Efficient and secure stored-value cards with leakage resilience

Fuchun Guo
University of Wollongong, fuchun@uow.edu.au

Yi Mu
University of Wollongong, ymu@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Efficient and secure stored-value cards with leakage resilience

Abstract

Stored-value cards (SVCs) are a kind of smartcards that can be used to store monetary values. SVCs have demonstrated a wide range of applications in commerce. SVCs are in general tamper-resistant, but they are very computationally weak and their security mechanisms are also weak against side-channel attacks. In this paper, we propose an efficient and secure stored-value card system. With the help of pre-computations, storedvalue cards in our scheme are only required to perform simple arithmetic operations and bitwise comparisons. Our system especially for SVCs is secure with unbounded computational leakage resilience. Our novel scheme provides a practical solution to sidechannel attacks for light-weight devices.

Keywords

cards, efficient, secure, leakage, stored, resilience, value

Disciplines

Physical Sciences and Mathematics

Publication Details

Guo, F., Mu, Y. & Susilo, W. (2012). Efficient and secure stored-value cards with leakage resilience. Computers and Electrical Engineering, 38 (2), 370-380.

Efficient and Secure Stored-Value Cards with Leakage Resilience

Fuchun Guo[*], Yi Mu, Willy Susilo

Centre for Computer and Information Security Research, School of Computer Science and Software Engineering
University of Wollongong, Wollongong, NSW2522, Australia

Abstract

Stored-Value Cards (SVCs) are a kind of smartcards that can be used to store monetary values. SVCs have demonstrated a wide range of applications in commerce. SVCs are in general tamper-resistant, but they are very computationally weak and their security mechanisms are also weak against side-channel attacks. In this paper, we propose an efficient and secure stored-value card system. With the help of pre-computations, stored-value cards in our scheme are only required to perform simple arithmetic operations and bitwise comparisons. Our system especially for SVCs is secure with unbounded computational leakage resilience. Our novel scheme provides a practical solution to side-channel attacks for light-weight devices.

Keywords: Stored-Value Cards, System Security, Weak Computation Capability, Side-Channel Attacks

1. Introduction

With strong demands on electronic payment services in commerce, Stored-Value Cards (SVCs) [1] have greatly gained their popularity. SVCs are a kind of physical devices equipped with a computer chip. Differing from debit cards, SVCs directly store monetary values that have been purchased. The money in an SVC can be paid for a purchase without the need of any online third party.

SVCs cannot handle complex computations. In practice, SVCs are very simple devices with a weak computing processor. The nature of limited computation capability in SVCs rules out the feasibility of strong cryptography. It is usually believed that they can only conduct light-weight cryptography. As a consequence, it causes various security attacks. The recent attack [2] on Oyster cards is an example. Oyster cards are SVCs used in public transport services in the United Kingdom. Recently, it has been shown that its cryptographic key can be obtained easily so that it can be used to clone a card.

Side-channel attacks on secure hardware implementation have been becoming a major concern. Side-channel attacks do not attack the computational properties on cryptographic algorithms, while they attack system implementation through such as electromagnetic analysis [3], power consumption analysis [4], timing analysis[5], and memory analysis [6]. The most effective side-channel attacks can extract cryptographic keys. Unfortunately, most cryptographic schemes are not leakage-resilient, and cannot be proven secure under side-channel attacks. Furthermore, because SVCs are cheap devices, they cannot shield leakage [7, 8, 9] or adopt strong elliptic curve cryptography [10] for leakage resilience.

Providing practical and effective side-channel resistant security schemes for SVCs is a challenging task. In this paper, we propose an efficient security system for SVCs, which is secure against leakage from side-channel attacks. The core of our scheme is a novel pre-computation scheme with the aid of message authentication codes [11] and digital signatures [12, 13, 14]. Our scheme is very efficient and provably secure with unbounded computational leakage resilience.

[*]*Corresponding author*
Email addresses: fg278@uow.edu.au (Fuchun Guo), ymu@uow.edu.au (Yi Mu), wsusilo@uow.edu.au (Willy Susilo)

1.1. *Related Work*

Many secure electronic payments are invented over the past decades. There are *online* scenario (e.g.[15]) and *offline* scenario (e.g.[16, 17, 18]). In the online scenario, when a client pays to a merchant, it requires the third party (e.g. bank) to complete money transaction. The debit card system falls into this scenario. In the offline scenario, money transactions from clients to merchants do not require any third party, and the stored-value card system falls into this scenario.

Definitely, the offline scenario provides a more flexible money transaction compared to the online scenario. This convenience, however, requires a more secure system since a dishonest client could double spend the same money token. E-cash and e-wallet are two distinct approaches and secure payments for the offline scenario.

E-cash was invented by Chaum [16] and has received lots of extensive studies (e.g. [17, 18]). In this system, clients can withdraw money tokens and spend them in an anonymous way. If a client double spends a token, his identity will be revealed. To be able to trace dishonest clients, it requires all clients to register their identities before joining the e-cash system.

E-wallet was first invented by Even and Goldreich [19], and it also receives many studies (e.g. [20, 21]). In comparison with e-cash system, the e-wallet system utilizes tamper-resistant devices to prevent clients from double spending. Clients do not need to register their identities before joining the system. In comparison with the e-cash system, the e-wallet system receives a wide range of clients without the guarantee of credits.

The stored-value card is one of e-wallets. However, these e-wallet systems (i.e. [19, 20, 21]) require the wallet to perform complex computations like exponentiations. The wallet therefore must be powerful enough to conduct heavy computations. Another security issue is that most of previous e-wallet systems do not take side-channel attacks into account. They could be insecure in the presence of side-channel attacks.

1.2. *Our Contribution*

In this paper, we propose an efficient and secure stored-value card system. The computations in stored-value cards are very small, and our system for stored-value cards is secure with unbounded computational leakage resilience. In comparison with existing systems for SVCs, our system significantly improves both efficiency and security. They are described in detail as follows.

- The stored-value cards in our system only conduct simple arithmetic operations and bitwise comparisons. These computations are negligible compared to exponentiations/pairings in existing e-wallet systems. Therefore, stored-value cards in our system can be light-weight equipped with cheap computer chips.
- The stored-value cards in our system do not need to shield any computation leakage in the presence of side-channel attacks. Our system are secure with unbounded computational leakage resilience. Even all accessed data in computation in stored-value cards are leaked to the adversary, the adversary still cannot break our system.

