Privacy-Preserving Billing for e-Ticketing Systems in Public Transportation

Florian Kerschbaum
SAP
Karlsruhe, Germany
florian.kerschbaum@sap.com

Hoon Wei Lim
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
hoonwei@ntu.edu.sg

Ivan Gudymenko
Faculty of Computer Science
Technische Universität Dresden, Germany
ivan.gudymenko@mailbox.tu-dresden.de

Abstract

Many electronic ticketing systems for public transportation have been deployed around the world. Using the example of Singapore's EZ-Link system we show that it is easy to invade a traveller's privacy and obtain his travel records in a real-world system. Then we propose encrypted bill processing of the travel records preventing any kind of privacy breach. Clear advantages of using bill processing instead of electronic cash are the possibility of privacy-preserving data mining analyses by the transportation company and monthly billing entailing a tighter customer relation and advanced tariffs. Moreover, we provide an implementation to demonstrate the feasibility of our solution.

1 Introduction

On the one hand, electronic tickets provide customers with many benefits in public transportation systems, such as ease-of-use, flexible tariffs and also monthly billing. On the other hand, electronic tickets are unique identifiers and make different travel records linkable. Public transportation companies obtain personalized travel records for each commuter which represents a significant privacy invasion.

In this paper, we analyze this privacy threat and propose a solution to it. First, we show how easy it is for an adversary to obtain these travel records in the Singapore EZ-Link system. The main problem is that the travel records are stored in clear in the back-end system and revealed to the customer without sufficient authentication. We emphasize that the EZ-Link system is currently deployed in the real world and its weaknesses undeniably demonstrate the need for better privacy protection. Second, we propose a full privacy-preserving solution where the travel records are processed only in an encrypted form by the transportation company. This solution proposal requires (partially) homomorphic, public-key cryptography, but remedies the privacy problem by only revealing unlinkable (encrypted) identifiers of the customers. We evaluate these computational requirements using a prototypical implementation showing that privacy is expensive, but achievable.

We highlight that our full-privacy solution supports monthly billing. Unlike the pay-upfront systems this provides the transportation company with a tighter customer relation and the customer with more flexibility regarding tariffs, but also requires processing the bill at the transportation company which poses the main privacy problem. Previous proposals for privacy in public transportation [30] only support pay-upfront using electronic cash. Proposals for privacy-preserving road toll pricing [8, 37] or electric vehicle charging [34] also allow monthly billing (with fixed fares), but have significantly higher requirements for the on-board unit than is implementable on an electronic ticket. Furthermore, our solution seems to be the first

to offer a win-win situation between commuters and the transportation company. Not only does our solution protect user privacy, but also allow privacy-preserving data mining by the transportation company. This way, we do not require the the transportation company to give up valuable aggregate information (related to commuters' travel history) that can be used for analyzing traffic patterns, detecting fraudulent transactions, and generally, improving their services.

In summary, our paper presents the following results:

- We provide a *privacy analysis of the Singapore EZ-Link* system. In our analysis, we focus on online (Internet-based) top-up services recently launched by EZ-Link. Particularly, we demonstrate, using freely downloadable traffic analyzers and a relatively cheap contactless card reader, how one can completely recover a commuter's recent travel records without the commuter's knowledge. We also highlight some web security concerns related to certificate management and a man-in-the-middle attack.

- We propose a *full privacy-preserving solution* based on partially homomorphic encryption. Our solution includes (i) a card authentication protocol relying on zero-knowledge proof (ZKP), and (ii) a bill computation protocol that uses Paillier's homomorphic encryption on bit vectors. We note that our authentication protocol is a new variant of Schnorr's identification scheme making use of encryption as a homomorphic one-way function, instead of the standard discrete log based one-way function. Further, we elaborate on how privacy-preserving data mining can be performed on encrypted travel records. To the best of our knowledge, this could not be achieved by previous proposals on privacy-preserving e-ticketing systems.

- We give a *feasibility evaluation* of our full privacy-preserving solution. In particular, we describe our implementation of the authentication protocol and the bill computation protocol. Our implementation result shows that the front-end (card-side) computational overhead of the authentication protocol is comparable to that of EZ-Link; while the back-end processing of encrypted travel records, as expected, is costly. However, since the back-end server can perform its task off-line (monthly billing), and through parallelization, we show that our solution is feasible.

The remainder of the paper is structured as follows. The next section presents our analysis of the Singapore EZ-Link system. In Section 3 we describe our full privacy-preserving solution before we summarize our evaluation results in Section 4. We review related work in Section 5. Lastly, Section 6 presents our conclusions and highlights some open problems.

2 Case Study: EZ-Link

Singapore is known for its highly efficient public transportation system, incorporating modern and world-class Mass Rapid Transit (MRT) train and bus services. Since 2002, commuters can pay for the fares for both MRT trains and buses in a convenient way using contactless, tap-and-go smart cards called *EZ-Link*. This is analogous to, for example, the *Calypso* card [11] deployed in some major cities in Europe and North & South America, the *Oyster* card [53] in London, and the *Octopus* card [38] in Hong Kong. In addition to all public buses and MRT trains, the EZ-Link card can now be used for Electronic Road Pricing (ERP) & Electronic Parking System (EPS) [32], printing & photo-copying services; as well as to make payment at selected food & beverage outlets, shopping & retail outlets, vending machines, and so forth.

Currently there are approximately over 8 millions of EZ-Link cards in circulation [26]. Each card can store up to 500 SGD (approx. 300 EUR or 390 USD) and has a lifespan of 5 years from the date it was first issued [24]. In line with the emergence of NFC-supported[1] mobile phones as digital wallets, for example Google Wallet and Isis Digital Wallet, EZ-Link is currently conducting tests on selected models of NFC-enabled smart phones in preparation for rolling out a new alternative to EZ-Link cards.

[1]Near Field Communication (NFC) is a short range wireless communication technology that provides intuitive and simple two-way data exchange between NFC-enabled electronic devices, such as mobile phones, digital cameras, kiosks and TVs.

In what follows, we focus on on-line top-up services recently offered by EZ-Link. Further details on technology and protocols employed by EZ-Link can be found in Appendix A.

2.1 Top-up

One purchases an EZ-Link card with a pre-paid stored value from any MRT station and selected bus interchanges. Subsequently card top-up (or reload) can be made at not only any general ticketing machine at all MRT stations, but also most convenience stores, post offices, and Automated Teller Machines (ATMs) or cash machines [24]. It seems that the high availability of top-up points has been a key factor in the widespread use of EZ-Link cards.

Recently, another top-up channel was launched—it is now possible that one performs top-up through the Internet using a portable (personal) reader that is connected to a PC. A similar top-up approach has also been made available by Octopus in Hong Kong [39]. One major benefit of such an approach is that it allows commuters to view and print their recent transaction records.

In the remaining section, we focus on the protocols underpinning two types of Internet-based EZ-Link top-up services, namely, *EZ-Online* and *Top & Tap*.

2.1.1 EZ-Link Online

This is an on-line service that allows commuters to view their card details, including the most recent 30 transactions (travel records), download discount coupons onto their cards, to make payment at selected on-line merchants, and more importantly, to reload their card credit [25].

