科技部補助專題研究計畫成果報告
期末報告

防帳卡號被盜之低成本免憑證電子錢網路付款可行性方案研究

計 畫 類 別 ：個別型計畫
計 畫 編 號 ：NSC 102-2221-E-343-004-
執 行 期 間 ：102年08月01日至103年07月31日
執 行 單 位 ：南華大學資訊管理學系

計 畫 主 持 人：周志賢

報 告 附 件 ：出席國際會議研究心得報告及發表論文

處 理 方 式 ：
1.公開資訊：本計畫涉及專利或其他智慧財產權，1年後可公開查詢
2.「本研究」是否已有嚴重損及公共利益之發現：否
3.「本報告」是否建議提供政府單位施政參考：否

中 華 民 國 103年10月31日

中 文 摘 要 ： 安全的電子付款工具在 B2C 或 C2C 電子商務中扮演很重要的角色。

中文關鍵詞： 電子錢，匿名性，匿名撤銷，身分盜用，金融卡盜刷

英 文 摘 要 ： Secure electronic payment instruments play an important role in retail electronic commerce. As in most countries, people in Taiwan often use credit cards for payments on Internet shopping. However, the information contained in the card includes sensitive data like card number, valid date, and CVC (Card Verification Code) which all easily suffer data leakage, or impersonation attacks. This may cause the banks, merchants, or user to suffer serious losses. In this paper, we propose an ID-based untraceable electronic cash without card number or personal information to mitigate the risk of data leakage. Meanwhile, our method also considers preventing the anonymity abuse and adds the function of anonymity revocation through a trust party. In addition, due to the proposed is an ID-based scheme, it has the advantage of PKI (Public Key Infrastructure) free and thus save the certificate management cost.

英文關鍵詞： Electronic Cash, Anonymity, Anonymity Revocation, Identity Theft, Financial Card Fraud

FINAL REPORT

ID-Based Certificateless Electronic Cash on Smart Card

against Identity Theft and Financial Card Fraud

Jue-Sam Chou

July 2014

Abstract

Secure electronic payment instruments play an important role in retail electronic commerce. As in most countries, people in Taiwan often use credit cards for payments on Internet shopping. However, the information contained in the card includes sensitive data like card number, valid date, and CVC（Card Verification Code）which all easily suffer data leakage, or impersonation attacks. This may cause the banks, merchants, or user to suffer serious losses. In this paper, we propose an ID-based untraceable electronic cash without card number or personal information to mitigate the risk of data leakage. Meanwhile, our method also considers preventing the anonymity abuse and adds the function of anonymity revocation through a trust party. In addition, due to the proposed is an ID-based scheme, it has the advantage of PKI (Public Key Infrastructure) free and thus save the certificate management cost.

Keywords: electronic cash, anonymity, anonymity revocation, identity theft, financial card fraud

Content

1 Introduction

Internet shopping nowadays has become an important consumption channel for people's daily life. However, it might make the consumer unsatisfactory about personal data leakage, identity theft and transactions' unsafety. According to an investigation in Taiwanese e-commerce yearly book 2011, the most frequent used payment tools for online shopping are: online credit card payment (74.6%), WebATM (i.e. online Automatic Teller Machine) account transfer (52.8%), physical ATM account transfer (46.6%), payment after shipment (37.5%), convenient store payment (37.5%), respectively. So, there are about three quarters of Taiwanese consumers experiencing online credit card payments. This is because Taiwanese have been familiar with the credit cards payments in physical stores, and the credit card usage is usually encouraged by bonus activities sponsored by the companies. In addition, online credit card payment does not require a card reader while compared with the WebATM account transfers in Taiwan. It only needs the consumers to input card number, CVC (Card Verification Code) and the valid date of the card. However, all these personal information can easily suffer data theft. For example, the customer's personal computer is likely to be embedded with Trojon horses to steal the data, and the keystrokes, communication packets and computer screens are also likely to be skimmed. Moreover, if the merchant websites are not properly managed, the transaction data will easily become the targets for criminals. According to a report from Taiwan Joint Credit Information Center, the percentage of online credit card frauds in the total credit card frauds is 40% in 2009, but it rises to 60% rapidly in the third quarter of 2011. Compared to the serious threat of online credit card frauds, WebATM payment has a lower risk in this aspect. This is because WebATM payment requires the consumer to insert his/her debit card into a card reader, which is connected to his/her own personal computer or notebook, and then enter his/her password through the keypad to enable account transfer. Such a solution builds a defense against Trojon horses stealing users' password [1]. However, users' bank accounts are still clearly transferred on the open network.

The above two most frequently used online payment tools in Taiwan are account-based. Both require the payer to provide personal identifiable information for the financial institutes to confirm the payments. From this, it can be easily seen that the card number / account number and the necessary individual information all need to be transferred on the net. This naturally brings the risk of data leakage or theft. To cope with the problem, untraceable electronic cash (e-cash) which contains no information about individuals was developed to resist both personal data theft and identity theft [2]. It typically composes of a series of random bits and a bank's signature, and therefore cannot be linked to any personal accounts.

However, the digital signature in the e-cash needs to use a public key cryptosystem, such as RSA, the most frequently used scheme. In that, the public key composes of a series of meaningless binary digits. For example,

```
Public exponent:
    0x10001
Modulus:
    13506641086599522334960321627880596993888147560566702752448514
    38515265106048595338339402871505719094417982072821644715513736
    80419703964191743046496589274256239341020864383202110372958725
    76235850964311056407350150818751067659462920556368552947521350
    08528794163773285339061097505443349998111500569772368909275623
```

In commercial applications, a meaningless RSA public key requires using a meaningful certification to link to a user's identity. It requires additional overhead to handle the certification management. For this reason, an ID-based cryptosystem was proposed which no longer demands the certification because of its using user identification as the public key. For example

```
Public key:
    Alice@xxx.com
```

The advantages of an ID-based public cryptosystem are that it requires neither PKI nor certification management, such as certificate enquiry, revocation, renewal, and so on. The purpose of this paper is to implement such an ID-based untraceable electronic cash to prevent personal data leakage and achieve more efficient public key usage. The organization of the remaining paper is as follows. The background knowledge is introduced in Section 2. Some security rationales are described in Section 3. Section 4 shows our proposal. Section 5 discusses its security and performance. Finally, a conclusion is given in Section 6.

2 Background Knowledge of Electronic Cash

Electronic cash (e-cash) like paper money can allow payers to pay without being traced. There have been many cryptographic scientists working within the field of e-cash system design [3-20] since Chaum first proposed the concept in 1982 [3].

