to assume that users do not need to check the validity of a TAT which immediately saves four exponentiations.

Although, in this paper we limited our considerations to the case of subway systems, P4R can easily be applied to toll collection systems as well. Here cars pass a toll booth at an entrance and exit of a highway. While the range of prices might be larger, this could easily be accounted for with different denominations of TATs. Further transportation systems, such as buses and trains could also benefit from P4R, if they could be modified in a way that a user is forced to pass a reader at the entrance *and* exit points of the transportation system. We leave an adaption of P4R to make it more suitable for this domain, e.g., by leaving it up to the user to show his RCT at the exit to get a refund, as an open problem.

Another interesting modification would be the extension of P4R to also allow for attribute-based discounts, like a percental discount on the fare for elderly people and children for example. While we believe this is not too hard to realize, it requires several extensions of the security and privacy model (e.g., re-defining authentic trip costs), the system (e.g., in the simplest case, introducing different types of TATs to represent attributes), and the proofs (show that attributes cannot be misused to get higher refunds). In fact, to construct a fairly complex and flexible attribute system with P4R, we could draw from ideas in [26] where such a system is built on top of Brands' e-cash. However, we would like to note that this type of modification would also reduce the level of privacy since additional side information is given to an adversary which helps to link trips (of anonymous users). Trips involving different attributes may not form a plausible trip sequence. Hence, one has to be careful not do introduce too many different attribute types and values.

Acknowledgements. We would like to thank Marc Fischlin for valuable input to the security and privacy proofs as well as the anonymous reviewers of FC 2013 for their helpful comments.

References

1. Josep Balasch, Alfredo Rial, Carmela Troncoso, Bart Preneel, Ingrid Verbauwhede, and Christophe Geuens. Pretp: Privacy-preserving electronic toll pricing. In *USENIX Security Symposium*, pages 63–78. USENIX Association, 2010. 3
2. Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM Conference on Computer and Communications Security*, pages 1087–1098. ACM, 2013. 4
3. Foteini Baldimtsi and Anna Lysyanskaya. On the security of one-witness blind signature schemes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT (2)*, volume 8270 of *Lecture Notes in Computer Science*, pages 82–99. Springer, 2013. 9
4. Mihir Bellare, Juan A. Garay, Ralf Hauser, Amir Herzberg, Hugo Krawczyk, Michael Steiner, Gene Tsudik, Els Van Herreweghen, and Michael Waidner. Design, implementation, and deployment of the ikp secure electronic payment system. *IEEE Journal on Selected Areas in Communications*, 18:611–627, 2000. 3
5. Erik-Oliver Blass, Anil Kurmus, Refik Molva, and Thorsten Strufe. PSP: private and secure payment with RFID. In Ehab Al-Shaer and Stefano Paraboschi, editors, *WPES*, pages 51–60. ACM, 2009. 3
6. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2003. 4

7. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004. 7
8. Stefan Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS-R9323, CWI, 1993. 3, 21
9. Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318. Springer, 1993. 8, 9
10. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321. Springer, 2005. 3
11. Jan Camenisch, Anna Lysyanskaya, and Mira Meyerovich. Endorsed e-cash. In *IEEE Symposium on Security and Privacy*, pages 101–115. IEEE Computer Society, 2007. 3
12. Jan Camenisch, Jean-Marc Piveteau, and Markus Stadler. An efficient electronic payment system protecting privacy. In Dieter Gollmann, editor, *ESORICS*, volume 875 of *Lecture Notes in Computer Science*, pages 207–215. Springer, 1994. 3
13. Sébastien Canard and Aline Gouget. Divisible e-cash systems can be truly anonymous. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 482–497. Springer, 2007. 3
14. Agnes Hui Chan, Yair Frankel, Philip D. MacKenzie, and Yiannis Tsiounis. Mis-representation of identities in e-cash schemes and how to prevent it. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT*, volume 1163 of *Lecture Notes in Computer Science*, pages 276–285. Springer, 1996. 8
15. David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO*, pages 199–203. Plenum Press, New York, 1982. 3
16. Bram Cohen. AES-hash. Available at `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/aes-hash/aeshash.pdf`, 2001. 26
17. Jeremy Day, Yizhou Huang, Edward Knapp, and Ian Goldberg. Spectre: spot-checked private ecash tolling at roadside. In Yan Chen and Jaideep Vaidya, editors, *WPES*, pages 61–68. ACM, 2011. 3
18. Morris Dworkin. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, May 2005. 26
19. E-ZPass. E-ZPass. http://www.e-zpassiag.com/ [Accessed: 2013-07-03], 2013. 1
20. Matthias Enzmann, Marc Fischlin, and Markus Schneider. A privacy-friendly loyalty system based on discrete logarithms over elliptic curves. In Ari Juels, editor, *Financial Cryptography*, volume 3110 of *Lecture Notes in Computer Science*, pages 24–38. Springer, 2004. 4, 6
21. Marc Fischlin, Anja Lehmann, and Dominique Schröder. History-free sequential aggregate signatures. In Ivan Visconti and Roberto De Prisco, editors, *SCN*, volume 7485 of *Lecture Notes in Computer Science*, pages 113–130. Springer, 2012. 4
22. Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001. 7
23. Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 119–132. Springer, 2004. 2
24. Thomas S. Heydt-Benjamin, Hee-Jin Chae, Benessa Defend, and Kevin Fu. Privacy for public transportation. In George Danezis and Philippe Golle, editors, *Privacy Enhancing Technologies*, volume 4258 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2006. 3
25. Gesine Hinterwälder, Christof Paar, and Wayne P. Burleson. Privacy preserving payments on computational rfid devices with application in intelligent transportation systems. In *RFIDSec*, volume 7739 of *Lecture Notes in Computer Science*, pages 109–122. Springer, 2012. 25
26. Gesine Hinterwälder, Christian T. Zenger, Foteini Baldimtsi, Anna Lysyanskaya, Christof Paar, and Wayne P. Burleson. Efficient e-cash in practice: Nfc-based payments for public transportation systems. In Emiliano De Cristofaro and Matthew Wright, editors, *Privacy Enhancing Technologies*, volume 7981 of *Lecture Notes in Computer Science*, pages 40–59. Springer, 2013. 31
27. Tibor Jager and Andy Rupp. The semi-generic group model and applications to pairing-based cryptography. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 539–556. Springer, 2010. 6, 34, 36
28. Florian Kerschbaum, Hoon Wei Lim, and Ivan Gudymenko. Privacy-preserving billing for e-ticketing systems in public transportation. In Ahmad-Reza Sadeghi and Sara Foresti, editors, *WPES*, pages 143–154. ACM, 2013. 3

29. Ueli M. Maurer. Abstract models of computation in cryptography. In Nigel P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005. 34
30. MBTA. Massachusetts Bay Transportation Authority: CharlieCards & Tickets. `http://www.mbta.com/fares_and_passes/charlie/` [Accessed: 2013-07-03], 2013. 1
31. Massachusetts Bay Transportation Authority (MBTA). MBTA ScoreCard 2013 March [Feb'13 Data]. http://www.mbta.com/about_the_mbta/scorecard/, 2013. 27
32. Sarah Meiklejohn, Keaton Mowery, Stephen Checkoway, and Hovav Shacham. The phantom tollbooth: Privacy-preserving electronic toll collection in the presence of driver collusion. In *USENIX Security Symposium*. USENIX Association, 2011. 3
33. Nicolas Meloni. New point addition formulae for ECC applications. In Claude Carlet and Berk Sunar, editors, *WAIFI*, volume 4547 of *Lecture Notes in Computer Science*, pages 189–201. Springer, 2007. 26
34. Peter L. Montgomery. Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of Computation*, 48(177):243–264, 1987. 25
35. K. Nohl, D. Evans, Starbug, and H. Plotz. Reverse-engineering a cryptographic RFID tag. In *17th USENIX Security Symposium*, pages 185–194. USENIX, 2008. 2
36. Christian Paquin. U-prove cryptographic specification v1.1 (revision 3). Technical report, Microsoft Research, 2013. 8
37. Jin Park, Jeong-Tae Hwang, and Young-Chul Kim. Fpga and asic implementation of ecc processor for security on medical embedded system. In *ICITA (2)*, pages 547–551. IEEE Computer Society, 2005. 2
38. Raluca A. Popa, Hari Balakrishnan, and Andrew J. Blumberg. Vpriv: Protecting privacy in location-based vehicular services. In *USENIX Security Symposium*, pages 335–350. USENIX Association, 2009. 3
39. Certicom Research. Standards for Efficient Cryptography – SEC 2: Recommended Elliptic Curve Domain Parameters. Available at `http://www.secg.org/collateral/sec2_final.pdf`, 2000. Version 1.0. 25
40. P. F. Riley. The tolls of privacy: An underestimated roadblock for electronic toll collection usage. *Computer Law & Security Report*, 24(6):521–528, 2008. 2
41. Ahmad-Reza Sadeghi, Ivan Visconti, and Christian Wachsmann. User privacy in transport systems based on rfid e-tickets. In Claudio Bettini, Sushil Jajodia, Pierangela Samarati, and Xiaoyang Sean Wang, editors, *PiLBA*, volume 397 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008. 3
42. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989. 7
43. I. J. Schur. Zur additiven Zahlentheorie. *Sitzungsberichte Preussische Akad. Wiss.*, pages 488–495, 1926. 13
44. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997. 6, 34
45. J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, second edition, 2003. 38
46. C. Hager (WBZ). Divorce lawyers using fast lane to track cheaters. `http://msl1.mit.edu/furdlog/docs/2007-08-10_wbz_fastlane_tracking.pdf`, 2007. 2
47. Hong Zhang, Jeremy Gummeson, Benjamin Ransford, and Kevin Fu. Moo: A Batteryless Computational RFID and Sensing Platform. `https://web.cs.umass.edu/publication/docs/2011/UM-CS-2011-020.pdf`, 2011. 5, 25