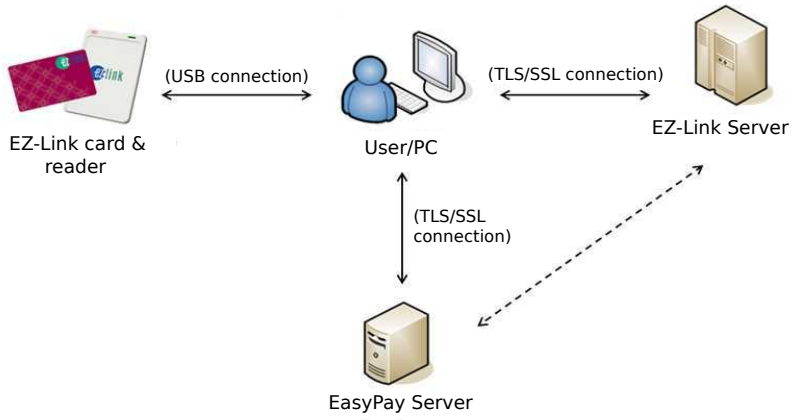

Figure 1: The EZ-Online service involving interactions between a card reader, a PC, the EZ-Link server and the EasyPay Server.

To use the EZ-Online service, one requires a CEPAS-compliant[2] EZ-Online card reader, which is sold at 39 SGD (approx. 24 EUR or 30 USD) per unit. The reader is then, through a USB cable, connected to a PC or laptop, which clearly needs to have access to the Internet. This way, the reader can interact with the EZ-Link server to retrieve the required information from a central database. Moreover, the commuter can make on-line payment through the EasyPay server using her credit/debit card. We illustrate this in Figure 1.

To start the service, one needs to go to the EZ-Online webpage and places her EZ-Link card on the reader. She is then prompted a window displaying information about her card (including card number, expiry date and remaining value in the card) and options to view her past transaction records and to top-up her card.

[2]CEPAS is a Singapore standard for contactless e-payment systems [51].

To view and/or to print up to the last 30 transaction records, one simply goes to the "Txn Record" tab. These records are retrieved from the EZ-Link server via a TLS/SSL secure channel. Each record contains information such as bus service number, time, date, and fare.

In order to top-up, one clicks on the "Top-Up Card" tap. The commuter is requested to select a top-up value. She is then redirected to the EasyPay webpage to make payment. Clearly, as with any typical secure on-line payment, an TLS/SSL connection is used between the PC and the EasyPay server to authenticate the server and to protect credit card details. Once the payment has been approved, the commuter is redirected back to the top-up window so that she can update her card's new balance.

2.1.2 Top & Tap

Alternatively, one can reload her card credit through the Internet using the Top & Tap service. As its name implies, the Top & Tap service comprises two parts. In the Top part, the commuter goes to the Top & Tap webpage, enters the 16-digit card application number (CAN) of the card that she wishes to top-up, selects a top-up amount for her card, and makes payment via EasyPay. In the Tap part, she then needs to update the balance on her card either using a portable EZ-Link reader or an AXS station.[3] However, unlike the EZ-Online service, one cannot view past transaction records.

2.2 Security and Privacy Challenges

While it is essential to provide a widely available, convenient top-up service to commuters in any pre-paid system, our study shows that it inherently introduces some security and privacy concerns, particularly when it is done via the Internet.

2.2.1 Leakage of Location Information

We examined if a CEPAS-compliant EZ-Link card leaks any sensitive or personal information related to the card owner by performing real top-up. We used the EZ-Online top-up protocol (as shown in Figure 1) since the associated protocol messages are easier to intercept as follows: we connected a portable EZ-Link card reader to a PC, placed a valid card on the reader, and performed top-up on the card's balance; we then used USBlyzer [55] to intercept real messages transmitted between the reader and the PC, and Wireshark [57] to intercept messages between the PC and the EZ-Link server. Both USBlyzer and Wireshark are traffic analyzers installed on the PC and we started the analyzers just before we placed the card on the reader.

Let `Top_Up` be the top-up value; `Rec_Short` denote a summary of past transaction records, *e.g.,* bus service number; and `Rec_Long` denote a more detailed past transaction records, *e.g.,* bus service number, time, date, and fare. The first three intercepted messages—between the reader (R) and the PC (P), and between the PC (P) and the EZ-Link server (S)—can then be illustrated as follows:[4]

$$
\begin{aligned}
&(1)\ R \rightarrow P : \texttt{CAN}, \texttt{Rec\_Short} \\
&(2)\ P \rightarrow S : \texttt{CAN}, \texttt{Rec\_Short} \\
&(3)\ S \rightarrow P : \texttt{CAN}, \texttt{Rec\_Long}, \texttt{Top\_Up} := 0 \\
&\qquad\vdots
\end{aligned}
$$

When a card is placed within the detectable proximity of the reader, the latter transmits the card application number (CAN) and a summary of transaction history stored on the card to the PC to which the reader is connected. This is illustrated in message (1). Subsequently, as shown in message (2), the PC simply forwards the message it received from the reader to the EZ-Link server. In message (3), the EZ-Link server, which has access to a database containing all transaction history of users, then returns a list of more detailed

[3] AXS stations are multi-application transactional terminals that are widely available in Singapore and operate 24 hours a day. They come with debit/credit card payment facilities and run on a high-speed ADSL broadband network.

[4] For simpler exposition, here we focus on only the first three messages that are relevant to our discussion.

transaction records associated with the submitted CAN. Also, the top-up value is initialized to 0 (this is just before a top-up value is chosen from the EZ-Link Online website). The rest of the protocol includes generation of a credit cryptogram by the server and a credit receipt record by the card. These are used for authentication and to prevent forgery, and are described in Appendix A.

A major privacy concern here is that the CAN and travel records associated with a card are transmitted in clear between the reader and the PC, and between the PC and the EZ-Link server. As a consequence, it is now possible to trace someone's travel records. One way to do it is to simply carry an EZ-Link reader and walk close enough to the target EZ-Link card, typically placed in a clothing pocket, wallet or handbag. If the reader (about the size of an iPhone) is properly hidden, for example in a hand pouch, and it is connected to a laptop with Internet access, it is then possible to recover the target's recent travel records without her knowledge in less than a second. Alternatively, one can purchase and customize a more powerful (longer range) off-the-shelf reader that is able to capture the CAN of any EZ-Link cards within the detectable proximity. This is analogous to wireless identity theft in which RFID-enabled credit cards can be scanned from a close proximity to obtain their owners' names, card numbers and expiration dates [29].

We do not know exactly why the card needs to send a summary of recent transactions to the server. However, we suspect that this may be related to fraud detection; for example, `Rec_Short` that the server received from a card can be used to "authenticate" the card in the sense that the data contained in `Rec_Short` should match that of the corresponding `Rec_Long` stored in the database. Otherwise, the card may be a cloned card with outdated travel records. Moreover, it seems that combining recent travel record information with a CAN makes a replay attack more challenging, assuming the travel records are updated frequently.

2.2.2 Other Findings

We also evaluated the Top & Tap service using a similar setup as illustrated in Figure 1, but without the card and the reader. Ironically, when the Top & Tap webpage was being accessed using Chromium version 25.0.1364.160, we were prompted a commonly seen, but somewhat unexpected, security warning: "The application's digital signature cannot be verified. Do you want to run the application?" When we clicked for more information, we saw: "The digital signature was generated with an untrusted certificate." According to Symantec [47], this behavior is due to the self signing of the application's certificate and the inability to check this certificate with an external certificate authority. We ignored the security warning and clicked "run". (Otherwise, we would not have been able to access the Top & Tap service.) A similar problem was encountered when we accessed the EZ-Online web page.