From the control viewpoint, e-cash systems fall into two categories: (1) bank-controlled and (2) P2P (peer-to-peer) -distributed. A bank-controlled e-cash system typically contains three roles: customer, bank and merchant, and three protocols: withdrawal, payment, and deposit. As a required function of an untraceable e-cash system, when a customer withdraws e-cash from an issuing bank and pays it to a merchant, and then the merchant deposits it at an acquiring bank, no one can link the e-cash to the customer. This is referred to as *anonymity* or *untraceability*. The main underlying technique is the blind signature scheme. Mondex [21] is one of the bank-controlled systems, produced by National Westminster Bank in the U. K., and has a great success in 1990s. It has absolute anonymity, but at the same time opens a perfect channel for criminals to untraceably transfer their illegal funds. On the other hand, P2P distributed e-cash system kills the role of central bank or authority and thus reduces the expensive bank-processing cost. Bitcoin [22-24] is a famous P2P distributed e-cash system and has been reigning over the cyberspace in the real world. All activities, including coin mintage, coin validness check, double-spending check, are done through the cooperation of the peer nodes on the Bitcoin P2P network. By just generating a public/private key pair, a user can join the Bitcoin network, and use the public key as his/her pseudonym to mine, exchange, buy, and spend the Bitcoin without revealing his/her real identity and location. Nevertheless, some privacy issues emerge because of the public transactions. For instance, one may trace sensitive transactions or de-anonymous social network data through using network topology, and thus violating users' privacy [25, 26].

To be a sound cash system, some essential properties should be centralized.

- **Verifiability**: The e-cash validity can be publicly examined.
- **Unforgeability:** E-cash should be issued only through specialized procedures. No one, including banks, can forge e-cash by any other ways.
- **Untraceability** (or **unlinkability**): It means that no one, including the bank, can know the e-cash owner when the cash is used legally. Although, the bank provides e-cash withdrawal service to its account holder, it cannot link any e-cash to the holder's identity.
- **Double-spending detection**: An e-cash system should prevent e-cash from double spending. If this occurs, the system should be able to get the cheater

efficiently.

- **Anonymity revocation:** When e-cash is illegal used such as, money laundering and tax evading, the system should has the ability to reveal its owner. An e-cash system with anonymity revocation is called fair e-cash system.

For fair e-cash systems [6, 10, 11, 15, 16, 19, 20], an additional trustee is involved in the escrow of some critical information, such as linking the owner's identity to the e-cash. Once a bank or a law enforcement agency requests anonymity revocation, the trustee can reveal the e-cash owner. In other words, the anonymity of e-cash is maintained if the e-cash is used legally, but is revoked if misused.

Pairing-based cryptography is greatly applied in various applications for latest two decades, because it is easier to design an ID-based cryptosystem and requires only about one sixth key length compared to RSA-based cryptosystems. There have been several pairing-based fair e-cash systems proposed. The work of Hufschmitt and Traoré [27] is provably secure, but needs many underlying building blocks (including bilinear pairing, Paillar encryption, double ElGamal encryption, and Fiat-Shamir heuristic) which make it very complicated. Fuchsbauer et al.'s fair e-cash [12], like Hufschmitt and Traoré's, is also a complex construction, because it uses public encryption primitive to achieve e-coin's blindness, and employs both commitment technique and zero-knowledge proof to ensure the encrypted content is well-constructed so as to let the inside tracing information can be disclosed when anonymity revocation is demanded. In addition, the underlying signature scheme (constructed from group signature) is also in complex concept. On the other hand, three pairing-based e-cash systems, Popescu and Oros' [28], Wang et al.'s [29], and Chen et al.'s [30], make their e-cash include a trustee-issued certificate which can be linked to e-cash owner only by the trustee himself. However, Chen et al. pointed that Popescu and Oros' scheme violates anonymity, and Wang et al.'s has a deficiency in that a malicious user can use an unregistered certificate to withdraw e-cash from a bank. Until now Chen et al.'s scheme [30] is the most efficient pairing-based fair e-cash systems among the above-mentioned, but it has two weaknesses. First, its security is not formally proved. Second, the blind factor used for protecting the message to be signed is always the same if the user employs the same certificate to withdraw e-cash.

3 Security Rationales

This section defines several terminologies used in this paper.

3.1 Intractable Problems and Assumptions

A major goal of cryptographic applications is to create a secure cryptographic scheme such that breaking the scheme can be reduced to solving an **intractable problem**. Formally, problems that can be solved in theory (e.g., given infinite time), but taking too long for their solutions to be useful in practice, are known as **intractable problems**. In complexity theory, a problem is **intractable** if no **probabilistic polynomial-time** (**PPT**) adversary can solve it with **non-negligible** probability. Followings are some intractable problems and assumptions related to elliptic curve cryptography [31, 32] used in this study. In them, we let G be an additive elliptic-curve group with prime order q and a base point P.

Definition 2.1 The Discrete Log (DL) Assumption states that the following problem is (τ, ε)-intractable: given a group $G = \langle P \rangle$ and a random point $Q \in G$, find the integer $a \in Z_q^*$ such that $Q = aP$, by taking at most time τ with a **negligible** probability ε.

Definition 2.2 The Computational Diffie-Hellman (CDH) Assumption states that the following problem is (τ, ε)-intractable: given a group $G = \langle P \rangle$ and two random points aP and $bP \in G$, compute abP, with at most time τ and probability ε, if ε is **negligible**.

Definition 2.2 The Variant Computational Diffie-Hellman (VCDH) Assumption states that the following problem is (τ, ε)-intractable: given a group $G = \langle P \rangle$, and three random points Q, aQ and $bQ \in G$, compute abQ, with at most time τ and probability ε, if ε is **negligible** [33].

Koblitz and Menezes [32] pointed out that the **DL** and **CDH** problems on a sufficiently large group are regarded as classical intractable problems. Rather, the **Decisional Diffie-Hellman (DDH)** problem — given three random points aP, bP, and $cP \in G$, decide whether $c = ab \pmod q$ — is believed to be **intractable** on any suitable group, except for the **gap Diffie-Hellman (GDH) group** in which there exists an efficient **bilinear pairing** function. That is, the **DDH** problem on a **GDH** group can be solved in polynomial time through **bilinear pairing**; we will describe this in the following section.

3.2 Bilinear Pairing

Weil pairing [34, 35] is a tuple (G_1, G_2, q, P, $\hat{e}$) where (G_1, +) and (G_2, $\cdot$) are two cyclic groups of order q, P is a generator of G_1, and $\hat{e}: G_1 \times G_1 \rightarrow G_2$ is a mapping which has the following properties:

1. Bilinearity: If a, b are two integers and $P, Q \in G_1$, one has $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.
2. Non-degeneracy: If P is a generator of G_1, then $\hat{e}(P, P)$ is a generator of G_2.
3. Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in G_1$.

An important result is that **Weil pairing** allows us to determine whether points aP, bP, cP in G_1 satisfies $ab = c \bmod q$, i.e.

$$ab = c \bmod q \quad \text{iff} \quad \hat{e}(aP, bP) = \hat{e}(P, cP).$$

In other words, for solving a **DDH** problem one requires only two evaluations of the **Weil pairing** for points in G_1. That is why the group G_1 is referred to as a **GDH** group because of the difference in difficulty between **CDH** and **DDH** problems in the group.

3.3 BLS Signature

Boneh, Lynn, and Shacham [35] proposed a short signature scheme, **BLS** in brief, from **Weil pairing** in 2001. The signature length is half the size of a DSA signature for a same security level, and the scheme should be constructed over a **GDH** group. **BLS** comprises the following system settings and some algorithms: **KeyGen**, **Sign**, and **Verify**.