A Redeeming a Foreign RT implies Solving the DL Problem

Lemma 4. *Let* $(G, G_T, h, p, e, d) \leftarrow \mathsf{KGenRT}(1^k)$*,* $W \subset \mathbb{Z}_{p-1}$*, and* $(h^{d^w})_{w\in W}$ *be given. Let* $\mathcal{O}_G$ *be a challenge oracle, which when queried, returns a random* $\mathsf{SN}_{\mathsf{RT}} \leftarrow G$ *and* $\mathcal{O}_d$ *be an oracle which on input* (RT, w)*, where* $w \in W$ *and* RT *may either be one of the values returned by* $\mathcal{O}_G$ *or a previous output of* $\mathcal{O}_d$ *itself, returns* RT^r*,* $\mathsf{RT}^{d^w r}$ *for some fresh* $r \leftarrow \mathbb{Z}_p^*$*. Assume there exists an adversary* $\mathcal{A}$ *that in time* t *and with probability* ϵ *outputs* $(\mathsf{SN}_{\mathsf{RT}}, v, R, \mathsf{RT}) \leftarrow \mathcal{A}^{\mathcal{O}_d, \mathcal{O}_G}(G, G_T, h, p, e, W, (h^{d^w})_{w\in W})$ *such that*

$$e(h, \mathsf{RT}) = e(h^{d^v}, \mathsf{SN}_{\mathsf{RT}}^R)\,,$$

with $0 < v < p-1$*,* $R \in \mathbb{Z}_p$*, and* $\mathsf{SN}_{\mathsf{RT}}$ *and* RT *are previous outputs of* $\mathcal{O}_G$ *and* $\mathcal{O}_d$*, respectively. Then there exists an algorithm* $\mathcal{B}$ *which solves the DL problem over* G *in time* $t' \approx t$ *and with success probability* $\epsilon' \approx \epsilon$*.*

Proof. Let $(G, G_T, h, p, e, d) \leftarrow \mathsf{KGenRT}(1^k)$ be the output of the generator which is executed by $\mathcal{B}$. Let $g, g^x \in G$, where $x \leftarrow \mathbb{Z}_p^*$, be an instance of the DL problem given as input to $\mathcal{B}$. The algorithm embeds a randomized instance of this problem into each output $\mathsf{SN}_{\mathsf{RT}}$ of $\mathcal{O}_G$ and $\mathsf{SN}_{\mathsf{RT}}^r$, $\mathsf{SN}_{\mathsf{RT}}^{d^w r}$ of $\mathcal{O}_d$.