The rest of this paper is organized as follows. In Section 2, we introduce and define stored-value card system. In Section 3, we provide some preliminaries that are related to our scheme. In Section 4, we present our scheme in detail. In Section 5, we analyze the security of our scheme. The final section is our conclusion.

2. Stored-Value Card System

2.1. *System Description*

As shown in Figure 1, a stored-value card system consists of the entities of Issuer, Client, Server and Merchant, and the devices of stored-value card C, top-up machine $\mathcal{S}$ and card reader $\mathcal{R}$. They are described as follows:

- *Issuer.* This is the entity who generates parameters for stored-value cards, top-up machines, and card readers. The issuer only involves in the setup of parameters, and is assumed to be a trusted party.
- *Client.* This is the entity who holds a stored-value card C to store monetary value. The stored-value card is assumed to be tamper-resistant, such that dishonest clients cannot break the device to extract sensitive information stored in cards.

- *Server.* This is the entity who provides a top-up machine $\mathcal{S}$ for recharging stored-value cards. When $\mathcal{S}$ recharges C, it generates an unforgeable recharge token and this token will recharge C successfully.
- *Merchant.* This is the entity who holds a card reader $\mathcal{R}$. With this card reader, the merchant receives money tokens, and collects money back from servers.

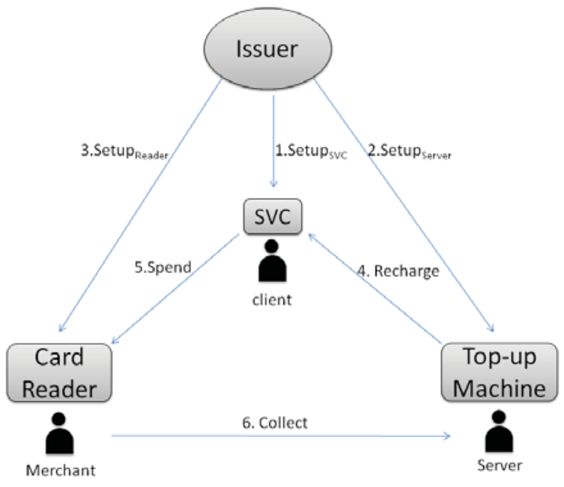

Figure 1: Stored-value card system.

2.2. *Features of SVC System*

The SVC system offers the following two features:

The first feature is no identity registration for clients. In the SVC system, different from E-cash, clients do not involve any cryptographic computation. All computations are secretly completed in stored-value cards. With stored-value cards, clients interact with servers for recharging and with merchants for paying. They are not required to respond any security issue; therefore, identity registration is not necessary.

The second feature is no third party for money transactions. When clients pay to merchants, the money transaction is not completed by the third party in a centralized database but in stored-value cards and card readers. Precisely, a stored-value card decreases its current balance and generates a money token; while a card reader verifies the money token and increases its current balance. All these computations do not involve any third party.

The above two features indicate that each stored-value card is uploaded with a secret key and unknown to clients. As a consequence, stored-value cards must be tamper-resistant such that no adversary can tamper the device to get the secret key; otherwise, it is impossible to achieve secure SVC system. For example, the adversary can forge money token or change the current balance in SVCs. We also note that there have been some attacks [22] against tamper resistance. The tamper resistance property however is essential for devices to securely store secret information. To achieve a secure SVC system capturing the above two features, we believe or assume that tamper resistance is available in future.

The three devices $C, \mathcal{R}$ and $\mathcal{S}$ are three kinds of devices in SVC system. Stored-value cards C are sold to a large group client. They should be of light-weight devices, and therefore they provide very limited computation capability. We assume that this light-weight device cannot shield the computation leakage in the presence of side-channel attacks. In other words, stored-value cards are tamper-resistant but with weak computation ability and computation leakage. On the other hand, card readers $\mathcal{R}$ are sold to merchants and they can be relatively expensive. We assume card readers are tamper-resistant with powerful computation capability and leakage resistance. In this work, the top-up machines $\mathcal{S}$ are assumed to be powerful and managed by a trusted server.

2.3. Security Requirements

A secure stored-value card system must fulfil the following security requirements.

Recharge Security. Unauthorized Stored-value cards should be prevented from successfully recharging. Only the top-up machine $\mathcal{S}$ is able to generate valid recharge tokens to recharge SVCs. Meanwhile, a recharge token should be prevented from double recharging.

Spend Security. Money tokens generated by SVCs should be prevented from illegally forging. A stored-value card can output a money token if and only if contains a sufficient monetary value. Meanwhile, a money token should be prevented from double spending.

Collect Security. Merchants should be prevented from collecting more monetary amount than they have received from clients. Let V_R be the amount of money received from SVCs , and let V_C be the amount of money to be collected. The security requires $V_R = V_C$.

Leakage-Resilient Security. SVCs are light-weight devices and we assume they cannot shield any computation leakage. Therefore, all accessed data in computation in stored-value cards must be leakage-resilient.

We say that a stored-value card system is secure if the amount of monetary value recharged into SVCs is equivalent to the amount of money collected from servers. That is, the sum of balance before and after money transactions is the same. When the recharge security, spend security and collect security all hold, we immediately get that the SVC system is secure.

2.4. Our Definition

In our definition, we assume both card readers and SVCs store a current balance, which cannot be tampered. When an SVC is recharged, its current balance increases. When a client pays to a merchant, the current balance in the SVC decreases and the current balance in the card reader increases. According to the figure 1, a stored-value card system can be defined as follows.

Setup: In this phase, an algorithm is run by the Issuer. The algorithm setups system parameters for $C, \mathcal{S}, \mathcal{R}$. The current balance in both C and $\mathcal{R}$ are initialized with zero.

Recharge: In this phase, an interactive protocol is run by the Client and the Server. Taking as input C, $\mathcal{S}$ and a monetary value V_c to recharge, the protocol increases current balance B_C in C as $B_C := B_C + V_c$, or outputs *fail*.

Spend: In this phase, an interactive protocol is run by the Client and the Merchant. Taking as input C, $\mathcal{R}$ and a monetary value V_s to pay, the protocol increases current balance $B_{\mathcal{R}}$ in $\mathcal{R}$ as $B_{\mathcal{R}} := B_{\mathcal{R}} + V_s$ and decreases B_C in C as $B_C := B_C - V_s$, or outputs *fail*.