- **System settings:** The system parameters (G_1, G_2, P, q, $\hat{e}$) are the same as the ones in Section 2.2, In addition, the system makes use of a hash algorithm H: $\{0, 1\}^* \rightarrow G_1$, mapping a string to a point in G_1.
- **KeyGen algorithm**: This algorithm picks a random number $x \in Z_q^*$ as the signing key and computes the corresponding verification key $X = xP$, which is a point in G_1.
- **Sign algorithm**: To sign on a message $m \in \{0, 1\}^*$, compute signature $S = xH(m)$, and output S.
- **Verify algorithm**: On inputting message m and its signature S, one can use the

verification key X to verify S by examining whether (P, X, $H(m)$, S) is a Diffie-Hellman tuple. In other words, one can check whether the equation $\hat{e}(X, H(m)) = \hat{e}(P, S)$ holds.

The **BLS** signature scheme is provably secure under the intractability of the **CDH** assumption.

3.4 Secure Hash Function

A hash function $H(.)$ is a transformation that takes a variable-size input m and returns a fixed-size string, which is called the hash value h (i.e., $h = H(m)$) or the *digest* of message m. A **secure hash function** [36] must be able to withstand all known types of cryptanalytic attacks. At a minimum, it must have the following properties:

1. **Pre-image resistance**: Given h, it should be computationally infeasible to find any message m such that $h = H(m)$. This concept is called a **one-way property**.
2. **Second-pre-image resistance**: Given an input m_1, it should be computationally infeasible to find another input m_2, where $m_1 \neq m_2$, such that $H(m_1) = H(m_2)$. This property is referred to as **weak collision resistance**.
3. **Collision resistance**: It should be computationally infeasible to find two different messages m_1 and m_2, such that $H(m_1) = H(m_2)$. Such a pair is called a cryptographic hash collision. This property is called **strong collision resistance.**

4 The Proposed Scheme

Our design goals are to propose a fair untraceability e-cash system and to minimize user side computations. The proposed system has four parties: trustee *T*, bank *B*, user *U*, and merchant *M*, and five protocols: license issuing, withdrawal, payment, deposit, and owner tracing. We assume that *T* is a trust third party. (To prevent the collusion, the private key of *T* can be separated into several shares held by different authorities through a share-secrecy technique.) The following details the system initialization and the five protocols. The notations are listed in Table 1 and the protocols are illustrated in Figure 1.

Initialization. The trustee *T* publishes system parameters $\{G_1, G_2, P, q, \hat{e}, H\}$, as defined in Section 3.2. It also publishes a mapping function H_1 which maps $\{0, 1\}^*$ to G_1. In addition, bank *B* registers its private key $x \in Z_q^*$ and its identity with a valid date, $ID_{BV} = ID_B||VD_B$, to the trustee *T*. *B* then obtains $Q_B = H_1(ID_{BV})$, $P_B = xQ_B$ and $\mathrm{SIG}_T(Q_B||P_B)$, where $\mathrm{SIG}_T(Q_B||P_B)$ is the trustee's signature on $Q_B||P_B$. Then, bank *B* makes the information, ID_{BV}, P_B and $\mathrm{SIG}_T(Q_B||P_B)$, public.

Table 1 Notations in the proposed scheme

x	Bank's private key
ID_{BV}	bank's pubic data indicates the bank's identity together with a valid period.
Q_B	$Q_B = H_1(ID_{BV})$ is bank's public data
P_B	$P_B = xQ_B$ is bank's public data
$\mathrm{SIG}_T(.)$	Trustee *T*'s signature on some message
w	A user-chosen license key
Q_L	$Q_L = wP_B$ is a license for e-cash
SK	a session key shared between a user and a merchant in a payment transaction

License issuing protocol. Once a user *U* who is an account holder of a bank *B* wants to employ e-cash as a payment tool, he/she should apply for a license to the trustee *T* in advance. *U* and *T* together do the following steps.

(1) *U* randomly chooses a **license key** $w \in Z_q^*$ and sends it with his/her bank's name to *T* through a secure and authenticated channel (e.g., they have performed mutually authentication and session key exchange in advance.).

(2) *T* fetches the bank's public information ID_{BV}, Q_B and P_B from its database.

(3) T computes a **license** as $Q_L = wP_B$ and then signs the license as $\mathrm{SIG}_T(Q_L)$. T stores $\{U, Q_L\}$ into its database and returns $\{Q_L, \mathrm{SIG}_T(Q_L), ID_{BV}, P_B\}$ to U.

(4) On receiving the message, U confirmes the validness of Q_L by examining the T's signature $\mathrm{SIG}_T(Q_L)$. If it is valid, U stores $\{w, Q_L, \mathrm{SIG}_T(Q_L), ID_{BV}, P_B\}$ into his/her smart card.

Withdrawal protocol. After having a valid **license**, a user U can withdrawal e-cash from his/her bank B. The detail steps between U and B are described below.

(1) U selects a random e-coin c, a random element R' in G_1 and two blind factors $a, b \in Z_q^*$ such that $ab = w \bmod q$, where w is U's **license key**. U then computes a blind message $M = b(H(c||R||ID_{BV}) + R)$ and sends M to B.

(2) B performs blind signing, $S' = xM$, by using its private key x, and then returns the blind signature S' to U.

(3) U unblinds the received S' by computing $S = aS'$ and $R = wR'$, and obtains e-cash $\{c, S, R\}$.

Here, the proposed e-cash verification equation is

$$\hat{e}(H_1(ID_{BV}), S) =? \;\; \hat{e}(Q_L, H(c||R||ID_{BV}))\,\hat{e}(P_B, R). \qquad \ldots\ldots \quad \textbf{Eq.(1)}$$

We show the proof below.

$$
\begin{aligned}
&\hat{e}(H_1(ID_{BV}), S)\\
=\;& \hat{e}(Q_B, S)\\
=\;& \hat{e}(Q_B, aS')\\
=\;& \hat{e}(Q_B, axM)\\
=\;& \hat{e}(Q_B, axb(H(c||R||ID_{BV}) + R'))\\
=\;& \hat{e}(Q_B, wxH(c||R||ID_{BV}))\,\hat{e}(Q_B, wxR')\\
=\;& \hat{e}(wxQ_B, H(c||R||ID_{BV}))\,\hat{e}(xQ_B, R).\\
=\;& \hat{e}(wP_B, H(c||R||ID_{BV}))\,\hat{e}(P_B, R).\\
=\;& \hat{e}(Q_L, H(c||R||ID_{BV}))\,\hat{e}(P_B, R).
\end{aligned}
$$

Payment protocol. After purchasing, a user U wants to pay a merchant M with the e-cash in his/her smart card. Suppose that the e-cash left is \$$d$ and the amount to be paid is \$$m$, where $d \geq m$. U and M will do the followings:

(1) U sends e-cash and the corresponding data, $pmsg = \{c, S, R\ Q_L, \mathrm{SIG}_T(Q_L), ID_{BV}, P_B\}$ to M.