More precisely, $\mathcal{B}$ computes $(h^{d^w})_{w \in W}$ to setup a problem instance for $\mathcal{A}$. Note that this is possible as $\mathcal{B}$ generated d itself. Also, the algorithm $\mathcal{B}$ simulates the two oracles: When $\mathcal{O}_G$ is queried, it returns g^y for some fresh $y \leftarrow \mathbb{Z}_p$. On input (g^y, w) for $\mathcal{O}_G$, the algorithm $\mathcal{B}$ returns $((g^x)^{yz}, (g^x)^{yzd^w}$ for some fresh $z \leftarrow \mathbb{Z}_p^*$. Note that xz is uniformly distributed over $\mathbb{Z}_p^*$ and forms the value r chosen by the original oracle. For a different form of input to $\mathcal{O}_G$, the algorithm behaves exactly like the original oracle.

Let us now consider an output $(\mathsf{SN}_{\mathsf{RT}}, \mathsf{RT}, v, R)$ of $\mathcal{A}$ satisfying the equation

$$
\begin{aligned}
& e(h, \mathsf{RT}) = e(h^{d^v}, \mathsf{SN}_{\mathsf{RT}}^R) \\
\Leftrightarrow\ & e(h, (\mathsf{SN}'_{\mathsf{RT}})^{d^{v'} R'}) = e(h^{d^v}, \mathsf{SN}_{\mathsf{RT}}^R) \\
\Leftrightarrow\ & e(h, (g^{y'})^{d^{v'} R'}) = e(h^{d^v}, (g^y)^R) \\
\Leftrightarrow\ & e(h, g)^{y' d^{v'} R'} = e(h, g)^{y d^v R} \\
\Leftrightarrow\ & y' d^{v'} R' \equiv y d^v R \bmod p
\end{aligned}
$$

As $\mathcal{B}$ simulates the oracles and RT as well $\mathsf{SN}_{\mathsf{RT}}$ are outputs by those oracles, the algorithm also knows d, y, y', v', and the value R'' in $R' = xR''$. Hence the discrete logarithm x can be computed as $x = (y')^{-1} y d^{v - v'} (R'')^{-1} R \bmod p$, where the right-hand side of this equation is defined with overwhelming probability.

B Σ-Incremental DH in the Semi-Generic Group Model

In this section we analyze the hardness of our new problem in the Semi-Generic Group Model (SGGM). This model has been proposed in [27] as a replacement for the Generic Group Model (GGM) [44] to analyze problems in pairing-based settings. In the SGGM the (elliptic curve) group G is modeled as a generic group, while the target group G_T is given in the standard model, i.e., algorithms may perform any computation over G_T that is possible (in the subgroup of a finite field). Compared to the GGM, the SGGM is closer to the standard model and thus provides stronger evidence towards hardness assumptions in pairing-based cryptography. We are able to reduce the Σ-Incremental DH assumption in this model to a variant of the DL assumption over the target group. Note that this result also implies the hardness of Σ-Incremental DH in the GGM since the considered variant of DL is intractable if we model the target group as generic group.

In the SGGM, an algorithm $\mathcal{A}$ interacts with a *semi-generic group oracle* $\mathcal{O}$, which computes the group operation, evaluates the pairing, and in our case also generates signatures, on behalf of $\mathcal{A}$. The oracle $\mathcal{O}$ receives as input an instance of the problem, i.e., $(h^{d^i})_{0 \le i \le n}$, as well as the secret signature key d. It maintains a list $\mathcal{E} \subseteq G$, with $\mathcal{E}_j$ denoting the j-th entry of the list, which is initialized with $(h^{d^i})_{0 \le i \le n}$. By $[a]$ we denote the smallest index j (also called encoding) such that $\mathcal{E}_j = a$.[8] The index $[a]$ is undefined, if $a \notin \mathcal{E}$. We may always

[8] As in the original paper, we base the formalization of the SGGM on Maurer's deterministic technique for encoding elements [29], but our proofs can be adapted to Shoup's random encoding technique [44] as well.

assume that semi-generic algorithms only provide defined indices as input to the oracle. During initialization of the list, the corresponding indices pointing to the contained elements are sent to the algorithm. The oracle implements the following procedures, which may be called by $\mathcal{A}$:

- `GroupOp`$([a],[b])$: This procedure takes as input two indices $[a],[b]$, determines the group elements $a, b \in G$ looking into the list $\mathcal{E}$, computes $c = a \cdot b \in G$, appends c to $\mathcal{E}$, and returns $[c]$.
- `BilinearMap`$([a],[b])$: This procedure takes as input two indices $[a],[b]$. It determines the corresponding group elements $a, b \in G$ and returns $e(a, b)$ in the standard representation of G (i.e., as finite field element).
- `Rand`(): This procedure samples some random $r \in \mathbb{Z}_p$, adds h^r to $\mathcal{E}$, and returns $[h^r]$.
- `GetSig`$([a])$: This procedure takes as input an index $[a]$, determines the group element $a \in G$, computes the signature a^d, adds it to $\mathcal{E}$, and sends $[a^d]$ to the algorithm.

After interacting with the oracle, $\mathcal{A}$ will eventually output m encodings $[c_1], \ldots, [c_m]$ and integers $v_1, \ldots, v_m \in \mathbb{Z}_p$, where m equals the number of calls to `Rand`(). Let $h^{r_1}, \ldots, h^{r_m}$ denote the m outputs of `Rand`() and u the number of calls to `GetSig`$(\cdot)$. It wins if $c_i = h^{r_i d^{v_i}}$ for all $i = 1, \ldots, m$ and $v_1 + \cdots + v_m > u$.

Theorem 3 describes our hardness result in the model specified above. Note that the reduction is efficient as long as n as well as the exponents v_i chosen by $\mathcal{A}$ are relatively small (i.e., polynomially bounded in $\log p$). In our refund scheme, this can be satisfied by bounding n (the maximal single-trip refund) and v_i (i.e., the total refund amounts) by a constant which is small compared to p. Admittedly, the reduction is not very tight and might be improvable. On the other side, it might be inevitable for a *lightweight* public key system to build on a potentially easier problem.

Theorem 3. *Suppose there exists a semi-generic group algorithm $\mathcal{A}$ solving the n-Σ-Incremental DH problem in time t, where $\mathcal{A}$ issues $u' \geq 0$ queries to* `GetSig` *and $m \geq 1$ queries to* `Rand`*, and with success probability ϵ. Let w' be an upper bound on the integers v_i, $\mathcal{A}$ outputs, and set $u := u' + n$. Then there exists an algorithm $\mathcal{B}$ solving the $2u$-DL problem over G_T in time $t' \approx t + \tilde{O}(w \log p)$, where $w = \max(w', u)$, and with success probability $\epsilon' \geq \frac{\epsilon}{m}$.*

In order to prove the theorem above, we need the following two lemmas. Essentially, we need the first lemma to show that Σ-incremental DH is a non-trivial problem in the sense that there is no algorithm that solves the problem with non-negligible probability without "looking" at the specific problem instance. The second lemma is used to show that a solution to a non-trivial CDH type problem (in the SGGM) reveals a discrete logarithm.

For the first lemma we make use of the definition of straight-line programs over a polynomial ring. Informally speaking, such a straight line program is a fixed sequence of operations on its inputs without branching or looping. In our case straight-line programs are restricted to add polynomials and multiply them by a fixed variable.

Definition 6. *A (k_1, k_2)-*restricted straight-line program (RSLP) *$\mathcal{S}$ over $\mathbb{Z}_p$ in inputs X_1, ..., X_ℓ is a sequence of polynomials $P_{-\ell}, \ldots, P_0, P_1, \ldots, P_{k_1+k_2} \in \mathbb{Z}_p[X_1, \ldots, X_\ell]$, where $P_{-\ell} = X_\ell$, ..., $P_1 = X_1$, $P_0 = 1$ and for all $1 \leq i \leq k_1 + k_2$ we have that $P_i = P_j \pm P_k$ or $P_i = P_j \cdot P_{-\ell}$ for some $j, k < i$. Moreover, the operations $\pm$ and $\cdot$ have been applied in this sequence k_1 and k_2 times, respectively.*

Lemma 5 can be shown by induction over k_2.