We intercepted all network traffic between the PC and the EZ-Link server using Wireshark. Unlike the EZ-Online service, no TLS/SSL connection is established between the PC and the EZ-Link server, and data is transmitted through TCP (rather than HTTP) on ports 55155 and 9999, respectively. However, TLSv1 is used to secure the communication between the PC and the EasyPay Server. Since the Top & Tap service is implemented using a Java applet, the web browser communicates with the server using serialized objects (instead of in the form of plaintext data typically used in a HTTP connection). We took a closer look at the Java applet using JavaSnoop [18]. We inspected how the Java applet behaves and processes data by inserting "hooks" at various Java class methods associated with the Top & Tap service. This way, we could intercept calls to the methods in real-time when a top-up is performed to see all the relevant parameters and values. Our findings show that the CAN is never authenticated nor encrypted, although the Java applet does perform a validity check on any entered CAN. Security is achieved through a reference number that binds the transaction with the CAN. The reference number, which comprises the 16-digit CAN and a 13-digit seemingly random number, was transmitted to the EasyPay server via a TLS-protected channel, and thus cannot be easily intercepted nor modified.

However, we remark that it can be dangerous to rely on users to check and appropriately act on security messages. When prompted warning dialogs, users tend to simply click through them without much hesitation. This opens up an opportunity for the attacker to install a rogue root certificate in the user's web browser through a malicious Java applet, as demonstrated in [4]. Consequently, this may cause considerable harm to the user and her PC. More seriously, when combining such injection of a fake certificate with a more sophisticated HTTPS hijacking (or stripping) technique [36], a man-in-the-middle attack is imminent. It is

now possible for the attacker to transparently hijack the TCP traffic between the user's PC and the EZ-Link server, such that the HTTPS link between the PC and the EasyPay server is redirected into a look-alike HTTPS link controlled by the attacker. Hence, any top-up performed by the user will then be effectively made to the attacker's choice of card.

On another note, by reverse engineering, we found out that the EZ-Online reader (shown in Figure 1) appears to be an ACR 122U NFC contactless smart card reader [1]. A closer check against technical specifications available from the Internet confirms that the reader does not perform any cryptographic operations, but rather simply functions as a device that relays messages between an EZ-Link card and a PC. This rules out any side-channel analysis aiming to extract secret keys stored on the reader.

2.3 Simple Fixes and Limitations

While EZ-Link cards offer convenience to commuters, it is unfortunate that they leak travel information, and thus, putting the commuters' location privacy at risk. To ensure privacy, it is clear that the travel records transmitted between the card/reader and the EZ-Link server needs to be protected. A trivial fix to this is to make use of a card-specific and password-derived key to protect the confidentiality of travel records. However, this is not a satisfactory approach because not only it introduces additional storage and protection requirements to the server, but also causes inconvenience to the commuters since they now need to remember a password.

A more challenging goal is that any transaction record should not be linked to a specific card, implying that a card should be authenticated to the server in an anonymous way. We note in passing that there exist low-cost (relying on symmetric key techniques) anonymous authentication protocols in the literature, for example Song and Mitchell's RFID protocol [50], that may be adopted to address our privacy concern. However, using such a solution provides only *partial* privacy protection because of the fundamental limitations of a symmetric key approach [30, 45]. Particularly privacy is achieved against a passive adversary but not a semi-honest or malicious insider, for example the transit company. Moreover, even though the symmetric key approach offers better efficiency, it is not clear how full privacy can be achieved while allowing the relevant parties to process transaction records. This motivates the need for a *full* privacy-preserving solution using techniques beyond symmetric key cryptography.

3 Privacy Preserving Billing

In order to provide full privacy at the transit station and in the back-end processing the travel records, we resort to an asymmetric key cryptographic approach. Our solution has additional computational cost, but combines an unlinkable pseudonym with the ability to provide regular billing and privacy-preserving data mining. We foresee that regular billing—instead of pre-paid fares—will provide additional benefits to the customer, such as ease-of-use or tailor-made tariffs. Furthermore, regular billing is implemented in several other networked technologies, such as cellular networks or road-toll pricing. Data mining enables the transportation company to analyze the (encrypted) travel records for, *e.g.*, network optimization and thereby realizing a major benefit of collecting them.

From a technical perspective it is hard to design a secure solution that implements regular billing and data mining without a back-end. The questions of who issues the bill and where are the records stored immediately arise. Nevertheless, it has already been shown by the privacy analysis of the Singapore EZ-Link card in Section 2.2.1 that storing travel records in the back-end poses a significant privacy problem. Our challenge is therefore to encrypt the transportation records, but still be able to process them in the back-end.

3.1 Prerequisites

We use an additively (*i.e.,* partially) homomorphic encryption scheme. In our implementation we used Paillier's encryption scheme [41]. Unlike fully homomorphic encryption [27] additively homomorphic encryption supports only addition as the homomorphic operation, but it is as efficient as public-key cryptography.

Paillier encryption operates over a finite field $\mathbb{Z}_N^*$ with modulus $N = pq$, where p, q are large primes. We denote $E(x, r)$ the encryption of plaintext x with randomization parameter r and $D(c)$ the decryption of ciphertext c to plaintext x and randomization parameter r:

$$D(E(x, r)) = x, r \mod N$$

The following homomorphic properties hold:

$$E(x, r)E(y, s) = E(x + y, rs) \mod N^2$$
$$E(x, r)^y = E(xy, r^y) \mod N^2.$$

In many cases the randomization parameter can be neglected and we write $E(x)$ or $D(c)$, respectively.

$$D(E(x)) = x \mod N$$

Paillier encryption is a semantically secure (IND-CPA) public-key encryption scheme. Semantic security implies that due to randomization an adversary cannot distinguish two ciphertexts, even if they are from the same plaintext.

We use the encryption scheme in a particular way. Except for the last step of processing we operate only on bits. Let r be a uniformly chosen, random number in $\mathbb{Z}_N$. The ciphertext for 1 is $E(0)$ and the ciphertext for 0 is $E(r)$. We denote $Enc(a)$ the encoding step and $Dec(x)$ the decoding step of decoding a random number to 0 and a 0 to 1.

By extending the technique of [46] we can now perform logical and operations on the ciphertexts. Let $E(x) = E(Enc(a))$ and $E(y) = E(Enc(b))$ be two encrypted bits. Then

$$a \wedge b = Dec(D(E(x)E(y))).$$

Note that this technique has a small probability of falsely decoding of what should be a 1 to a 0. The result of the addition $x + y$ is uniformly distributed in $\mathbb{Z}_N$ and $0 \in \mathbb{Z}_N$. Therefore the chance of this error is $2^{-\sigma}$ where σ is the bit length of the modulus (*i.e.,* linear in the security parameter).

Furthermore, we can perform equality comparison of integers. Let a and b be two integers in $\mathbb{Z}_N$. Then

$$(a == b) = Dec(D(E(a)E(b)^{-1})).$$

In order to process multiple ciphertexts using homomorphic encryption they need to be encrypted using the same key. It is necessary to highly safe-guard the private key. We entrust the private key of the encryption scheme to a trusted key-managing authority which could, *e.g.,* be implemented by a public department. We also show how to distribute this key-managing authority in Section 3.4.