(2) On receiving the message, M confirmes the validness of Q_L by examining T's signature $\mathrm{SIG}_T(Q_L)$. If it is valid, M makes a challenge, K, by randomly selecting an integer $k \in Z_q^*$ and computing $K = kP_B$, and sends K to U.

(3) On receiving the challenge K, user U computes $SK = wK = wkP_B$ and the **owner-signature** $S_m = wH(SK||t||m)$ using **license key** w, where t is the current time. U then decreases e-cash balance in the smart card by \$$m$, stores $\{t, m, K\}$ into the smart card, and sends $\{t, m, S_m\}$ to M.

(4) On receiving $\{t, m, S_m\}$, M computes $Q_B = H_1(ID_{BV})$ and $SK' = kQ_L = kwP_B$. M then verifies whether the payer U is the owner of license Q_L by checking $\hat{e}(P_B, S_m) =? \hat{e}(H(SK||t||m), Q_L)$. If the equation holds, M verifies whether the e-cash is valid by checking $\hat{e}(Q_B, S) =? \hat{e}(Q_L, H(c||R||ID_{BV}))\hat{e}(P_B, R)$. If both checks passed, M stores a **payment record**, $\{c, S, R\ Q_L, \mathrm{SIG}_T(Q_L), ID_{BV}, P_B, k, t, m, S_m\}$, into its database.

License issuing: ***User***		***Trustee T***
1.Selects license key $w \in Z_q^*$.	$\xrightarrow{w,\ BankName}$	
		2.Fetchs the bank's pubic data ID_{BV}, Q_B, P_B.
	$\xleftarrow{Q_L, \mathrm{SIG}_T(Q_L), ID_{BV}, P_B}$	3.Computes $Q_L = wP_B$, $\mathrm{SIG}_T(Q_L)$.
4.Verifies Q_L with $\mathrm{SIG}_T(Q_L)$ and Stores $\{w, Q_L, \mathrm{SIG}_T(Q_L), ID_{BV}, P_B\}$		
Withdrawal: ***User***		***Bank B***
1. Selects $c, R' \in G_1$, $a, b \in Z_q^*$ such that $ab = w \bmod q$, and computes $M = b(H(c\|R\|ID_{BV})+R)$.	$\xrightarrow{M}$	
	$\xleftarrow{S'}$	2.Computes $S' = xM$.
3. Computes $S = aS'$, $R = wR'$, and obtains $e\text{-}cash=\{c, S, R\}$.		
Payment: ***User***		***Merchant M***
(Note: t- current time, m – paid money)		
1.$pmsg = \{c, S, R, Q_L, \mathrm{SIG}_T(Q_L), ID_{BV}, P_B\}$	$\xrightarrow{pmsg}$	
		2. Verifies Q_L with $\mathrm{SIG}_T(Q_L)$. Selects $k \in Z_q^*$, and computes
	$\xleftarrow{K}$	$K = kP_B$.
3.Computes $SK = wK$ and $S_m = wH(SK\|t\|m)$.		
	$\xrightarrow{t, m, S_m}$	4. Computes $Q_B = H_1(ID_{BV})$, $SK = kQ_L$ Verifies $\hat{e}(P_B, S_m) =? \hat{e}(H(SK\|t\|m), Q_L)$, $\hat{e}(Q_B, S) =? \hat{e}(Q_L, H(c\|R\|ID_{BV}))\hat{e}(P_B, R)$, Stores payment record into DB.

Fig. 1. The proposed protocols

Deposit protocol. On the end of a business day, merchant *M* sends **payment record**s in batch to bank *B* for e-cash depositing. For each **payment record,** *B* takes the following actions.

(1) verifies the validness of the license Q_L by examining $\mathrm{SIG}_T(Q_L)$.
(2) fetches the corresponding Q_B and P_B from its database using the received ID_{BV}.
(3) confirms the ownership of the **payment record** by computing $SK = k \cdot Q_L = kwP_B$ and checking to see if $\hat{e}(P_B, S_m) = \hat{e}(H(SK||t||m), Q_L)$ holds.
(4) verifies the validness of the e-cash by evaluating $\hat{e}(Q_B, S) =? \hat{e}(Q_L, H(c||R||ID_{BV}))\,\hat{e}(P_B, R)$.
(5) checks to see if the **payment record** is a duplicate. If so, we impute this misuse to the dishonest *M* who doubly deposits it.
(6) checks if the e-cash is over-spent. This means that *B* will sum all payment amounts relating to the same e-coin *c*; if the total amount is over the face value, we impute this misuse to the dishonest *U* who overspends the e-cash.
(7) If all above checks passed, *B* accepts the **payment record** and credits $\$m$ into the *M*'s account.

Owner tracing protocol. When *e-cash* is overspent or abused by criminals and these misuse behaviors have been determined or are undergoing investigation by the court, *B* or a law enforcement agency can request trustee *T* to revoke the anonymity of the *e-cash*. As we know that *e-cash* must be presented with a valid **license** Q_L, the requestor therefore submits the Q_L of the suspected e-cash to *T*. Upon receiving Q_L, *T* retrieves the corresponding record $\{U, Q_L\}$ from its database and successfully reveals the owner of Q_L.

5 System Analysis and Evaluation

We analyze the proposed system in terms of privacy, security, system functions and fraud prevention.

5.1 Privacy and Security Analysis

Regarding the privacy and security of the proposed e-cash, the following seven questions must be answered.

Question 1. (Anonymity Issue) Can a bank link a specific user to e-cash {c, S, R} between or after a withdrawal process?

In a withdrawal process, bank B first authenticates user U as its account holder to provide subsequent withdrawal services and finally debit U's account. Then, if B can recognize any data items in the yielded e-cash, say c, S or R (all items indeed *are not* revealed to B in the withdrawal process.), it would be able to link the e-cash to the user U. From this observation, we must examine the data items one by one as follows:

(1) Item c cannot be recognized as it is hidden in the one-way hash function $H(.)$ and shuffled by both U's one-time random numbers: secrecy b and blind factor R', i.e., c is transformed into a blind message $M = b(H(c||R||ID_{BV}) + R)$.

(2) Items S and R cannot be known by B, since U does not reveal them in and after the withdrawal process.

Question 2. (Anonymity Issue) Is a bank B able to link the returned e-cash, i.e. a payment record, $\{c, S, R\ Q_L, SIG_T(Q_L), ID_{BV}, P_B, k, t, m, S_m\}$, to any previous withdrawal transcript, $\{M, S'\}$, and thus link it to the identity of user U?

We examine each data item in a payment record as follows.

(1) Item c is just a random string and cannot be linked to any withdrawal transcripts.

(2) Item S ($=aS'$) reveals nothing about S', since the randomly chosen integer a makes the variable S uniformly distributed. Thus S cannot be linked to any specific S'.

(3) Item R ($=wR'$) reveals nothing, since U randomly chooses point R' makes R uniformly random, and thus cannot be related to any withdrawal transcripts.

(4) The other items $\{Q_L, SIG_T(Q_L), ID_{BV}, P_B, k, t, m, S_m\}$ are never seen in the withdrawal message flows, so it cannot be linked to any previous withdrawal transcripts as well.