Collect: In this phase, the Merchant collects money from the Server and the monetary amount is $B_{\mathcal{R}}$. Then, Server resets $B_{\mathcal{R}}$ to 0.

3. Preliminaries and Definition

3.1. Message Authentication Code

A message authentication code (MAC) scheme is composed of the following three algorithms.

KeyGen: On input a security parameter 1^λ, the algorithm returns a private key K.

TagCom: On input a message M from the message space and the secret key K, the algorithm returns an authentication tag $\tau_M = \mathsf{MAC}_K[M]$.

Verify: On input an authenticated message (M, τ_M) and the private key K, the algorithm returns *accept* or *reject*.

A secure MAC scheme should be existentially unforgeable against chosen-message attacks [11]. The security notion can be defined using a game model between a challenger and an adversary as follows.

Setup: The challenger runs the KeyGen algorithm to generate a private key K.

Query: The adversary queries a tag on message M_i, which is adaptively chosen. The challenger runs the TagCom algorithm to generate the tag τ_{M_i}, which is sent to the adversary. Let q be the number of tag queries.

Forgery: The adversary outputs a forged authenticated message (M^*, τ_{M^*}) and wins the game if the forged tag is correct on the new message M^* that was not queried in the query phase.

Definition 1. *A message authentication code scheme is (t, q, ϵ) existentially unforgeable against chosen-message attacks if there exists no adversary $\mathcal{A}$ who makes q queries at most can forge a valid tag with probability ϵ at least in t time.*

3.2. Digital Signatures

A digital signature scheme consists of the following three algorithms.

KeyGen: On input a security parameter 1^λ, the algorithm returns a key pair (pk, sk), where pk denotes the public key and sk denotes the signing key.

Sign: On input a message M from the message space and the signing key sk, the algorithm returns a signature $\sigma_M = \mathsf{Sign}_{sk}[M]$.

Verify: On input a signed message (M, σ_M) and the public key pk, the algorithm returns *accept* or *reject*.

The standard security notion [23] for digital signatures is existentially unforgeable against chosen-message attacks. A game can be simulated by considering a challenger and an adversary as follows.

Setup: The challenger uses the KeyGen algorithm to generate a key pair (pk, sk), and sends the public key pk to the adversary.

Query: The adversary queries a signature on message M_i, which is adaptively chosen. The challenger uses the Sign algorithm to generate the signature σ_{M_i}, which is sent to the adversary. Let q be the number of signature queries.

Forgery: The adversary outputs a forged signed message (M^*, σ_{M^*}) and wins the game if the forged signature is correct on the new message M^* that was not queried in the query phase.

Definition 2. *A digital signature scheme is (t, q, ϵ) existentially unforgeable against chosen-message attacks if there exists no adversary $\mathcal{A}$ who makes q queries can forge a valid signature with probability ϵ at least in t time.*

3.3. Bilinear Pairing and Complexity Assumption

Let $\mathbb{PG} = (\mathbb{G}, \mathbb{G}_T, p, g, e)$ be a bilinear pairing group, where $\mathbb{G}, \mathbb{G}_T$ are multiplicative groups of prime order p, g is a generator of $\mathbb{G}$, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is the bilinear map. The bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ should satisfy that

- For all $g \in \mathbb{G}, a, b \in \mathbb{Z}_p$, $e(g^a, g^b) = e(g, g)^{ab}$.
- $e(g, g) \neq 1$ if g generates the group $\mathbb{G}$.

The bilinear pairing $\mathbb{PG}$ defined above is a symmetric pairing. There exists another pairing definition, named as asymmetric pairing $\mathbb{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$. Here, the bilinear map e is defined as $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. This asymmetric pairing is utilized to construct shortest possible signatures. More detailed discussions can be found in [12, 24].

Our scheme uses the technique of digital signatures [12, 13] which are based on the hardness of Computational Diffie-Hellman problem (CDH). We simply revisit this assumption as follows.

Definition 3. *The CDH problem is (t, ϵ)-hard if given $g, g^a, g^b \in \mathbb{G}$ for over random choice of $g \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, there exists no adversary who can output g^{ab} in t-time with probability ϵ at least.*

4. Our Scheme

4.1. Description

The core of a secure SVC system is the recharge phase and the spend phase. Before presenting our scheme, we first give a high-level description.

We utilize a message authentication code (MAC) to secure the recharge phase. We assume a success recharge will let the current balance in SVCs increase a fixed monetary value denoted by V_c. To recharge an SVC, the top-up machine generates a recharge token (i.e., authentication tag), and sends it to the SVC. The SVC verifies the recharge token and increases its current balance when valid. Since the authentication tag of a secure MAC scheme is unforgeable, no adversary can forge recharge tokens.

We utilize a digital signature to secure the spend phase. We set the money token to be an unforgeable digital signature. To pay monetary value amount of V_s to a merchant, the client uses his SVC to generate the money token (i.e., signature on V_s). The SVC firstly checks the current balance $B_C \geq V_s$, then decreases its balance as $B_C := B_C - V_s$, and finally returns a signature on V_s. Upon receiving this money token, the current balance $B_{\mathcal{R}}$ in the reader increases its current balance as $B_{\mathcal{R}} := B_{\mathcal{R}} + V_s$. Since digital signatures of a secure signature scheme is unforgeable, no adversary can forge money tokens.

The above descriptions show the basic sketch of applying message authentication codes and digital signatures in our system. However, they are not secure in this way and the system must be able to withstand double-recharging attacks, double-spending attacks and side-channel attacks.

Double-recharging attacks. We withstand the double-recharging attacks by utilizing an unique string and an index for SVCs. Precisely, each SVC is assigned with an unique string ID_C and a counter C_1 initialized with $C_1 := 1$. To recharge an SVC, the top-up machine reads its (ID_C, C_1) and generates an authentication tag on (ID_C, C_1). Upon receiving a recharge token, the SVC reads (ID_C, C_1) and checks the validness of this token on (ID_C, C_1). If valid, increases its current balance as $B_C := B_C + V_c$ and increases its counter as $C_1 := C_1 + 1$. We have that the unique string will prevent different SVCs from being recharged with the same token, and the update of counter will prevent the same SVC from being recharged with the same token.