This authority (or set of authorities) then offers an additional processing service: negation. On input of $E(0)$ it returns $E(1)$ and on input of $E(r)$ it returns $E(0)$. We denote this operation as $NEG(\cdot)$. Note that the recipient of the negated bit can easily randomize it by (homomorphically) multiplying it with a plaintext random number. Let $E(x) = E(Enc(a))$ be an encrypted bit. Then

$$\neg a = Dec(D(NEG(E(x)))).$$

Using the negation service, one can implement any functions on these bits. The operations NOT and AND (NAND) suffice to implement any functionality in a binary circuit. The back-end can therefore theoretically perform any operation on those (encrypted) bits. Nevertheless, for performance reasons it is necessary to minimize the number of invocations of the negation service, since it implies network communication and costly decryption. Our evaluation results confirm this design choice.

In the next section we present our solution for regular billing of transportation records, but we emphasize that the public transportation provider is not limited to this kind of processing.

3.2 Billing

Our main goal is to encrypt each travel record consisting of a traveller identifier, location and time. Since time can be inferred by the creation time of the record, it remains unencrypted. The traveller identifier is hidden using (randomized) encryption, such that any two encrypted identifiers are unlinkable. Also, the location is encrypted using randomized encryption and therefore unlinkable. This implements the strongest form of privacy of the entire travel record. The back-end cannot infer any information about the travel except that any has taken place (at this time).

We divide our algorithm in two steps: *Entry/Exit* and *Billing*. The entry and exit steps are performed when a traveller enters or exits the transportation system. They record (and encrypt) the information stored later in the travel record. The stored information is then further processed in the back-end during the billing phase. This back-end system computes the monthly bill.

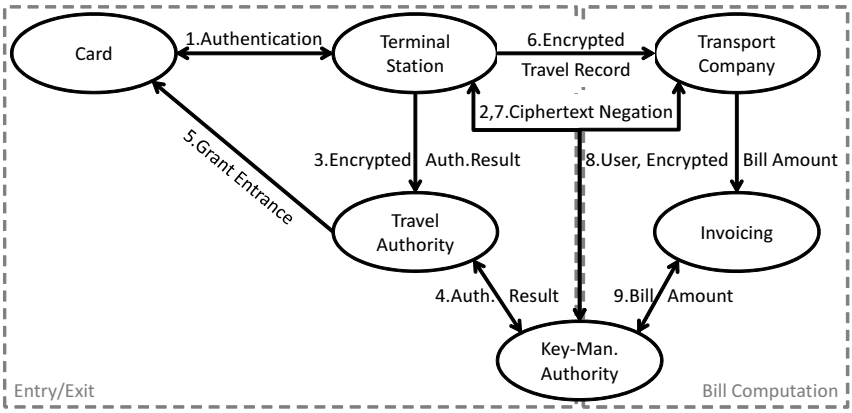

Figure 2: Full Privacy-Preserving Architecture

The flow of information between the individual parties is depicted in the architecture diagram of Figure 2. The computation of the billing amount is performed by the transportation company's back-end system.

3.2.1 Entry/Exit

We aim to protect the identifier of a traveller on the smart card, *i.e.,* the traveller's smart card carries a unique identifier v. Nevertheless, this identifier is not to be revealed in plaintext outside the card. The card will always encrypt the identifier using our Paillier encryption scheme (with the common public key). Therefore the card also carries the public key—but not the private key—of our Paillier encryption scheme.

On entry or exit, the card delivers a secret identifier $E(v)$. We emphasize again that the secret identifier is completely unlinkable due to randomized encryption. We explain the authentication of this identifier later.

On entry the card stores a bit on the card, such that it can only be used for exit next time. Similarly, on exit the card sets this bit, such that it can only be used for entry next time. If a card has a bit stored for entry (exit), it cannot be used to exit (enter). Furthermore, if the update of the bit fails or a false bit is presented an alarm should be raised and entry or exit should be denied. This prevents a fraudster from entering and exiting with two different cards—one for entry and one for exit.

$$
\begin{array}{lcl}
C \to S & : & E(r,s),\ E(v,t) \\
S \to C & : & a \\
C \to S & : & D(E(r,s)E(v,t)^a)) = D(E(r+va, st^a)) = (r+va, st^a)
\end{array}
$$

Figure 3: Card Authentication ZKP

The entry (or exit) station adds time and the encrypted location to the travel record. We represent the location as a bit vector. For each possible location l there is a corresponding bit l_i. Then on enter (or exit) we set the bit of this location to 1 and all others to 0. We encrypt this vector bit-wise for each location bit l_i. We use the bit encryption scheme, *i.e.,* $E(r)$ for 0 and $E(0)$ for 1, as described in Section 3.1.

Authentication The card needs to prove its validity on entry. The actual identity of the card needs to remain anonymous. We therefore use a zero-knowledge proof (ZKP).

The ZKP follows the Schnorr zero-knowledge proof of knowledge of discrete logarithm, but uses Paillier encryption instead of modular exponentiations. Let C be the card (prover), S be the server (verifier) and r, s, t and a be a uniformly chosen, random numbers in $\mathbb{Z}_N$. Our ZKP is depicted in Figure 3.

Note that the card does not have to perform any decryption and therefore does not need the private key. It can compute the random parameter st^a from its input random parameters to $E(r, s)$ and $E(v, t)$ and a. The server verifies that $E(r + av, st^a) = E(r, s)E(v, t)^a$.

The server needs to also verify that the identifier is still valid and the customer has paid its bills. A fraudster must not be able to enter with a modified card presenting an old (revoked) or even forged identifier v. The server obtains all possible identifers u_i for v and computes

$$\begin{aligned} E(c) &= E(\neg \bigwedge_i \neg(v == u_i)) \\ &= NEG(\prod_i NEG(E(v)E(u_i)^{-1})). \end{aligned}$$

The server sends $E(c)$ to a trusted third party which has it decrypted by the key-managing authority. If it is 1, then entry (or exit) is granted; if it is 0, entry (or exit) is denied.[5] The trusted third party can be the same as the key-managing authority, but also needs access to the transportation infrastructure in order to grant access. Therefore we assume that the trusted third party is implemented by a travel authority.

If the number of all identifiers is very large, then this computation can become very costly. In order to mitigate this, the anonymity of a traveller's identifier can be limited to k-anonymity by also transmitting a public identifier u along the with the (encrypted) secret identifier v. This public identifier should be k-anonymous and can be used to limit the number of identifiers to compare against.

3.2.2 Bill Computation

After the billing period there is a trail of records of the following (encrypted) form stored at the backend of a public transportation company:

$$R_i : time, secretidentifier = E(u), location = \vec{E}(l_i).$$

In order to compute a bill for identifier u^* we consider each (possible) entry record (R_e) and each (possible) exit record (R_x, $x > e$) and compute a matrix T for each possible combination of locations with entries

$$\begin{aligned} E(t_{i,j}) = E((R_e.u == u^*) \wedge R_e.l_i \wedge (R_x.u == u^*) \wedge \\ R_x.l_j \wedge \bigwedge_{k=e+1}^{x-1} \neg R_k.u == u^*) \\ = E(R.u)E(u^*)^{-1}E(R.l_i)E(R_x.u)E(u^*)^{-1} \\ E(R_x.l_j) \prod_{k=e+1}^{x-1} NEG(E(R_k.u)E(u^*)^{-1}). \end{aligned} \quad (1)$$

[5] Alternatively, we may also use accumulator-based revocation techniques [12, 14] with some efficiency trade-off between the client and the server.

The entry $t_{i,j}$ is true iff the customer u^* travelled from location l_i to location l_j. The matrix T consists of all 0 entries if the records R_e and R_x do not correspond to such a valid trip. We can limit the search for possible pairs of records (R_e and R_x) using the time provided in the record, *e.g.,* a maximum travel time of 90 minutes. In order not to miss a travel that exceed this time we can insert an "excess" record with a timestamp of the the current time plus maximum travel period when entering.