To summarize the analysis of the above two questions, we conclude that the bank cannot link e-cash to any particular user. Therefore, we claim that the proposed e-cash system possesses untraceability and thus assure users' privacy.

Question 3. (Unforgeability Issue) Can user U forge e-cash by only using his/her registered license Q_L without the bank's involvement?

In this case, the user U does not have any advantage since if he/she did so, he/she would be traced through the disclosure of Q_L. However, criminals (or malicious customers) might use a dummy account to gain some advantages. If a criminal registers a license key w on the trustee, and thereby obtains a valid license $Q_L = wP_B = wxQ_B$, can he successfully forge valid e-cash $\{c^*, S^*, R^*\}$ by himself/herself? According to our protocol, the e-cash must pass the e-cash verification of equation **Eq.(1)**, i.e. $\hat{e}(H_1(ID_{BV}), S^*)$ should be equal to $\hat{e}(Q_L, H(c^*\|R^*\|ID_{BV}))\,\hat{e}(P_B, R^*)$. We give the derivation in the following.

$$
\begin{aligned}
& \hat{e}(Q_L, H(c^*\|R^*\|ID_{BV}))\ \ \hat{e}(P_B, R^*) \\
=\ & \hat{e}(wxQ_B, H(c^*\|R^*\|ID_{BV}))\ \ \hat{e}(xQ_B, R^*) \\
=\ & \hat{e}(Q_B, wxH(c^*\|R^*\|ID_{BV}))\ \ \hat{e}(Q_B, xR^*) \\
=\ & \hat{e}(Q_B, wxH(c^*\|R^*\|ID_{BV})) + xR^*) \\
=\ & \hat{e}(H_1(ID_{BV}), x(wH(c^*\|R^*\|ID_{BV}) + R^*))
\end{aligned}
$$

should be equal to $\hat{e}(H_1(ID_{BV}), S^*)$.

Thus, if a malicious U first chooses an integer for c^* and a G_1 element for R^*, then the value S^* should be equal to $x(wH(c^*\|R^*\|ID_{BV}) + R^*)$. However, U does not have the knowledge of bank B's private key x. So U is unable to forge a valid S^* to satisfy the verification equation. From another viewpoint, how about the malicious U first determines c^* and S^* and then try to find a valid R^* to satisfy $R^* = S^* - x(wH(c^*\|R^*\|ID_{BV}))$. It is obvious that finding R^* is hard to due to the one-way properties of the secure hash function and that U has no information about B's private key x.

Question 4. (Unforgeability Issue) Can bank B make e-cash by itself ?

If bank B collects enough spent e-cash including valid license Q_L, can it use them and B's private key x to forge *e-cash* $\{c^*, S^*, R^*\}$? We also observe the equation expansion in the question 3. When malicious B first determines an integer as c^* and a G_1 element as R^*, S^* cannot be computed as $w(xH(c^*\|R^*\|ID_{BV}) + R^*)$ due to that B has no knowledge about w. That is, B cannot extract a license key w from any collected license Q_L (= wP_B) due to **DL** assumption.

Question 5. (Unforgeability Issue) Can an adversary forge valid e-cash ?

From Questions 3 and 4, we can easily see that neither a bank which only knows x nor a user who only knows w can successfully forge e-cash.

Question 6. Can an adversary reuse an eavesdropped payment transcript to pay the e-cash?

For this question, we argue that only the e-cash owner with a **license key** can generate a valid **owner-signature** S_m for the payee's one-time random challenge K. More specifically, S_m is analogous to the **BLS** signature where the **license key** is the signing key and the **license** is the verification key. The signature verification tuple $\{P_B, Q_L, H(SK||t||m), S_m\}$ can be seen as a **VCDH** $\{P_B, Q_L = wP_B, H(SK||t||m) = vP_B, S_m = wvP_B\}$ tuple for some integer v. Thus, we conclude that the **owner-signature** S_m is as secure as the **BLS** signature which is provably secure.

Question 7. Can an adversary deposit the e-cash eavesdropped from a payment transcript to his/her bank account?

For this, we argue that only the merchant who really participates in the payment transaction can prove that he/she is the payee, because only the true payee knows the discrete logarithm of K in the transcript (This security is based on **DL** assumption.).

5.2 Function Evaluation

We discuss our e-cash system in the features of anonymity, verifiability, unforgeability, bank-off-line, divisibility, anonymity revocation, over-spending prevention, and double-depositing prevention. According to the analysis in Sec. 5.1, our system possesses **anonymity** and **unforgeability** and is also **anonymity-revocable** through the owner-tracing protocol. In addition, it does not need an on-line bank when a payer pays. It therefore is a **bank-off-line** system. Furthermore, also according to the security analysis, our system can prevent **double-depositing**. As for the function of **over-spending prevention**, the following three assurances can guarantee the tracing of the over spenders.

1. Our e-cash is **unforgeable**. This implies that no one, including a valid user and the bank, can forge e-cash (see the analysis of Question 3 through 5 in Section 5.1). Valid e-cash must be issued only through a legal withdrawal process.
2. The payer must be the e-cash owner in a payment, because only the owner can generate the **owner-signature**, the response to the payee's random

challenge (see the analysis of Question 6).

3. The merchant who can present a valid **payment record** must be the true payee in the payment. (Also see the analysis of Question 7)

Under these three guarantees, when e-cash returns, the bank can believe that this e-cash must be spent by its owner, and that the owner can be traced when needed.

Finally, we discuss the **divisibility** of our e-cash system. For this issue, we adopted a user-self-control approach like the one in the works of Chaum [4] and Fujisaki and Okamoto [19]. We believe that in general, a customer will honestly spend his/her e-cash instead of overspending it. This stems from the fact that e-cash payment is a kind of micro payment, and a user will not take the risk of losing his/her credit for such a small amount of money. Moreover, the above three assurances also guarantee the correct accumulation of the spent money related to a same c. Once overspending occurs, the user will inevitably be traced and his/her credit will be broken.

5.3 Computational Load

In Table 2, we compare the proposed license-issuing, withdrawal, and payment protocols with those of Chen et al.'s [30] in computational load. In the tables, "**P**" indicates a pairing computation, "**M**" a scalar multiplication (which repeatedly adds an elliptic-curve point for specific times, e.g., cP is $P + P + \ldots + P$, by adding point P totally c times), "**H**" a hash computation, and "**E**" a symmetric encryption. For computational comparison, we adopt the BLS signature scheme which needs 1**M** and 1**H** for signing, and 2**P** for verifying as our trustee's signature $SIG_T(.)$. Table 2 shows the comparison results. As we know that the pairing is expensive in computation time. The cost of a pairing is about 7.5 times of a scalar multiplication on a 3.0GHz Intel Pentium 4 [39], and 22 ~ 38 times on ATmega 128L [40, 41]. The other computations like modular addition, modular multiplications, and elliptic-curve point addition are minor while compared to paring and scalar multiplication. We thus ignore them in the comparison.