Double-spending attacks. We withstand the double-spending attacks by utilizing an unique string and an index for readers. The approach is similar to that against double-recharging attacks. Similarly, each reader is assigned with an unique string $ID_{\mathcal{R}}$ and a counter C_3 initialized with $C_3 := 1$. To receive a money token amount of V_s monetary value from a client, the reader reads $(ID_{\mathcal{R}}, C_3)$ and sets the message as $(ID_{\mathcal{R}}, C_3, V_s)$. Upon receiving a money token (i.e., signature) on this message, the reader reads $(ID_{\mathcal{R}}, C_3)$ and checks the validness of this token on $(ID_{\mathcal{R}}, C_3, V_s)$. If valid, increases its current balance as $B_{\mathcal{R}} := B_{\mathcal{R}} + V_s$ and increases its counter as $C_3 := C_3 + 1$. We have that the unique string will prevent different readers from accepting the same money token, and the update of counter will prevent the same reader from double accepting the same money token.

Side-channel attacks. We withstand the side-channel attacks by utilizing pre-computations and an index for SVCs. The computation in SVCs could leak sensitive information in the recharge phase and the spend phase. In the recharge phase, all valid recharge tokens have been pre-computed and stored in SVCs, such that there is no computation requiring the private key and signing key. The SVC only compares bit strings when receiving a recharge token from outside. A recharge token is valid if and only if it is associated with the current state of (ID_C, C_1), which cannot be modified by an adversary. We require the counter C_1 to increase as $C_1 := C_1 + 1$ when the verification fails. Even all valid tokens are leaked though the bitwise comparison, these tokens are no longer valid because they do not fulfil the current state of (ID_C, C_1). In the spend phase, we construct a new signature scheme modified from [14] to pre-compute all money tokens such that the generation of money tokens in SVC will not leak the signing key.

4.2. Construction

We utilize the Boneh-Lynn-Shacham signature scheme [12] as the MAC scheme, which is provably secure under the CDH assumption and provides the shortest signature length. We construct a new signature scheme variant from [13, 14], which is provably secure and computational-leakage resilient under the CDH assumption. Our scheme is described as follows:

Setup: The Issuer generates system parameters in Steps 1-3.

Step 1. The Issuer generates the key K for message authentication and generates the public/signing key pair (pk, sk) for signature operations.

$$K = \alpha, \quad pk = (\mathbb{PG}, g_1, g_2, u_0, u_1, \cdots, u_n), \quad sk = g_2^{\beta},$$

The MAC scheme is defined as follows. α is randomly chosen from $\mathbb{Z}_p$. Let $H : \{0,1\}^* \rightarrow \mathbb{G}$ be a collision-resistant hash function, where the input is an arbitrary string and the output is an element of the pairing group $\mathbb{G}$ [12]. The MAC function is defined as $\mathsf{MAC}_K(M) = H(M)^{\alpha}$.

The key pair (pk, sk) is defined as follows. $\mathbb{PG}$ is the pairing group. β is randomly chosen from $\mathbb{Z}_p$ and g_1 is set as g^{β}. The elements $g_2, u_0, u_1, \cdots, u_n$ are chosen at random from $\mathbb{G}$. The length of messages to be signed is an n-bit string. We discuss the size n in later sections.

Step 2. The Issuer uses K to compute recharge parameters (RP) and uses sk to compute spend parameters (SP).

$$RP = (ID_C, N_1, C_1, Q_1, Q_2, \cdots, Q_{N_1}), \ SP = (N_2, C_2, P_1, P_2, \cdots, P_{N_2}).$$

The parameter RP is defined as follows. ID_C is an unique string for each C. N_1 denotes the maximal number of times for recharge. C_1 is a counter initialized with $C_1 := 1$. Each Q_i is computed as

$$Q_i = H(ID_C \parallel i)^{\alpha} \in \mathbb{G} \text{ , for all } i = 1, 2, \cdots, N_1.$$

The parameter SP is defined as follows. N_2 denotes the maximal number of times for spending. C_2 is a counter initialized with $C_2 := 0$. Each P_i for all $i = 1, 2, \cdots, N_2$ is independently computed by the Issuer below.

- Choose at random $r \in \mathbb{Z}_p$ and $t_1, t_2, \cdots, t_n \in \mathbb{Z}_p$ such that

$$t_1 + t_2 + \cdots + t_n = 0 \mod p.$$

- Compute $P_i = \Big((s_{0,1}, s_{1,1}), (s_{0,2}, s_{1,2}), \cdots, (s_{0,n}, s_{1,n}), s_r\Big)$, which is defined as

$$\left(\begin{array}{c|c|c|c|c} s_{0,1} & s_{0,2} & s_{0,3} & \cdots & s_{0,n} \\ \hline s_{1,1} & s_{1,2} & s_{1,3} & \cdots & s_{1,n} \end{array}, \ s_r \right) = \left(\begin{array}{c|c|c|c} g_2^{\beta}(u_0 u_1^0)^r \cdot g^{t_1} & (u_2^0)^r \cdot g^{t_2} & \cdots & (u_n^0)^r \cdot g^{t_n} \\ \hline g_2^{\beta}(u_0 u_1)^r \cdot g^{t_1} & (u_2)^r \cdot g^{t_2} & \cdots & (u_n)^r \cdot g^{t_n} \end{array}, \ g^r \right).$$

Here, we define $u_i^0 = 1_{\mathbb{G}}$ for all $i = 1, 2, \cdots, n$.

Step 3. The Issuer pre-loads C with (B_C, RP, SP), $\mathcal{S}$ with K, and $\mathcal{R}$ with $(B_{\mathcal{R}}, ID_{\mathcal{R}}, C_3, pk)$ by a secure channel. Here, the current balance $B_C, B_{\mathcal{R}}$ are initialized with $B_C := 0$ and $B_{\mathcal{R}} := 0$. $ID_{\mathcal{R}}$ is an unique for each reader $\mathcal{R}$. C_3 is another counter initialized with $C_3 := 1$.

All parameters for the stored-value card C, top-up machine $\mathcal{S}$ and card reader $\mathcal{R}$ are listed in Table 1.

	C	$\mathcal{S}$	$\mathcal{R}$
Current Balance	B_C		$B_{\mathcal{R}}$
Recharge Parameters	$(ID_C, N_1, C_1, Q_1, Q_2, \cdots, Q_{N_1})$	K	
Spend Parameters	$(N_2, C_2, P_1, P_2, \cdots, P_{N_2})$		$(ID_{\mathcal{R}}, C_3, pk)$

Table 1: The uploaded parameters.