We maintain an array of fares from location l_i to location l_j. Let $f_{i,j}$ be the fare for this trip. We now use the negation service in a slightly different way. We no longer compute on bits, but instead on integers. First, we invoke the negation service on the encrypted $t_{i,j}$. Let $E(\neg t_{i,j}) = NEG(E(t_{i,j}))$ denote the result. Note that $E(\neg t_{i,j}) = E(1)$ if u^* took the trip and $E(\neg t_{i,j}) = E(0)$ if not. We compute

$$f_{R_e,R_x} = \prod_{i,j} E(\neg t_{i,j})^{f_{i,j}}.$$

The fare is $f_{R_e,R_x} = E(f_{i,j})$, if the records correspond to a valid trip of u^* and $f_{R_e,R_x} = E(0)$ if not. Finally, we compute the (encrypted) monthly bill as

$$f_{u^*} = \prod_{R_e,R_x} f(R_e, R_x).$$

The identifier u^* and the encrypted billing amount f_{u^*} are sent to a billing service for invoicing.

3.3 Security and Privacy Analysis

Our billing protocol in Section 3.2.2 implements a secure computation. Nevertheless we do not resort to the standard semi-honest or malicious security model of Goldreich [28]. Instead, we prove (Theorem 1) that the transportation company cannot distinguish the travel records of any (k-anonymous) two user identifiers.

Theorem 1. *The transportation company cannot distinguish the user identifiers $E(R_i.u)$ and $E(R_j.u)$ of any two travel records R_i and R_j.*

Proof. The adversary (transportation company) is given the public key and its messages consist only of ciphertexts. The theorem is then a direct corollary of the IND-CPA security of Paillier encryption proven in [41]. Even if the adversary could choose two challenge plaintexts, it would not be able to distinguish them (except with negligible probability). □

Note that this theorem is stronger than semi-honest security, since the transportation company may deviate from the protocol and still the privacy of the user is not violated. Nevertheless, it is weaker than malicious security, since the correct computation of the billing amount is not guaranteed. We provide a similar Theorem 2 for the location information.

Theorem 2. *The transportation company cannot distinguish the locations $E(R_i.l_h)$ and $E(R_j.l_k)$ of any two travel records R_i and R_j.*

Proof. Similar to Theorem 1. □

We also prove the correct authentication of the card on entrance.

Theorem 3. *The card authentication zero-knowledge proof (Figure 3) is complete, sound and honest-verifier zero-knowledge.*

Proof. For completeness, observe that the verification condition $E(r + av, st^a) = E(r, s)E(v, t)^a$ holds, if the card provides the correct inputs.

We present a simulator of the protocol for its zero-knowledge property and an extractor for its soundness property. The simulator of the server's (verifier's) view without knowledge of the secret identifier proceeds as follows in reverse order of the protocol:

$S \to A_i$	:	$E(x) = E(Enc(a))$
$A_i \to \mathbb{A}_{\neg i}$	:	$E(r_i \xleftarrow{R} \mathbb{Z}_N)$
$\mathbb{A}$	:	$c = E(x \sum_i r_i) = E(x) \circ \prod_i E(r_i)$
$A_i \to \mathbb{A}_{\neg i}$	:	$[\![x \sum_i r_i]\!]_i = D_i(c)$
$A_i \to S$	:	$E(Enc(\neg a)) = E(\neg x \sum_i r_i)$

Figure 4: Distributed Negation Service

1. Simulate uniformly random values for $r + va$ and st^a.
2. Simulate a uniformly random value for $E(v,t)$ and compute $E(r,s) = E(r + va, st^a)(E(v,t)^a)^{-1}$.

The extractor of the card's (prover's) secret identifier v with black-box rewinding access to the card proceeds in the same way as Schnorr's scheme – namely as follows:

1. Let the card commit to $E(r,s)$, $E(v,t)$ and take a breakpoint.
2. Send challenge a_1 and record $r + va_1$.
3. Rewind the card to the breakpoint before sending the challenge.
4. Now, send the challenge a_2 ($\neq a_1$) and record $r + va_2$.
5. Compute $v = ((r + va_1) - (r + va_2))(a_1 - a_2)^{-1}$.

□

3.4 Distributing the Negation Service

Instead of Paillier's encryption scheme we can use Damgård and Jurik's generalization to threshold decryption [20]. Let n be the total number of authorities and $t-1$ the threshold of admissible colluding authorities. Let $[\![s]\!]_i$ denote a (t,n) Shamir's secret share [48] of a secret s.[6] Let d be the decryption key of the Damgård-Jurik encryption scheme. Initially the i-th authority obtains $[\![d]\!]_i$. With this share it can perform the following operation.

$$D_i(E(x)) = [\![x]\!]_i$$

A single party can still perform encryption using only the public key. From t secret shares $[\![x]\!]_i$ (for distinct i) the authorities can recover x.

It is now easy to distribute the negation service, but we intend to strengthen it against decryption oracle attacks. In a straightforward implementation any authority still obtains the plaintext. We therefore randomize the plaintext by multiplying it with a commonly chosen random number. This prevents the authorities from learning the plaintext, if it is not zero. We use the protocol by Cramer *et al.* [16] to multiply the plaintexts of two ciphertexts. We denote this protocol as $E(xy) \leftarrow E(x) \circ E(y)$.

Let A_i be the $i-th$ authority and $\mathbb{A}_{\neg i}$ be the set of all authorities except the i-th. We denote $\xleftarrow{R} \mathbb{Z}_N$ a uniformly random choice in $\mathbb{Z}_N$. The resulting protocol is depicted in Figure 4.

We are aware that we could further strengthen the protocol using Toft's bit decomposition [52], such that even in case of a zero plaintext no authority would learn the plaintext. We present these protocols as an alternative heightening the bar against decryption oracle attacks, but keeping performance impact low. Cramer *et al.*'s protocol [16] is quite resource-intensive and Toft's bit decomposition [52] requires an invocation per plaintext bit whereas we require only one invocation. Furthermore, our efficient version can also be used for decryption of the result (with very minor modification).

[6] We omit the modulus, since it is dependent on the parameters of the encryption scheme.

3.5 Data Mining

Next to billing our architecture allows the transportation company to perform data mining on its travel records in a privacy-preserving way. This data mining is a major reason for the transportation company to track users. Competing approaches to privacy-preserving e-Tickets in transportation [30, 45] based on anonymous credentials prevent the collection of the necessary data. Our approach does not only allow monthly billing, but also the analysis of travel records, *e.g.,* for network optimization.

Privacy-preserving data mining has long been subject to research, see *e.g.,* [2, 3, 33, 56] for an overview. This literature contains a number of protocols for data mining on encrypted data. These protocols are optimized in their choice and use of encryption for the specific data mining algorithms investigated. This ensures high efficiency of the protocols.

We emphasize that our mode of encrypted bill processing allows computation of arbitrary algorithms. Our mode of encryption (Section 3.1) using the negation service is generically applicable to all computations. Still, it favors some computations over others in terms of performance. In lack of a complete set of required analyses (some analyses may be performed on demand in an ad-hoc manner) we exemplarily present a common analysis that can be performed efficiently.

A transportation company may want to discover their most frequently used routes—maybe dependent on the time of the day. For this, it may want to compute a histogram of the used routes, *i.e.,* the number of times each route has been taken. Note that in Section 3.2.2 we compute the binary (encrypted) value $E(t_{i,j})$ for route from location l_i to l_j for each trip. Note also that this value is encoded as 0 or 1 and not a random number, since it is fresh from the negation service.