Lemma 5. *Let $k_1, k_2, \ell \in \mathbb{N}$ and $\mathcal{S}$ be a (k_1, k_2)-RSLP in inputs $X_1, \ldots, X_\ell$. For $i = 1, \ldots, \ell - 1$ let Q_i denote a monomial of the RSLP of form $Q_i = X_i X_\ell^{e_i} \in \mathbb{Z}_p[X_1, \ldots, X_\ell]$, where $0 \leq e_i \leq k_2$, whose degree e_i is maximal with respect to all monomials of this form. Then it holds that $e_1 + \cdots + e_{\ell-1} \leq k_2$.*

Lemma 6 has been implicitly used and proven as part of the proof of Theorem 3 in [27].

Lemma 6. *Let $P \in \mathbb{Z}_p[X_1, \ldots, X_\ell]$ be a non-zero polynomial and $x_1, \ldots, x_\ell \in \mathbb{Z}_p$ be elements such that for some $s \in \{1, \ldots, \ell\}$ holds that $Q := P(x_1, \ldots, x_{s-1}) \not\equiv 0 \bmod p$ and $P(x_1, \ldots, x_s) \equiv 0 \bmod p$. Furthermore, let $\mathbb{M}$ denote the set of all monomials of $Q \in \mathbb{Z}_p[X_s, \ldots, X_\ell]$. Then there is at least one non-zero monomial of the form $aX_s^{e_s} M \in \mathbb{M}$, where $e_s > 0$ and $M = X_{s+1}^{e_{s+1}} \cdots X_\ell^{e_\ell}$ for some $e_{s+1}, \ldots, e_\ell \geq 0$. Let $aX_s^{e_s} M^*$ be an arbitrary but fixed monomial of this form. Then the univariate polynomial*

$$\sum_{a' X_s^{e'_s} M^* \in \mathbb{M}} a' X_s^{e'_s} \in \mathbb{Z}_p[X_s]$$

is non-zero and x_s is one of its roots.

Proof (Theorem 3). Given an instance of the $2u$-DL problem, $\mathcal{B}$'s task is to setup an instance of the n-Σ-incremental DH problem in the semi-generic model in a way that it can leverage a solution to the latter computed by $\mathcal{A}$ to solve the $2u$-DL instance. In particular, $\mathcal{B}$ will play the role of the semi-generic oracle. Since $\mathcal{A}$ is "blind" with respect to the internal details of G, G_T, e, `Rand`, and `GetSig`, we set $G := G_T, e(h, h) := h$ and simulate the aforementioned operations on this structure.

We will now describe our reduction algorithm $\mathcal{B}$: $\mathcal{B}$ takes as input an instance $a_0 = h, a_1 = h^x, a_2 = h^{x^2}, \ldots, a_{2u} = h^{x^{2u}}$ of the $2u$-DL problem over G_T. It then chooses $i^* \leftarrow \{1, \ldots, m+1\}$. If $i^* = m+1$, then the unknown DL x is treated as the private key d of the signature scheme, else x is used as the random choice r_{i^*} (see `Rand` below). Furthermore, $\mathcal{B}$ chooses $r_1, \ldots, r_{i^*-1}, r_{i^*+1}, \ldots, r_{m+1} \leftarrow \mathbb{Z}_p^*$ where d is set to r_{m+1} if $i^* \neq m+1$.

$\mathcal{B}$ sets up an n-Σ-incremental DH problem instance and simulates the semi-generic group oracle $\mathcal{O}$ for $\mathcal{A}$ as follows:

- The internal list $\mathcal{E}$ is initialized with $(h^{d^i})_{0 \leq i \leq n}$. The corresponding encodings are sent out to $\mathcal{A}$.
- `GroupOp` can be simulated since $\mathcal{B}$ knows how to perform the group operation on $G = G_T$.
- `Rand` is simulated as follows: Let i denote the number of calls to `Rand` so far. Then $h^{r_{i+1}}$ is added to $\mathcal{E}$ and $[h^{r_{i+1}}]$ is sent out.
- `GetSig`$([b])$ is simulated as follows: Let $k \leq u'$ denote the number of times this operation has been called so far. Let us consider the case $i^* = m+1$ first. In this case, the input can be written as $b = \prod_{i=0}^{k+n} (h^{z_i})^{d^i}$, where $d = x$ is unknown but the parameters n, k, and z_i are known to $\mathcal{B}$. Since $\mathcal{B}$ is given h^{d^i} for $1 \leq i \leq n+k+1$ (as part of the $2u$-DL instance), the signature on b can be computed as $\prod_{i=0}^{k+n} (h^{d^{i+1}})^{z_i} = \prod_{i=0}^{k+n} a_{i+1}^{z_i}$. Let us now consider the case $i^* \neq m+1$. In this case the secret key $d = r_{i^*}$ has been chosen by $\mathcal{B}$ and thus it can sign b as usual.