Recharge: Let V_c be the amount of monetary value for recharging and fixed for each time. The interaction between C and $\mathcal{S}$ is described in Steps 4-6.

Step 4. C reads and sends $M_c = ID_C \parallel C_1$ to $\mathcal{S}$ if $C_1 \leq N_1$. Otherwise, outputs *fail*.

Step 5. $\mathcal{S}$ generates the recharge token $\tau_{M_c} = H(M_c)^{\alpha}$ using K and sends it to C. Meanwhile, the Client is charged with the equivalent monetary value V_c.

Step 6. C accesses Q_{C_1} to verify the correctness of $\tau_{M_c} = Q_{C_1}$. If it is equal, increases its balance $B_C := B_C + V_c$ and increases its counter by one as $C_1 := C_1 + 1$; otherwise, outputs *fail* and increases its counter as $C_1 := C_1 + 1$ only.

The interaction of this phase is given in Figure 2.

$$
\begin{array}{lcl}
\qquad C & & \qquad \mathcal{S} \\
\textbf{4. } C_1 \overset{?}{\leq} N_1 & & \\
\quad M_c = ID_C \parallel C_1 & & \\
 & \xrightarrow{\;\;M_c\;\;} & \\
 & & \textbf{5. } \tau_{M_c} = H(M_c)^{\alpha} \\
 & \xleftarrow{\;\;\tau_{M_c}\;\;} & \\
\textbf{6. } \tau_{M_c} \overset{?}{=} Q_{C_1} & & \\
\quad B_C := B_C + V_c & & \\
\quad C_1 := C_1 + 1 & &
\end{array}
$$

Figure 2: The description of recharge phase.

Spend: Let V_s be the amount of monetary value to pay. The interaction between C and $\mathcal{R}$ is described in Steps 7-9.

Step 7. $\mathcal{R}$ reads $(ID_{\mathcal{R}}, C_3)$ and sends $M_s = ID_{\mathcal{R}} \parallel C_3 \parallel V_s$ to C.

Step 8. C outputs *fail* if $C_2 + 1 = N_2$. Otherwise, C checks its current balance B_C. If $B_C \geq V_s$, C decreases its balance $B_C := B_C - V_s$, increases its counter C_2 by one as $C_2 := C_2 + 1$ and generates the money token σ_{M_s} for $\mathcal{R}$ using P_{C_2}; otherwise, outputs *reject*.

The signature σ_{M_s} is generated as follows. Let the message $M_s = m_1 m_2 \cdots m_n$ be an n-bit string and P_{C_2} be

$$P_{C_2} = \left(\begin{array}{ccccc} s_{0,1} & s_{0,2} & s_{0,3} & \cdots & s_{0,n} \\ s_{1,1} & s_{1,2} & s_{1,3} & \cdots & s_{1,n} \end{array} , \; s_r \right).$$

C accesses $s_{m_i,i}$ and s_r for all i, and outputs the signature σ_{M_s}.

$$\sigma_{M_s} = \left(s_{m_1,1}, s_{m_2,2}, \cdots, s_{m_n,n}, s_r \right).$$

Step 9. $\mathcal{R}$ uses pk to verify the money token σ_{M_s} of M_s. If the signature is correct, increases the balance $B_{\mathcal{R}} := B_{\mathcal{R}} + V_s$ and increases its counter by one as $C_3 := C_3 + 1$; otherwise, outputs *reject* and increases its counter as $C_3 := C_3 + 1$ only. The verification is computed as follows. Given (M_s, σ_{M_s}), $\mathcal{R}$ computes $\sigma_1 = s_{m_1,1} \cdot s_{m_2,2} \cdots s_{m_n,n}$, sets $\sigma_2 = s_r$ and checks that

$$e\big(\sigma_1, g\big) = e(g_1, g_2) e\Big(\sigma_2, u_0 \prod_{i=1}^{n} u_i^{m_i}\Big).$$

The interaction of this phase is given in Figure 3.

$$\begin{array}{lcl}
\qquad\mathcal{R} & & \qquad\mathcal{C} \\
\textbf{7. } M_s = ID_{\mathcal{R}} \parallel C_3 \parallel V_s & & \\
& \xrightarrow{M_s} & \\
& & \textbf{8. } C_2 \overset{?}{=} N_2 \\
& & \quad B_C := B_C - V_s \\
& & \quad C_2 := C_2 + 1 \\
& & \quad \sigma_{M_s} \leftarrow P_{C_2} \\
& \xleftarrow{\sigma_{M_s}} & \\
\textbf{9. } \sigma_{M_s} \overset{?}{=} \mathsf{Sign}_{sk}[M_s] & & \\
\quad B_{\mathcal{R}} := B_{\mathcal{R}} + V_s & & \\
\quad C_3 \ := C_3 + 1 & &
\end{array}$$

Figure 3: The description of spend phase.

Collect: The Merchant collects $B_{\mathcal{R}}$ monetary amount from the Server, and the Server initializes $B_{\mathcal{R}}$ with $B_{\mathcal{R}} = 0$.

4.3. Main Features

Efficiency. Stored-value cards in our scheme require very small computations. In the recharge phase and spend phase, they only conduct bitwise comparisons and simple arithmetic operations. Especially, no complex computation such as exponentiation is required for signature generation using P_i. These simple computations can be definitely performed in light-weight cards.

Security. Our scheme is secure and captures all security requirements defined in Section 2. We will prove that there exists no adversary who can break the sum of balance before and after money transactions in polynomial time with a non-negligible probability.

4.4. Memory Requirement

The memory requirement of SVCs in our scheme is mainly dominated by Q_i and P_i. Let $|X|$ be the length of X. We estimate that each SVC requires a storage capacity about

$$N_1|Q_i| + N_2|P_i| = N_1|\mathbb{G}| + N_2(2n+1)|\mathbb{G}|. \tag{1}$$

For a message $M_s = ID_{\mathcal{R}} \parallel C_3 \parallel V_s$ with an n-bit length to be signed, we can set (say) $n = 75$. $ID_{\mathcal{R}}$, C_3 and V_s can be defined as follows.