Table 2. Computational load comparison

		Chen et al.'s[30]	*Ours*
License Issuing	*Trustee*	1**E** + 3**M** + 1**H**	2**M** + 1**H**
	User	3**P** + 1**M** +1**H**	2**P** + 1 **H**
Withdrawal	*Bank*	3**P** + 3**M** + 1**H**	1**M**
	User	3**P** + 4**M** + 1**H**	3**M** + 1 **H**
Payment	*Merchant*	3**P** + 2**M** + 1**H**	7**P** + 2**M**
	User	0	2**M**

From Table 2, we see that our license-issuing and withdrawal protocols are better than Chen et al.'s in computational time. Especially, a user needs not do the time-consuming pairing in withdrawal phase. Our payment protocol needs 2 scalar multiplications for the user and 4 more pairings for the merchant. This is because our payment protocol can resist an adversary reusing any e-cash he eavesdropped while Chen et al.'s scheme cannot. To sum up, our design achieves the goal that the computation at the user side is minimized much more.

6 Implementation

We use ASUS S400CS notebook with CPU Intel i5-3317U, 1.7GHz and 4G DDR3 memory. The OS is Linux Ubuntu 10.0. Then we refer the website of Stanford Pairings Based Crypto (PBC) to build PBC library. Before building PBC library, one must install package M4, GMP library, flex, and bison through “apt-get install” tool. Then download PCB source codes from the web, extract them and build the library as follows.

```
./configure
make
make install
```

To verify the library, one can build a sample program, BLS signature.

```
gcc bls.c -L. -I/usr/local/include/pbc -lpbc -lgmp
```

Then run the BLS signature and obtain the result as follows.

```
yalin@Yalin-S400CA:~/libpbc/example$ a.out < ../param/a.param
Short signature test
system parameter g =
[5112673021108360787783600100271312131136706540265724662989199565874807981701206
279998578005499322237979804815518272161767325759937954811376023943882181005,
21442544563169537530046933119761045468213139925482358330616788072549833018612O
29970650502363895770321072392955136442921813634889819631373190805855420091 73]
private key = 542559187083824065163733473754519256747925069671
public key =
[5208226549803109552217531532344822592362038181775703033479470033053113721729301
37613203958402806165826726198848453259849535723823817689610268754853922 1973,
40698646022856959976462692142717315870129070732593995271604033042771162696630 2
67291137345147779184182528526642802769835915754139433383519764622826135583 33]
message hash =
[630807567735233679090628126329498672966163176938701969125777863072058930197565
924763228068451405757172376428711265387073482683992998658471098924482627 3472,
6142454701008503581188871307879701426487817074338647796416651458053866219368 26
5234706270304034263890451990757541502223250918907047682554979511124774751109]
```

signature =

[2461739398160868885226508507432090961065486611887801580584389950705192164362026264713924158259449199309930555941863532657852544755708770237760623103077942,

1395612829302401228241080322404480002396077105728756583168513429478170540170889284679269142852291134118500899592793476922919176893432610948195432736128712]

compressed =

2F00BB45689DA4706D5823387524B6806E03F1A31923B67C11AA28E8BA857DCEB21E13F7073D13EC65DC6C444E2EA1CBC3537A787E0267F946FB42E5E9EA6E3600

decompressed =

[2461739398160868885226508507432090961065486611887801580584389950705192164362026264713924158259449199309930555941863532657852544755708770237760623103077942,

1395612829302401228241080322404480002396077105728756583168513429478170540170889284679269142852291134118500899592793476922919176893432610948195432736128712]

f(sig, g) =

[8078865875342867350016437334021096360692083833158234153410577750877622675342304814930020593497277261414623669343859198563687104717448715478287199790485854,

814997088712056477741664337680614175353387303958700919054154956206256803959895532119338489939166808431254756310916125210987674806418235389079113929993007]

f(message hash, public_key) =

[8078865875342867350016437334021096360692083833158234153410577750877622675342304814930020593497277261414623669343859198563687104717448715478287199790485854,

814997088712056477741664337680614175353387303958700919054154956206256803959895532119338489939166808431254756310916125210987674806418235389079113929993007]

signature verifies

x-coord =

2F00BB45689DA4706D5823387524B6806E03F1A31923B67C11AA28E8BA857DCEB21E13F7073D13EC65DC6C444E2EA1CBC3537A787E0267F946FB42E5E9EA6E36

de-x-ed =

[2461739398160868885226508507432090961065486611887801580584389950705192164362026264713924158259449199309930555941863532657852544755708770237760623103077942,

7385097970360911294196701662349569813410806093685451627860139969788305090709333672399356036570371087304654959176788840536358536473884870376729697262096079]

signature verifies on second guess

random signature doesn't verify

To implement our e-cash program, we adopt type A pairings. Type A pairings are constructed on the curve $y^2 = x^3 + x$ over the field F_q for some prime q = 3 mod 4. Both G_1 and G_2 are the group of points E(F_q), so this pairing is symmetric. It turns out #E(F_q) = q + 1 and #E(F_q^2) = $(q + 1)^2$. Thus the embedding degree k is 2, and hence G_T is a subgroup of F_q^2. The order r is some prime factor of q + 1.

7 Conclusion

E-cash is a desire payment tool which links no information about personal account numbers or card numbers and is different from typical financial cards, including credit cards and debit cards. As an interesting result, e-cash solution can resist online payment frauds arisen from account or card number theft / leakage. This paper presented an ID-based certificateless e-cash to attain this goal while considering lower computational load on the user side. Certificateless system can save both the infrastructure building cost and the transaction processing cost. The analysis and evaluation show that the proposed e-cash scheme is of security, privacy preservation, efficiency and is practical for use.

Reference

[1] http://paysecure.com.tw/Default.aspx?tabid=101&mid=482&ItemId=44.

[2] Ashrafi M. Z. and Ng S. K., Privacy-preserving e-payments using one-time payment details, Computer Standards & Interfaces 31 (2009) 321–328, 2009.

[3] D. Chaum, Blind Signatures for Untraceable Payments. In Crypto’82, 199-203.

[4] D. Chaum, A. Fiat, M. Naor, Untraceable electronic cash, Proc. Advances in Cryptology–Crypto’88, 1990, 319–327.

[5] S. Brands, Untraceable off-line cash in wallet with observers, Proc. of Advances in Cryptology–Crypto’93, LNCS 773, Springer, 1993, 302–318.

[6] J. Camenish, U. Maurer, M. Stadler, Digital payment systems with passive anonymity-revoking trustee, Proc. of ESORICS’96, 1996, 33–43.

[7] X. Chen, F. Zhang, S. Liu, ID-based restrictive partially blind signatures and applications, Journal of Systems and Software, 80(2), 2007, 164–171.

[8] Z. Eslami, M. Talebi, A new untraceable off-u electronic cash system, Electronic Commerce Research and Applications, 10(1), 2011, 59–66.

[9] C. Fan, W. Chen, Y, Yeh, Date attachable electronic cash, Computer communications, 23 (4), 425–428, 2000.

[10] Y. Frankel, Y. Tsiounis, M. Yung, Indirect discourse proofs: achieving fair off-line electronic cash, ASIACRYPT’96, LNCS 1163, Springer, 244–251.

[11] Y. Frankel, Y. Tsiounis, M. Yung, Fair Off-Line e-cash Made Easy, Proceedings of ASIACRYPT’98, LNCS 1514, Springer, 257–270.