Let $E(t_{i,j,k})$ be this value for trip k. We compute $n_{i,j}$, i.e. the number of times the route from l_i to l_j has been taken as

$$E(n_{i,j}) = E(\sum_k t_{i,j,k}) = \prod_k E(t_{i,j,k})$$

The resulting ciphertext $E(n_{i,j})$ can be sent to the key-managing authorities for decryption. Nevertheless, a large set of data mining analyses may violate the privacy of users, even if performed semi-honestly. The output of these analyses may allow additional inferences. A provably secure method against these kind of inferences is differential privacy [19]. Some random noise with expected value 0 and variance depending on the sensitivity of the analysis to individual records needs to be added. The key managing authorities can simply choose the necessary random value and add it to the result before returning it to the transportation company.

This method ensures the privacy of the customers under all analyses performed semi-honestly, but it also enables the transportation company to reap most benefits from data collection.

4 Feasibility Study

To evaluate the practicality of our solution, we implemented our authentication and bill computation protocols. Since EZ-Link already supports retail payment through NFC-enabled smart phones [44, 17], which are soon likely to also be used in public transportation, we worked with an NFC-enabled phone instead of a contactless smart card in our implementation. However, we note that the server-side implementation of our solution is on-going work, and at the time of writing, we are not able to provide a full-fledged efficiency analysis involving the interactions between different entities as illustrated in Figure 2.

4.1 Front-end

In order to estimate the efficiency of the front-end of our solution, we implemented the card-side computations required for the authentication ZKP (shown in Figure 3) on an NFC-enabled smart phone. The model chosen for our tests is Samsung I9070 Galaxy S Advance NFC representing a middle-class smart phone with a 1 GHz dual core processor and installed operating system Android 2.3.6 Gingerbread (with API version 10). We

use a slightly optimized version of the Java implementation of Paillier's encryption scheme by Liu [35] with standard key length of 1024 bits.

According to our implementation, on average, a single Paillier homomorphic encryption operation takes 65 ms; while the computation time required to perform the authentication ZKP is 131.03 ms (without considering the server). This is comparable to the benchmark of 140 ms per transaction (involving both the card and the terminal/server) for EZ-Link [43], even though our protocol achieves additional security and privacy properties.

4.2 Back-end

We now consider the efficiency of our bill computation protocol described in Section 3.2.2. As explained in Section 3.1, the dominant computational cost of our protocol comes from the negation service, *i.e.,* the $NEG(\cdot)$ operator. In what follows, we give a theoretical efficiency analysis of this operation. Particularly, we consider the number of invocations of the $NEG(\cdot)$ operation with respect to the number of travel records, n, and the number of locations, l.

During each protocol run, the negation service is used for two tasks: (i) to compute the elements of matrix T (see equation (1)), and (ii) to perform the negation of the elements of matrix $T : t_{i,j}^{neg} = NEG(t_{i,j})$. Let N_u and N_t denote the numbers of calls to the negation service to perform tasks (i) and (ii), respectively, for each protocol run.

Let's first consider N_u. To compute the elements of matrix T, which is formed element-wise for each pair of travel records: an entry record, R_e, and an exit record, R_x, as shown by equation (1), the negation is invoked in order to check if the current user has already checked out in one of the previous travel records. (Note that here, we consider the respective check-in event only once.) Hence, for n travel records, we have

$$N_u = \sum_{i=1}^{n-2} \frac{i(i+1)}{2}.$$

On the other hand, to compute N_t, we consider the total number of pairs $\{R_e, R_x\}$, denoted by k. Let $\mathcal{N}$ be a set of travel records associated with a single protocol run. We then have $|\mathcal{N}| \leq n$, since $x > e$ for each pair $\{R_e, R_x\}$. Within $\mathcal{N}$, each location vector $\vec{l_i}$ forms $(n-i)$ pairs with the consecutive location vectors (up to $\vec{l_n}$). This implies that k is in fact the number of ordered pairs in $\mathcal{N}$, that is $k = n(n-1)/2$. Since the negation is carried out element-wise for each matrix T formed from each pair, we have

$$N_t = l^2 k = \frac{l^2 n(n-1)}{2}.$$

Finally, the total number, N, of calls to the negation service for each protocol run is

$$N = N_u + N_t = \sum_{i=1}^{n-2} \frac{i(i+1)}{2} + \frac{l^2 n(n-1)}{2} \approx n^3 \tag{2}$$

for sufficiently large n and where $l \ll n$.

We stress that bill computation can be performed *off-line* in the back-end and therefore does not have any timing impact on the performance of check-in/check-out events taking place in the front-end part of the system. Furthermore, the $NEG(\cdot)$ operation can be easily parallelized to further improve the overall performance of our billing computation protocol. To see this, we note that each element of matrix T within each protocol run can be computed independently, and hence rendering a highly parallelizable negation service. Let p denote the number of parallel negation requests that the negation service is able to support, and let τ be the average processing time of a single negation request. Then, by considering equation (2), the total time required to perform all requests to the negation service in parallel is expected to be

$$\frac{\tau N}{p} = \frac{\tau N_u}{p} + \frac{\tau N_t}{p} \tag{3}$$

in comparison with τN if the requests were to be performed sequentially. Hence, parallelization of the negation service can potentially speed-up the overall efficiency of our billing computation protocol by up to a factor of p.

To give a more accurate estimate, let us now consider some concrete numbers. Let $n = 10^4$. Let also $\tau = 0.15$ s, according to our implementation of the bill computation protocol with the Java Development Kit (JDK-7) [40] and Eclipse IDE for Java Developers [22] on an Intel Core 2 Quad 2.66 GHz machine running Windows 7. Plugging in these numbers to equations (2) and (3), we have $N \approx 10^{12}$ and the computational overhead for sequentially executing all the negation requests is estimated to be roughly 1.5×10^{11} s or 4.2×10^7 hours. However, by utilizing parallel processing with $p = 10^6$, which is well within our reach with current technology, such as MPI parallel computing [7] and Google's MapReduce [21], the computational overhead can be reduced to approximately 1.5×10^5 s or 42 hours. While this is still considerably expensive, more practical results seem achievable with further optimization of the protocol and advancement in high-performance parallel computing.

5 Related Work

Early work on privacy in public transport systems was given by Heydt-Benjamin *et al.* [30]. The proposed a privacy-preserving solution by combining techniques from e-cash and anonymous credential systems [13, 15] and recent advances in proxy re-encryption and re-signature systems [5, 6]. This way, the burden of key management can be shifted to more powerful mobile computing devices, such as mobile phones and PDAs, and thus requiring an e-ticket to store only the public portion of a secure key pair. Subsequently, Sadeghi *et al.* also investigated the problem of user privacy in public transport systems [45]. Particularly, they proposed an anonymous e-ticket system that relies on physically unclonable functions (PUFs) [54] and an anonymizer, which is either a dedicated hardware device or a software running on a mobile computing device. The anonymizer is assumed to be trusted to perform rerandomizable public key encryption to achieve unlinkability between transactions.

However, both approaches described in [30, 45] considered upfront e-payment, while we take a different, monthly billing approach. Further, no concrete performance analysis has been given, and thus it is not clear how costly is their public-key anonymous credential systems. Also, no travel records can be collected and hence the transportation company may not perform any data mining analyses on them.