- $|ID_{\mathcal{R}}| = 25$. According to our definition, each $\mathcal{R}$ should be assigned with a different $ID_{\mathcal{R}}$ number. When $|ID_{\mathcal{R}}| = 25$, it allows 2^{25} different readers as the upper bound.
- $|C_3| = 35$. Each payment request from SVCs will increase the counter C_3 by one. We set $|C_3| = 35$ so that each $\mathcal{R}$ can operate 2^{35} times of money transactions in total.
- $|V_s| = 15$. V_s is the monetary value. When it is a 15-bit string, it can represent 2^{15} different monetary coins in payment. For example, let 10 cents be the minimal unit in spending, we have $V_s \approx 3,000$ dollars as the maximal value.

For the pairing group $\mathbb{G}$, we can utilize the asymmetric pairings [24] to reduce the storage. When 80-bit security is set, it means that the representation of group elements from $\mathbb{G}$ is as short as 160 bits in length, i.e., $|\mathbb{G}| = 160$. Putting $n = 75$ and $|\mathbb{G}| = 160$ into equation (1), we yield the following result

$$N_1|Q_i| + N_2|P_i| = N_1 \cdot 20 + N_2 \cdot 3020 \text{ (Bytes)}. \tag{2}$$

Now, we describe the choice of N_1, N_2 for our scheme. In practice, suppose a client recharges his SVC once for every week on average and uses four times every day on average. We give the memory requirement in Table 2 for SVCs depending on the term of validity.

	One Weak	One Month	One Year	Five Years	Ten Years
$N_1\|Q_i\| + N_2\|P_i\|$	85 KBytes	363 KBytes	4.5MBytes	22.5MBytes	45 MBytes

Table 2: Memory requirement depending on the term of validity for SVCs.

5. Security Analysis

5.1. Security Model

Let V_R be the amount of monetary value recharged into SVCs and let V_C be the amount of money to be collected. We say that a stored-value card system is secure if and only if $V_R = V_C$. We use the following model to define the security in terms of the sum of balance. The model is interacted between a challenger and an adversary. The challenger generates $\mathcal{C}, \mathcal{S}, \mathcal{R}$ and the adversary can make queries to them. Our security model allows the adversary to gain unbounded leakage from $\mathcal{C}$, assuming that only accessed data in computation leaks information. The security model is defined as follows.

Setup: The challenger generates the keys K and (pk, sk), and pre-loads parameters into $\mathcal{C}, \mathcal{S}, \mathcal{R}$ defined the same as the setup phase. Then, $\mathcal{C}, \mathcal{S}, \mathcal{R}$ are given to the adversary.

Query. In this phase, the adversary can make queries on the following list.

- Recharge Token. The adversary acts as $\mathcal{C}$ and makes a recharge token query to $\mathcal{S}$ on the adaptively chosen M_{c_i}. $\mathcal{S}$ responds to $\mathcal{C}$ by following Step 4. Let the adversary repeat q_1 times at most in this phase.
- Money Token. The adversary acts as $\mathcal{R}$ and makes a money token query to $\mathcal{C}$ on the adaptively chosen M_{s_i}. $\mathcal{C}$ responds to $\mathcal{R}$ by following Step 8. The adversary repeats it for q_2 times at most in this phase.
- Recharge On $\mathcal{C}$. The adversary acts as $\mathcal{S}$ and makes a recharge request to $\mathcal{C}$. $\mathcal{C}$ responds to $\mathcal{S}$ by following Step 4 and Step 6. The adversary repeats it for q_3 times at most in this phase.
- Increase On B_R. The adversary acts as $\mathcal{C}$ and makes a spend request to $\mathcal{R}$. $\mathcal{R}$ responds to $\mathcal{C}$ by following Step 7 and Step 9. The adversary repeats it for q_4 times at most in this phase.
- Leakage Query. The adversary makes leakage queries from $\mathcal{C}$. Observe that the leakage happens only in Step 4, Step 6, and Step 8. Let L be all data used in computation during these steps. L is sent to the adversary.
- Collect Query. The adversary acts as $\mathcal{R}$ and makes collect request to the challenger. The challenger responds to the adversary by following the collect phase.

Win. Let V_R be the monetary amount of recharge tokens queried by the adversary and V_C be the amount of money to be collected from the adversary. The adversary wins the game and breaks the security if $V_C > V_R$.

Definition 4. *A stored-value card system is $(t, q_1, q_2, q_3, q_4, \epsilon)$-secure against the attacks defined in the game, if there exists no adversary $\mathcal{A}$, who can win the game in running time t with probability ϵ at least after making the queries defined as above.*

5.2. Security Proof

We prove our scheme is secure against attacks defined in the above model. We show that if there exist attacks on our scheme, we can utilize the attacks to forge valid signatures of the signature schemes [12, 13]. With the security proofs in [12, 13], we can further program the reduction proofs for reducing the forged signatures to solving the computational difficulty of CDH problem. Since these two schemes are provably secure under the CDH assumption. Without redundancy, we present the reduction to forging valid signatures only, which are equivalent to solving the CDH problem.

Theorem 1. *Our scheme is $(t_1 + t_2, q_1, q_2, q_3, q_4, \epsilon)$-secure assuming that*

- *The Boneh-Lynn-Shacham signature scheme is (t_1, q, ϵ) secure, where $q = \{q_1, q_3\}_{max}$.*
- *The Waters signature scheme is (t_2, q_2, ϵ) secure.*

Proof 1. *Suppose there exists an adversary $\mathcal{A}$ who can $(t_1 + t_2, q_1, q_2, q_3, q_4, \epsilon)$-break our scheme. We construct an algorithm $\mathcal{B}$ that breaks the security of the Boneh-Lynn-Shacham signature scheme (BLS) or the Waters signature scheme (Wat). The algorithm $\mathcal{B}$ receives the challenge public keys from BLS and Wat, its challenge output is a forged signature of BLS or Wat using the adversary's ability. The interaction between the algorithm $\mathcal{B}$ and the adversary $\mathcal{A}$ is as follows.*

Setup. *Let (pk_B, sk_B) be the key pair of the BLS signature scheme and (pk_W, sk_W) be the key pair of the Wat signature scheme. $\mathcal{B}$ receives pk_B, pk_W from BLS and Wat, and sets $K = sk_B, pk = pk_W$. Then, $\mathcal{B}$ simulates as $C, \mathcal{R}, \mathcal{S}$ for the adversary.*