– `BilinearMap`$([b],[c])$ can also be simulated perfectly: Let k denote again the number of `GetSig` operations so far. First, we consider the case $i^* = m+1$. Here one can easily show that

$$e(b,c) = e(h^{\sum_{i=0}^{k+n} z_i d^i}, h^{\sum_{j=0}^{k+n} z'_j d^j}) = \prod_{i=0}^{k+n}\prod_{j=0}^{k+n}(h^{d^{i+j}})^{z_i z'_j} = \prod_{i=0}^{k+n}\prod_{j=0}^{k+n} a_{i+j}^{z_i z'_j}$$

where z_i and z'_j are known to $\mathcal{B}$. In other words, by knowing $h^{d^{i+j}}$ for all $0 \le i,j \le k+n$ and $0 \le k \le u'$ one can compute the pairing for *all* elements provided by $\mathcal{A}$. Since $\mathcal{B}$ is given a $2u$-DL instance (where $u = u'+n$) this is ensured. It remains to consider the case $i^* \neq m+1$, i.e., $r_{i^*} = x$. Here we can write

$$e(b,c) = e(h^{z_0 + z_1 r_{i^*}}, h^{z'_0 + z'_1 r_{i^*}}) = h^{z_0 z'_0}(h^{r_{i^*}})^{z_0 z'_1 + z'_0 z_1}(h^{r_{i^*}^2})^{z_1 z'_1} = a_0^{z_0 z'_0} a_1^{z_0 z'_1 + z'_0 z_1} a_2^{z_1 z'_1}$$

for unknown r_{i^*} and known z_0, z'_0, z_1, z'_1. So again we can compute the output of e since, at least, we know a_0, a_1, a_2.

Obviously, in this way $\mathcal{B}$ has set up a proper instance of Σ-incremental DH in the SGGM and is able to perfectly simulate the oracle for this instance.

How can $\mathcal{A}$'s output be leveraged? Let us assume that $\mathcal{A}$ has done m queries to `Rand` during its run and let $x_i = h^{r_i}$ denote the corresponding random elements. Then the algorithm eventually outputs m encodings $[c_1], \ldots, [c_m]$ and integers $v_1, \ldots, v_m \in \mathbb{Z}_p$. With probability ϵ, we have $c_i = x_i^{d^{v_i}} = h^{r_i d^{v_i}}$ for all $i = 1, \ldots, m$ and $v_1 + \cdots + v_m > u'$. Note that every element c_i computable by $\mathcal{A}$ can be written as $c_i = h^{P_i(r_1,\ldots,r_m,d)}$, where

$$P_i = \sum_{j=0}^{u} a_j D^j + \sum_{j=0}^{u'}\sum_{\ell=1}^{m} a_{j,\ell} R_\ell D^j \in \mathbb{Z}_p[R_1, \ldots, R_m, D]$$

is a polynomial known by $\mathcal{B}$. So in other words, $\mathcal{A}$ wins if

$$P_i(r_1, \ldots, r_m, d) = (R_i D^{v_i})(r_1, \ldots, r_m, d) \bmod p$$

for $i = 1, \ldots, m$ and $v_1 + \cdots + v_m > u'$. From Lemma 5 we know that for at least one i it holds that $P_i \not\equiv R_i D^{v_i} \bmod p$. Clearly, it is easy for $\mathcal{B}$ to identify such a a polynomial.