There exist many other anonymous authentication protocols for RFIDs, see for example [9, 10, 31]. However, they either provide privacy against only unauthorized readers and eavesdroppers, or cannot be directly applied to our privacy-preserving e-ticketing system that supports monthly billing.

6 Conclusions and Open Problems

We highlighted some privacy issues in real world e-ticketing systems. We showed that it is possible to preserve the privacy of commuters against not only an eavesdropper, but also even the transportation company itself. We achieved this using partially homomorphic encryption, a more costly approach than existing solutions that rely on symmetric key techniques, however. Nevertheless, our solution supports monthly billing, implying that most of our computations with dominant overhead can be performed off-line. More importantly, we allow privacy-preserving mining of encrypted travel records. We performed a proof-of-concept implementation of our authentication and billing protocols and demonstrated that full privacy is achievable, although expensive. Hence, while this work presents an important first step, there remain at least two immediate open problems that need to be addressed.

First, naturally, we would like an even more efficient privacy-preserving billing protocol that allows data mining. A second important, but challenging, open problem is to alleviate or completely eliminate the need for trusted authorities in the billing architecture.

References

[1] Advanced Card Systems. ACR122U USB NFC reader. `http://nfc-reader.com/pages/contactless-readers/documents`.

[2] R. Agrawal, and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2000.

[3] C. Aggarwal, and P. Yu. Privacy-preserving data mining – models and algorithms. *Advances in Database Systems 34*. Springer, 2008.

[4] A. Alsaid and C.J. Mitchell. Installing fake root keys in a PC. In *Proceedings of the 2nd European Public Key Infrastructure Workshop (EuroPKI)*, pages 227–239. Springer, 2005.

[5] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *Proceedings of the 12th Network and Distributed System Security Symposium (NDSS)*, 2005.

[6] G. Ateniese, and S. Hohenberger. Proxy re-signatures: new definitions, algorithms, and applications. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS)*, 2005.

[7] P. Balaji, D. Buntinas, D. Goodell, W. Gropp, S. Kumar, E.L. Lusk, R. Thakur, and J.L. Träff. MPI on a million processors. In *Proceedings of the 16th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 20–30. Springer, 2009.

[8] J. Balasch, A. Rial, C. Troncoso, B. Preneel, I. Verbauwhede, and C. Geuens. PrETP: privacy-preserving electronic toll pricing. In *Proceedings of the 19th USENIX Security Symposium*, 2010.

[9] L. Batina, J. Hoepman, B. Jacobs, W. Mostowski, and P. Vullers. Developing efficient blinded attribute certificates on smart cards via pairings. In *Proceedings of the 9th IFIP International Conference (CARDIS)*, 2010.

[10] M. Burmester, B. de Medeiros, and R. Motta. Robust, anonymous RFID authentication with constant key-lookup. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, 2008.

[11] Calypso Networks Association. What's Calypso? `http://www.calypsonet-asso.org/index.php?rubrique=main_10`.

[12] J. Camenisch, M. Kohlweiss, and C. Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *12th International Workshop on Practice and Theory in Public Key Cryptography (PKC)*, 2009.

[13] J. Camenisch, and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology (EUROCRYPT)*, 2001.

[14] J. Camenisch, and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology (CRYPTO)*, 2002.

[15] J. Camenisch, and A. Lysyanskaya. Signature schemes and anonymous credentials from Bilinear maps. In *Advances in Cryptology (CRYPTO)*, 2004.

[16] R. Cramer, I. Damgård, and J. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology (EUROCRYPT)*, 2001.

[17] S. Clark. DBS, StarHub and EZ-Link to begin Singapore NFC pilot in December. `http://www.nfcworld.com/2010/11/29/35271/dbs-starhub-and-ez-link-to-begin-singapore-nfc-pilot-in-december/`

[18] A. Dabirsiaghi. JavaSnoop: How to hack anything written in JavaSnoop. Black Hat USA Technical Conference, 2010.

[19] C. Dwork. Differential privacy. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP)*, 2006.

[20] I. Damgård, and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography (PKC)*, 2001.

[21] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI)*, 2004.

[22] Eclipse IDE for Java EE Developers. `http://www.eclipse.org/`

[23] EZ-Link. EZ-Link Card System and Technology. `http://www.ezlink.com.sg/business/sys-tech.php`.

[24] EZ-Link. EZ-Link FAQs. `http://www.ezlink.com.sg/services/faqs/ez-link-faqs.php`.

[25] EZ-Link. EZ-Online and Top & Tap FAQs. `http://www.ezlink.com.sg/services/faqs/ez-online-faqs.php`.

[26] EZ-Link. Key Statistics. `http://www.ezlink.com.sg/corporate/keystats.php`.

[27] C. Gentry. Fully Homomorphic Encryption using Ideal Lattices. In *Proceedings of the 41st ACM Symposium on Theory of Computing (STOC)*, 2009.

[28] O. Goldreich. Foundations of Cryptography: Volume 2 – Basic Applications. *Cambridge University Press*, 2004.

[29] T.S. Heydt-Benjamin, D.V. Bailey, K. Fu, A. Juels, and T. O'Hare. Vulnerabilities in first-generation RFID-enabled credit cards. In *Proceedings of the 11th International Conference on Financial Cryptography and Data Security (FC)*, 2007.

[30] T.S. Heydt-Benjamin, H.-J. Chae, B. Defend, and K. Fu. Privacy for Public Transportation. In *Proceedings of the 6th Workshop on Privacy Enhancing Technologies (PET)*, 2006.

[31] J. Hoepman, and R. Joosten. Practical schemes for privacy and security enhanced RFID. In *Proceedings of the 4th IFIP International Workshop on Information Security Theory and Practices (WISTP)*, 2010.

[32] Land Transport Authority. Electronic Road Pricing. `http://www.lta.gov.sg/content/lta/en/motoring/erp_.html`.

[33] Y. Lindell, and B. Pinkas. Privacy-preserving data mining. *Journal of Cryptology 15(3)*, 2002.

[34] J. Liu, M. Au, W. Susilo, and J. Zhou. Enhancing location privacy for electric vehicles (at the right time). In *Proceedings of the 17th European Symposium on Research in Computer Security (ESORICS)*, 2012.

[35] K. Liu. Java Code for Paillier's Cryptosystem, 2006. `http://www.csee.umbc.edu/~kunliu1/research/Paillier.html`

[36] M. Marlinspike. New tricks for defeating SSL in practice. *Black Hat DC*, 2009.

[37] S. Meiklejohn, K. Mowery, S. Checkoway, and H. Shacham. The Phantom Tollbooth: Privacy-Preserving Electronic Toll Collection in the Presence of Driver Collusion. In *Proceedings of the 20th USENIX Security Symposium*, 2011.

[38] Octopus. Octopus Technology. http://www.octopus.com.hk/octopus-for-businesses/octopus-technology/en/index.html.

[39] Octopus. Top Up with AAVS. http://www.octopus.com.hk/easy-reloading/top-up-with-aavs/en/index.html.

[40] Oracle Corporation. The Java Development Kit 7. http://www.oracle.com/technetwork/java/javase/downloads/index.html.

[41] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology (EUROCRYPT)*, 1999.

[42] S. Prakasam. Status of CEPAS deployment. Land Transport Authority, http://www.itsc.org.sg/upload/download/aicp/CEPAS_Update.pdf, 2010.

[43] S. Prakasam. The evolution of e-payments in public transport: Singapore's experience. *Japan Railway & Transport Review*, 50:36–38, 2008.