[12] G. Fuchsbauer, D. Pointcheval, D. Vergnaud, Transferable constant-size fair e-cash, Cryptology and Network Security–CANS’09, LNCS 5888, 226–247.

[13] A. Chan, Y. Frankel, P. MacKenzie and Y. Tsiounis, “Misrepresentation of identities in E-cash schemes and how to prevent it,” ASIACRYPT’96, LNCS 1163, 276–285, 1996.

[14] S.Von Solms and D. Naccache, On blind signatures and perfect crimes, Computer security, 11(6), 581–583, 1992.

[15] E.F. Brickell, P. Gemmell and D.W. Kravitz, Trustee-based tracing extensions to anonymous cash and the making of anonymous change, Proc. of the Sixth Annual Symposium on Discrete Algorithms, ACM/SIAM, 457–466, 1995.

[16] J. Camenish, U. Maurer and M. Stadler, Digital payment systems with passive anonymity-revoking trustee, Proc. of ESORICS’96, 33–43, 1996.

[17] Y. Frankel, Y. Tsiounis and M. Yung, Indirect discourse proofs: achieving fair off-line electronic cash, ASIACRYPT’96, LNCS 1163, 244–251, 1996.

[18] Y. Frankel, Y. Tsiounis and M. Yung, “Fair off-line e-cash made easy,”

ASIACRYPT’98, LNCS 1514, 257–270, 1998.

[19] E. Fujisaki and T. Okamoto, Practical escrow cash system, Proc. of Security Protocols Workshop 1996, LNCS 1189, 33–48, 1996.

[20] M. Gaud, and J. Traoré, On the anonymity of fair offline e-cash, Computer aided verification, LNCS 274, Springer, 34–50, 2003.

[21] F. Stalder, Failures and successes: notes on the development of electronic cash, Inf Soc18(3), 209–219, 2002

[22] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system. Consulted, 1, 2012. Available: http://www.bitcoin.org/bitcoin.pdf.

[23] G. Zorpette, The beginning of the end of cash. Spectrum, IEEE, 49(6), 27-29, 2012.

[24] M. E. Peck, The cryptoanarchists’ answer to cash. Spectrum, IEEE, 49(6), 50-56, 2012.

[25] Miers, I., Garman, C., Green, M., & Rubin, A. D. (2013). Zerocoin: Anonymous Distributed E-Cash from Bitcoin. In IEEE Symposium on Security and Privacy.

[26] Reid, F., & Harrigan, M. (2013). An analysis of anonymity in the bitcoin system. In Security and Privacy in Social Networks (pp. 197-223). Springer New York.

[27] Hufschmitt, E. and Traoré, J., Fair Blind Signatures Revisited, Pairing-based cryptography–Pairing 2007, LNCS 4575, 268–292.

[28] Popescu, C. and Oros, H., An off-line electronic cash system based on bilinear pairings, Systems, Signals and Image Processing, 2007.

[29] Wang, S., Chen, Z. and Wang X., A new certificateless electronic cash scheme with multiple banks based on group signatures. IEEE International Symposium on Electronic Commerce and Security, 2008.

[30] Chen, Y. L., Chou, J. S., Sun, H. M., and Cho, M. H. A novel electronic cash system with trustee-based anonymity revocation from pairing, Electronic Commerce Research and Applications, 10, 6, 2011, 673–682.

[31] Menezes, A.J., Okamoto T. and. Vanstone, S. A, Reducing elliptic curve logarithms to logarithms in a finite field, IEEE transaction on information theory, 39(5), Sep. 1993.

[32] Koblitz, N. and Menezes A. Intractable problems in cryptography. Proc. of the 9th international conference on finite fields and their applications, Aug. 2010.

[33] Maurer, U.M. and Wolf, S, Diffie-Hellman oracles. Advances in cryptology–Crypt’96, LNCS 1109, Springer-Verlag, 268–282, 1996.

[34] Boneh, D. and Franklin, M., Identity-based encryption from the Weil pairing, Proc. of Advances in Cryptology-Crypt’01, LNCS#2139, Springer, 2001, 213–229.

[35] Boneh, D., Lynnm B. and Shacham H., Short signatures from the Weil

pairing, Advances in Cryptology–Asiacrypt'01, LNCS 2248, Springer-Verlag, 514–532, 2001.

[36] Menezes, A. Oorschot P. and Vanstone S., *Handbook of applied cryptography*, CRC Press,323–330, 1996.

[37] Miyaji, A., Nakabayashi, M., & Takano, S. (2001). New explicit conditions of elliptic curve traces for FR-reduction. IEICE transactions on fundamentals of electronics, communications and computer sciences, 84(5), 1234-1243.

[38] Shim, K. A., Lee, Y. R., & Park, C. M. (2013). EIBAS: An efficient identity-based broadcast authentication scheme in wireless sensor networks. Ad Hoc Networks, *11*(1), 182-189.

[39] Miyaji, A., Nakabayashi, M., & Takano, S. (2001). New explicit conditions of elliptic curve traces for FR-reduction. IEICE transactions on fundamentals of electronics, communications and computer sciences, 84(5), 1234-1243.

[40] N. Gura, A. Patel, A. Wander, H. Eberle, S.C. Shantz, Comparing elliptic curve cryptography and RSA on 8-bit CPUs, in: Proceedings of CHES'04, 2004, 119–132.

[41] L.B. Oliveira, A. Kansal, B. Priyantha, M. Goraczko, F. Zhao, SecureTWS: authenticating node to multi-user communication in shared sensor networks, in: Proceedings of IPSN'08, 2009, 289–300.