To summarize, with probability ϵ, $\mathcal{B}$ obtains a polynomial $\Delta := P_i - R_i D^{v_i} \in \mathbb{Z}_p[R_1, \ldots, R_m, D]$ such that $\Delta \not\equiv 0 \bmod p$ and $\Delta(r_1, \ldots, r_m, d) \equiv 0 \bmod p$. This can be split into disjoint events $\mathcal{E}_1, \ldots, \mathcal{E}_{m+1}$, where for $1 \le j \le m$ $\mathcal{E}_j$ is defined by

$$\Delta(r_1, \ldots, r_{j-1}) \not\equiv 0 \text{ and } \Delta(r_1, \ldots, r_j) \equiv 0$$

and $\mathcal{E}_{m+1}$ is defined by

$$\Delta(r_1, \ldots, r_m) \not\equiv 0 \text{ and } \Delta(r_1, \ldots, r_m, d) \equiv 0.$$

Denoting the probability of event $\mathcal{E}_j$ by α_j, it holds that $\epsilon = \alpha_1 + \cdots + \alpha_{m+1}$. Now assume that event E_{i^*} occurs, which happens with probability ϵ/m. In this case we know from Lemma 6 that we can extract an univariate polynomial Δ' from Δ such that x is a root of this polynomial. The coefficients of this polynomial can be easily computed since the coefficients of Δ are known and $r_1, \ldots, r_{i^*-1}$ have been chosen by $\mathcal{B}$. By applying an efficient standard

algorithm for computing roots of polynomials over $\mathbb{Z}_p$, such as [45, Algorithm 14.15], $\mathcal{B}$ can find the wanted DL x by computing all roots of the polynomial Δ'. These are at most $w = \max(w', u)$ different roots which can be computed in time $\tilde{O}(w \log p)$ [45, Corollary 14.16]. Whether a root x' equals x can be tested by verifying $h^{x'} \stackrel{?}{=} a_1$. □

C Overall Security Argument

In order to prove Theorem 1 we basically need to show that an adversary, $\mathcal{A}$, as defined in the TA security definition cannot achieve to get more reimbursements than the cost of TATs withdrawn minus the real trip costs (unless a double spending has been detected).

Let $\mathfrak{R}_1, \ldots, \mathfrak{R}_k$ denote the claimed and granted refunds, $\mathfrak{R}'_1, \ldots, \mathfrak{R}'_{k'}$ denote the issued single-trip refunds, let ℓ be the number of bought TATs, $\mathfrak{U}$ the cost of a TAT, m the real number of trips, and $\mathfrak{F}_1, \ldots, \mathfrak{F}_m$ the authentic fares of these trips. Then we need to prove that: $m \leq \ell$ (i.e., the adversary cannot do more trips than the number of purchased TATs) and

$$\sum_{i=1}^{k} \mathfrak{R}_i \leq \ell\mathfrak{U} - \sum_{j=1}^{m} \mathfrak{F}_j.$$

Since we assume that in each of the m trips the protocols at the exit turnstiles have been executed, a fare $\mathfrak{F}'_i$ and a corresponding refund $\mathfrak{R}'_i = \mathfrak{U} - \mathfrak{F}'_i$ has been calculated for each of these trips (i.e., $k' = m$). From Lemma 3 we know that the RT system guarantees that

$$\sum_{i=1}^{k} \mathfrak{R}_i \leq \sum_{j=1}^{m} \mathfrak{R}'_j = \sum_{j=1}^{m} \mathfrak{U} - \mathfrak{F}'_j$$

or, in other words $\mathcal{A}$ cannot cheat by claiming higher values on the RTs that he collected (else he would have to break the $\sum$-incremental DH assumption). Then, Lemma 2 yields that unless an RCT double spending has been detected

$$\sum_{i=1}^{k} \mathfrak{R}_i \leq \sum_{j=1}^{m} \mathfrak{U} - \mathfrak{F}'_j = \sum_{j=1}^{m} \mathfrak{U} - \mathfrak{F}_j.$$

Thus, it is impossible for an adversary to create fake RCTs and use them to collect refunds (or else he would be breaking unforgeability of the MAC scheme).

Finally, from Lemma 1 we know that unless a TAT double spending has been detected it holds that $m \leq \ell$ and thus

$$\sum_{i=1}^{k} \mathfrak{R}_i \leq \sum_{j=1}^{m} \mathfrak{U} - \mathfrak{F}'_j = \sum_{j=1}^{m} \mathfrak{U} - \mathfrak{F}_j \leq \ell\mathfrak{U} - \sum_{j=1}^{m} \mathfrak{F}_j. \square$$