[44] L. Qing. Singapore NFC services to finally go live. http://www.zdnet.com/sg/singapore-nfc-services-to-finally-go-live-7000002071/

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

[46] T. Sander, A. Young, and M. Yung. Non-Interactive CryptoComputing For NC1. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1999.

[47] Symantec Corporation. The application's digital signature cannot be verified. Do you want to run the application? http://www.symantec.com/docs/TECH143548, 2010.

[48] A. Shamir. How to share a secret. *Communications of the ACM 22(11)*, 1979.

[49] L. Sim, E. Seow, and S. Prakasam. Implementing an enhanced integrated fare system for Singapore. *Public Transport International*, 53:34–37, 2004.

[50] B. Song, and C.J. Mitchell. Scalable RFID security protocols supporting tag ownership transfer. *Computer Communications*, 34(4):556–566, Apr 2011.

[51] Singapore Standard. Specification for contactless e-purse application. SS 518:2006.

[52] T. Toft. Constant-rounds, almost-linear bit-decomposition of secret shared values. In *Proceedings of the Cryptographers' Track at the RSA Conference (CT-RSA)*, 2009.

[53] Transport for London. What is Oyster? http://www.tfl.gov.uk/tickets/14836.aspx.

[54] P. Tuyls, and L. Batina. RFID-Tags for anti-counterfeiting. In *Proceedings of the Cryptographers' Track at the RSA Conference (CT-RSA)*, 2006.

[55] USBlyzer. USB Protocol Analyzer and USB Traffic Sniffer for Windows. http://www.usblyzer.com/.

[56] J. Vaidya, C. Clifton, and M. Zhu. Privacy-preserving data mining. *Advances in Information Security 19*. Springer, 2006.

[57] Wireshark. Go Deep. http://www.wireshark.org/.

A Background on EZ-Link

Technology A newer generation of EZ-Link cards (issued since 2008) is compliant to the Singapore Standard—Specification for Contactless e-Purse Applications (CEPAS) [51]. Briefly, CEPAS provides a command set for performing e-purse operations on a stored-value smart card. It focuses on commands for debit, credit and transaction logging, with the syntax and format follow closely to that of the ISO/IEC 7816 standard series for electronic identification cards with contacts. CEPAS is designed to handle a large volume (approx. 20 millions [43]) of daily transactions, with each transaction speed expected to be less than 140 ms. From a security perspective, 3-DES in CBC mode is used as the main building block for mutual authentication, transaction logging, and both debit & credit commands. Nevertheless, CEPAS is designed with the flexibility to include modern cryptographic algorithms, such as AES and ECC, in a later stage [42]. Moreover, CEPAS is compatible with NFC (ISO/IEC 14443, ISO/IEC 18092), and hence supporting GSM SIM applications in NFC-enabled phones.

Protocols Each EZ-Link card is assigned a card serial number (CSN) and a card application number (CAN). The former is generated by the card manufacturer (to identify a specific card manufacturer), while the latter is generated by the e-ticket (or e-purse) issuer, where both are unique 8-byte values. Also, a CEPAS-compliant EZ-Link contactless smart card has at least one debit key and two credit keys used to perform the Debit (deduction) and Credit (increment) commands, respectively. These are 16-byte 3-DES keys generated using a key diversification technique, essentially a function over a master key, the CAN and (optionally) the CSN values. Further, the card has a 3-byte purse transaction counter (PTC), which is incremented by one each time the Debit or Credit commands are performed. When the PTC reaches its maximum value of $2^{24} - 1$, the card will not accept any further Debit and Credit commands [51]. This is an essential security feature designed particularly to prevent and detect fraudulent transactions.

Authentication A terminal (or reader) stores and protects one or more master keys in secure access modules (SAMs); while a card stores its debit/credit keys (derived from the master key) hidden in key files. With these keys, the terminal (T) and the card (C) mutually authenticate each other using a symmetric key approach. Prior to authentication, the terminal should have first obtained the CAN value of the communicating card as part of a card selection or anti-collision[7] process. Let `Card_Rand` and `Term_Rand` be random numbers generated by the card and the terminal, respectively. Let also $\mathsf{3DES}_K(\cdot)$ be the 3-DES encryption algorithm in CBC mode with key K. We give a simplified version of the authentication protocol as follows (see [51] for further details):

$$
\begin{aligned}
&(1)\ T \to C : \texttt{Get\_Challenge}\\
&(2)\ C \to T : \texttt{Card\_Rand}\\
&(3)\ T \to C : \texttt{Term\_Rand}, \mathsf{3DES}_{K_S^0}(\texttt{Trans\_Header}, \texttt{TRP}, \cdots)\\
&(4)\ C \to T : \mathsf{3DES}_{K_S^0}(\mathsf{3DES}_{K_S^1}(\texttt{Trans\_Header}), \cdots)
\end{aligned}
$$

Here, `Trans_Header` denotes a transaction header containing information about the transaction type, amount, date, and time; while `TRP` denotes terminal reference parameters.[8] We note that the authentication protocol above incorporates a Debit or Credit command (to be discussed in the following sections), in the sense that the card and the terminal make use of information from performing the Debit/Credit command to authenticate each other.

T and C share a session key K_S^0 computed from the concatenation of both the card's and the terminal's random numbers:

$$K_S^0 = \mathsf{3DES}_K(\texttt{Card\_Rand} \| \texttt{Term\_Rand})$$

[7] This refers to different ways to keep radio waves from one device from interfering with radio waves from another device.

[8] TRP is an arbitrary 4-byte value which allows the e-ticket issuer to incorporate dynamic transaction-related information into the cryptographic computation of the transaction. From a security viewpoint, this is to provide freshness to a protocol run.

where K is a debit or credit key that the terminal computes with its master key and the CAN value it received earlier. If the card is valid and has the debit/credit key corresponding to its CAN (that is submitted to the terminal), it should also derive the same K_S^0 as the terminal did, and thus, be able to recover the `Trans_Header` and `TRP` information in message (3). The protocol also makes use of a "signing key" of the form:

$$K_S^1 = \mathsf{3DES}_K(\texttt{PTC}, \texttt{TRP}, \texttt{Balance}, \cdots)$$

where `Balance` denotes the balance of the card. If the terminal is able to decrypt and recover the correct `Trans_Header`, the card is authenticated.

Debit The Debit command is designed to perform deduction, reading back of the remaining balance and updating of a transaction log file, all as a single atomic operation. The transaction amount is a signed integer, where a negative value indicates a debit and a positive[9] value indicates a credit [51].

As mentioned, the Debit command is executed as part of the authentication protocol shown before. The Debit process starts when the terminal creates and sends message (3) of the authentication protocol, where $\mathsf{3DES}_{K_S^0}(\texttt{Trans\_Header}, \texttt{TRP}, \cdots)$ is regarded as a "debit cryptogram". The card then checks the validity of the debit cryptogram and if it meets all the security conditions. If the check passes, the card prepares and returns a debit receipt record containing a "signed certificate" of the form $\mathsf{3DES}_{K_S^1}(\texttt{Trans\_Header})$ and encrypts it with its session key, as illustrated in message (4).

Credit The Credit command accepts two keys for security reasons. Apparently this is a security policy requirement for some issuers. Unlike the Debit command, the Credit amount must be positive and it is mandatory for the card operating system to check and enforce [51].

The Credit process is essentially similar to the Debit process. We now require that the terminal generates a credit cryptogram, and the card produces a credit receipt record.

[9] A positive value here allows refund or reimbursement, for example.