Query. *In this phase, the adversary can make queries on the following list.*

- Recharge Token. *The adversary acts as C and makes a recharge token query to $\mathcal{S}$ on the adaptively chosen M_{c_i}. Upon receiving this query, $\mathcal{B}$ responds by querying the signature on M_{c_i} to the BLS scheme. Let the signature be $\mathsf{Sign}^B_{sk_B}[M_{c_i}]$, $\mathcal{B}$ sets $\tau_{M_{c_i}} = \mathsf{Sign}^B_{sk_B}[M_{c_i}]$ and sends it to the adversary $\mathcal{A}$. We have that $\mathcal{B}$ performs a correct simulation as $\mathcal{S}$ in generating recharge tokens.*

- Money Token. *The adversary acts as $\mathcal{R}$ and makes a money token query to C on the adaptively chosen M_{s_i}. Suppose it denotes monetary value V_{s_i}. If $B_C < V_{s_i}$, $\mathcal{B}$ outputs* reject. *Otherwise, $\mathcal{B}$ responds by querying the signature on M_{s_i} to the Wat scheme. Let the signature be $\mathsf{Sign}^W_{sk_W}[M_{s_i}] = (\sigma_1, \sigma_2)$ and $M_{s_i} = m_1 m_2 \cdots m_n$. $\mathcal{B}$ randomly choose $t'_1, t'_2, \cdots, t'_{n-1} \in \mathbb{Z}_p$ and sets*

$$s_{m_1,1} = g^{t'_1}, s_{m_2,2} = g^{t'_2}, \cdots, s_{m_{n-1},n\text{-}1} = g^{t'_{n-1}}, s_{m_n,n} = \frac{\sigma_1}{g^{\sum_{i=1}^{n-1} t'_i}}, s_r = \sigma_2.$$

$\sigma_{M_{s_i}} = (s_{m_1,1}, s_{m_2,2}, \cdots, s_{m_n,n}, s_r)$ is sent to the adversary $\mathcal{A}$ as the money token on M_{s_i}.

There must exist universally random $t_1, t_2, \cdots, t_n \in \mathbb{Z}_p$ such that

$$\begin{array}{rcl} s_{m_1,1} & = & g^{t'_1} = g_2^{\beta}(u_0 u_1^{m_1})^r \cdot g^{t_1} \\ s_{m_i,i} & = & g^{t'_i} = (u_i^{m_i})^r \cdot g^{t_i} \ \ \text{for all } i = 2, 3, \cdots, n\text{-}1 \\ s_{m_n,n} & = & \dfrac{\sigma_1}{g^{\sum_{i=1}^{n-1} t'_i}} = (u_n^{m_n})^r \cdot g^{t_n}. \end{array}$$

Therefore, we have that $\mathcal{B}$ performs a correct simulation as C in generating money tokens.

- Recharge On C. *The adversary acts as $\mathcal{S}$ and makes a recharge request to C. Without loss of generality, let the counter C_1 be $C_1 = i$ and the message outputted from C be $M_{c_i} = ID_C \parallel i$. $\mathcal{B}$ checks that τ is a valid signature on M_{c_i} using the public key pk_B, and then falls into different results.*

  – *$\tau = \mathsf{Sign}^B_{sk_B}[M_{c_i}]$ and τ is queried by $\mathcal{B}$ from the BLS scheme. $\mathcal{B}$ increases its current balance $B_C := B_C + V_c$ and the counter C_1 by one;*

  – *$\tau = \mathsf{Sign}^B_{sk_B}[M_{c_i}]$ and τ is forged by the adversary $\mathcal{A}$. $\mathcal{B}$ stops this simulation and outputs (M_{c_i}, τ) as the forged signature to break the BLS scheme.*

  – *$\tau \neq \mathsf{Sign}^B_{sk_B}[M_{c_i}]$. $\mathcal{B}$ rejects the recharge query, increases the counter C_1 by one and queries the signature on M_{c_i} to BLS. Let the signature be $\mathsf{Sign}^B_{sk_B}[M_{c_i}]$.*

  We have that $\mathcal{B}$ performs a correct simulation as C in recharging.

- Increase On B_R. *The adversary acts as C and makes a spend request to $\mathcal{R}$. $\mathcal{B}$ simulates the same as the Step 7 and Step 9. Let the adversary repeat q_4 times in this phase. If the money token $(M_{s_i}, \sigma_{M_{s_i}})$ is correct, store it.*

- Leakage Query. *The adversary queries leakages from C. Observe that the leakage happens only in Step 4, Step 6, and Step 8, when only accessed data in computation leaks information. Let L be all accessed data during these steps. L is specified as follows.*
  - **Step 4.** *In this step, only C_1, N_1 are accessed. $\mathcal{B}$ sends them to the adversary;*
  - **Step 6.** *In this step, C_1, B_C, Q_{C_1} are accessed. We have that $\mathcal{B}$ got $Q_{C_1} = \mathsf{Sign}^{B}_{sk_B}[M_{c_i}]$ from the above query phase. $\mathcal{B}$ sends them to the adversary;*
  - **Step 8.** *In this step, only C_2, N_2 are accessed besides all elements to compose $\sigma_{M_{s_i}}$. $\mathcal{B}$ sends them to the adversary;*

  We have that $\mathcal{B}$ performs a correct simulation associated with unbounded leakage from C.

Win. *If $\mathcal{A}$ wins the game, there must exist one additional signature (stored in $\mathcal{R}$) which was not generated by the challenger. Otherwise, we have $V_R = V_C$. Without loss of generality, let $(M_{s_1}, \sigma_{M_{s_1}})$ be the additional signature forged by the adversary. $\mathcal{B}$ sets $(M_{s_1}, \sigma_{M_{s_1}})$ as the forged signature to break the Wat scheme.*

This completes our simulation proof. We have proven Theorem 1.

6. Conclusion

Stored-Value Cards (SVCs) are light-weight devices with weak computation capability, and the system for them should be leakage-resilient against side-channel attacks. We proposed an efficient and secure scheme for SVCs based on message authentication codes and digital signatures. With the help of pre-computations, only simple bitwise comparisons and arithmetical operations are required in SVCs, and our scheme for SVCs are secure with unbounded computational leakage resilience. Our scheme is novel for electronic payments capturing both computational efficiency and security against side-channel attacks.

References

[1] Clemons, E.K., Croson, D.C., Weber, B.W.. Reengineering money: The mondex stored value card and beyond. In: HICSS (4) 1996. IEEE; 1996, p. 254–261.

[2] Garcia, F.D., de Koning Gans, G., Muijrers, R., van Rossum, P., Verdult, R., Schreur, R.W., et al. Dismantling mifare classic. In: Jajodia, S., López, J., editors. ESORICS 2008; vol. 5283 of *LNCS*. Heidelberg: Springer; 2008, p. 97–114.

[3] Quisquater, J.J., Samyde, D.. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In: Attali, I., Jensen, T.P., editors. E-smart 2001; vol. 2140 of *LNCS*. Heidelberg: Springer; 2001, p. 200–210.

[4] Kocher, P.C., Jaffe, J., Jun, B.. Differential power analysis. In: Wiener, M.J., editor. CRYPTO 1999; vol. 1666 of *LNCS*. Heidelberg: Springer; 1999, p. 388–397.

[5] Kocher, P.C.. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Koblitz, N., editor. CRYPTO 1996; vol. 1109 of *LNCS*. Heidelberg: Springer; 1996, p. 104–113.

[6] Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., et al. Lest we remember: Cold boot attacks on encryption keys. In: van Oorschot, P.C., editor. USENIX Security Symposium 2008. LNCS; Heidelberg: USENIX Association; 2008, p. 45–60.

[7] Ishai, Y., Sahai, A., Wagner, D.. Private circuits: Securing hardware against probing attacks. In: Boneh, D., editor. CRYPTO 2003; vol. 2729 of *LNCS*. Heidelberg: Springer; 2003, p. 463–481.

[8] Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In: Naor, M., editor. TCC 2004; vol. 2951 of *LNCS*. Heidelberg: Springer; 2004, p. 258–277.

[9] Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.. Private circuits ii: Keeping secrets in tamperable circuits. In: Vaudenay, S., editor. EUROCRYPT 2006; vol. 4004 of *LNCS*. Heidelberg: Springer; 2006, p. 308–327.

[10] Ghosh, S., Alam, M., Chowdhury, D.R., Sengupta, I.. Parallel crypto-devices for gf(p) elliptic curve multiplication resistant against side channel attacks. Computers & Electrical Engineering 2009;35(2):329–338.

[11] Bellare, M., Canetti, R., Krawczyk, H.. Keying hash functions for message authentication. In: Koblitz, N., editor. CRYPTO 1996; vol. 1109 of *LNCS*. Heidelberg: Springer; 1996, p. 1–15.

[12] Boneh, D., Lynn, B., Shacham, H.. Short signatures from the weil pairing. J Cryptology 2004;17(4):297–319.

[13] Waters, B.. Efficient identity-based encryption without random oracles. In: Cramer, R., editor. EUROCRYPT; vol. 3494 of *LNCS*. Heidelberg: Springer; 2005, p. 114–127.

[14] Guo, F., Mu, Y.. Optimal online/offline signature: How to sign a message without online computation. In: Baek, J., Bao, F., Chen, K., Lai, X., editors. ProvSec 2008; vol. 5324 of *LNCS*. Heidelberg: Springer; 2008, p. 98–111.

[15] Chaum, D.. Online cash checks. In: Quisquater, J.J., Vandewalle, J., editors. EUROCRYPT 1989; vol. 434 of *LNCS*. Heidelberg: Springer; 1989, p. 288–293.
[16] Chaum, D.. Blind signatures for untraceable payments. In: David Chaum Ronald L. Rivest, A.T.S., editor. CRYPTO 1982. New York: Plenum; 1982, p. 199–203.
[17] Camenisch, J., Hohenberger, S., Lysyanskaya, A.. Compact e-cash. In: Cramer, R., editor. EUROCRYPT 2005; vol. 3494 of *LNCS*. Heidelberg: Springer; 2005, p. 302–321.
[18] Au, M.H., Susilo, W., Mu, Y.. Practical anonymous divisible e-cash from bounded accumulators. In: Tsudik, G., editor. Financial Cryptography 2008; vol. 5143 of *LNCS*. Heidelberg: Springer; 2008, p. 287–301.
[19] Even, S., Goldreich, O.. Electronic wallet. In: CRYPTO 1983. New York: Plenum; 1983, p. 383–386.
[20] Chaum, D., Pedersen, T.P.. Wallet databases with observers. In: Brickell, E.F., editor. CRYPTO 1992; vol. 740 of *LNCS*. Heidelberg: Springer; 1992, p. 89–105.
[21] Brands, S.. Untraceable off-line cash in wallets with observers (extended abstract). In: Stinson, D.R., editor. CRYPTO 1993; vol. 773 of *LNCS*. Heidelberg: Springer; 1993, p. 302–318.
[22] Anderson, R., Kuhn, M.. Tamper resistance - a cautionary note. In: The Second USENIX Workshop on Electronic Commerce Proceedings. November 1996, p. 1–11.
[23] Goldwasser, S., Micali, S., Rivest, R.L.. A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput 1988;17(2):281–308.
[24] Boneh, D., Boyen, X.. Short signatures without random oracles. In: Cachin, C., Camenisch, J., editors. EUROCRYPT 2004; vol. 3027 of *LNCS*. Heidelberg: Springer; 2004, p. 56–73.

Fuchun Guo received his B.S. and M.S. degrees from Fujian Normal University in 2005 and 2008, respectively. Now, he is a doctoral student at the School of Computer Science and Software Engineering, University of Wollongong. His research interests include public-key cryptography and network security, in particular, cryptographic protocols and applications.

Yi Mu received his PhD from the Australian National University in 1994. He currently is an associate professor, Head of School of Computer Science and Software Engineering and the director of Centre for Computer and Information Security Research, University of Wollongong. His current research interests include network security, computer security, and cryptography. He has published over 250 research papers. Yi Mu is the editor-in-chief of International Journal of Applied Cryptography and serves as editor for nine other international journals. He is a senior member of the IEEE and a member of the IACR.

Willy Susilo received a Ph.D. in Computer Science from University of Wollongong, Australia. He is a Professor at the School of Computer Science and Software Engineering and the director of Centre for Computer and Information Security Research (CCISR) at the University of Wollongong. He is currently holding the prestigious ARC Future Fellow awarded by the Australian Research Council (ARC). His main research interests include cryptography and information security. He has served as a program committee member in dozens of international conferences. He is a senior member of the IEEE and a member of the IACR.