Appendix

I Main Program

```
//
// ID-Based Certificateless Electronic Cash
//
#include <stdio.h>
#include <pbc.h>
#include <pbc_test.h>
#include <string.h>
#include <time.h>

int main(int argc, char **argv) {
	pairing_t pairing;
	element_t P;
	element_t x, w, y;		//x: bank's private key,	w: user's license key
	element_t Q_B, P_B, Q_L, Y, H_M2G1, SIG_T;
	element_t temp1, temp2;
	char ID_BV[] = "FCBKTWTP20141231";  //Bank's identity
	unsigned char data[256];
	int i, len;

	element_t a,b,c,tmpr, RR, R, M, SS, S, tmpG1, tmpG2;
	element_t in1[2], in2[2];
	element_t k, K, SK, Sm;
	time_t curtime;
	char datetime[32];



	printf("** E_CASH START **\n");
	pbc_demo_pairing_init(pairing, argc, argv);

	element_init_G2(P, pairing);
	element_init_Zr(x, pairing);
```

```
element_init_Zr(w, pairing);
element_init_Zr(y, pairing);
element_init_G2(Q_B, pairing);
element_init_G2(P_B, pairing);
element_init_G2(Q_L, pairing);
element_init_G2(Y, pairing);
element_init_G1(H_M2G1, pairing);
element_init_G1(SIG_T, pairing);
element_init_GT(temp1, pairing);
element_init_GT(temp2, pairing);


printf("\n****************************************");
printf("\n*              Initialization Phase                    *");
printf("\n****************************************\n");
element_random(P);
element_printf("G2 generator P = %B\n", P);

//generate trusstee's private key
element_random(y);
element_printf("Trustee private key y = %B\n", y);

element_pow_zn(Y, P, y);
element_printf("Trustee public Y = %B\n", Y);

//generate bank's private key
element_random(x);
element_printf("Bank private key x = %B\n", x);

//compute Q_B = H(ID_BV)
element_from_hash(Q_B, ID_BV, 16);
printf("Bank ID||VDate = %s\n", ID_BV);
element_printf("Bank public Q_B = %B\n", Q_B);

element_pow_zn(P_B, Q_B, x);
element_printf("Bank public P_B = %B\n", P_B);
```

```
printf("\n****************************************");
printf("\n*              License Issuing Phase                  *");
printf("\n****************************************\n");
//generate User's license key
element_random(w);

printf("\n----------------------------------------------------------------------\n");
printf("User -> Trustee :\n");
element_printf("w = %B\n", w);
printf("BankName = FIRST BANK\n");
printf("----------------------------------------------------------------------\n\n");

//Compute Q_L & SIG_T(H(Q_L))
printf("Trustee fetches bank's identity %s, Q_B, P_B\n", ID_BV);
printf("Trustee Computes license Q_L...\n");
element_pow_zn(Q_L, P_B, w);
//Compute SIG_T(Q_L)
printf("Trustee signs Q_L...\n");
len = element_length_in_bytes_compressed(Q_L);
element_to_bytes_compressed(data, Q_L);
printf("computing the hash of Q_L first\n");
element_from_hash(H_M2G1, data, len);
element_pow_zn(SIG_T, H_M2G1, y);

printf("\n----------------------------------------------------------------------\n");
printf("Trustee -> User:\n");
	//element_printf("Q_L = %B\n", Q_L);
//len = element_length_in_bytes_compressed(Q_L);
	printf("Q_L(%d bytes compressed in HEX) = ", len);
//element_to_bytes_compressed(data, Q_L);
for(i=0; i<len; i++) {
	printf("%02X", data[i]);
}
printf("\n");
	//element_from_bytes_compressed(Q_L, data);
	//element_printf("decompressed = %B\n", Q_L);

len = element_length_in_bytes_compressed(SIG_T);
```

```
        //element_printf("SIG_T = %B\n", SIG_T);
        printf("SIG_T(H(Q_L)) (%d bytes compressed in HEX) = ", len);
element_to_bytes_compressed(data, SIG_T);
for(i=0; i<len; i++) {
        printf("%02X", data[i]);
}
printf("\n");
        //element_from_bytes_compressed(SIG_T, data);
        //element_printf("decompressed = %B\n", SIG_T);

printf("ID_BV = %s\n", ID_BV);

len = element_length_in_bytes_compressed(P_B);
        //element_printf("P_B = %B\n", P_B);
        printf("P_B(%d bytes compressed in HEX) = ", len);
element_to_bytes_compressed(data, P_B);
for(i=0; i<len; i++) {
        printf("%02X", data[i]);
}
printf("\n");
        //element_from_bytes_compressed(P_B, data);
        //element_printf("decompressed = %B\n", P_B);
printf("----------------------------------------------------------------------\n\n");


printf("User verifies SIG_T...\n");
// compute e(SIG_T, P)
element_pairing(temp1, SIG_T, P);
element_printf("computing e(SIG_T, P) = %B\n", temp1);

// compute e(H(Q_L), Y)    should match above
element_pairing(temp2, H_M2G1, Y);
element_printf("computing e(H(Q_L), Y) = %B\n", temp2);

if (!element_cmp(temp1, temp2)) {
        printf("** Signature SIG_T verifies **\n");
        printf("User stores w, Q_L, SIG_T, ID_BV, P_B\n");
} else {
```

```
        printf("*BUG* signature does not verify *BUG*\n");
    }


    printf("\n******************************************");
    printf("\n*              E-Cash Withdrawal Phase                  *");
    printf("\n******************************************\n");

    element_init_Zr(a, pairing);
    element_init_Zr(b, pairing);
    element_init_Zr(c, pairing);
    element_init_Zr(tmpr, pairing);
    element_init_G2(RR, pairing);
    element_init_G2(R, pairing);
    element_init_G2(M, pairing);
    element_init_G2(S, pairing);
    element_init_G2(SS, pairing);
    element_init_G2(tmpG2, pairing);
    element_init_G2(in2[0], pairing);
    element_init_G2(in2[1], pairing);
    element_init_G1(tmpG1, pairing);
    element_init_G1(in1[0], pairing);
    element_init_G1(in1[1], pairing);

    element_random(c);
    element_printf("User generates a random coin c = %B\n", c);
    element_random(b);
    element_printf("User generates a random blind factor b = %B\n", b);
    element_div(a, w, b);                    // b tmpG1, S, tmpr);
    element_pairing(temp1, tmpG1, Q_B);
    /// computinf e(H(c||R||ID), Q_L) * e(R, P_B)= w/a (mod r)
        //element_mul(tmpr, a, b);
        //element_printf("--yalin--examine ab =? w mod r, ab = %B\n", tmpr);
    element_printf("User computes blind factor a =    w/b (mod r) = %B\n", a);
    element_random(RR);
    element_printf("User generates a random point R' = %B\n", RR);
    element_mul_zn(R, RR, w);            //computing R = w * RR
    element_printf("User computes R = w * RR' = %B\n", R);
```

```
//concatenate c||R||ID_BV
len = 0;
len += element_to_bytes(data+len, c);
len += element_to_bytes(data+len, R);
strcpy(data+len, ID_BV); len += strlen(ID_BV);
printf("User concatenates c||R||ID_BV, result lengh = %d\n", len);
element_from_hash(M, data, len);
element_printf("H(c||R||ID_BV) = %B\n", M);
//computes b * (H(c||R||ID_BV)+RR)
element_add(tmpG2, M, RR);
element_mul_zn(M, tmpG2, b);

printf("----------------------------------------------------------------------\n");
printf("User -> Bank:\n");
len = element_length_in_bytes_compressed(M);
element_to_bytes_compressed(data, M);
	printf("Blind message M (%d bytes compressed in HEX) = ", len);
for(i=0; i<len; i++) {
	printf("%02X", data[i]);
}
printf("\n");
printf("----------------------------------------------------------------------\n");

printf("Bank blindly signs on M using its private key x...\n");
element_mul_zn(SS, M, x);

printf("----------------------------------------------------------------------\n");
printf("Bank -> User:\n");
len = element_length_in_bytes_compressed(SS);
element_to_bytes_compressed(data, SS);
	printf("Blind signature S' (%d bytes compressed in HEX) = ", len);
for(i=0; i<len; i++) {
	printf("%02X", data[i]);
}
printf("\n");
printf("----------------------------------------------------------------------\n");
```

```
element_mul_zn(S, SS, a);
element_printf("User unblind S' to get S = %B\n", S);

////////////    VERIFY E-CASH    //////////
// computing    e(S, Q_B)
element_set1(tmpr);
element_mul_zn(tmpG1, S, tmpr);
element_pairing(temp1, tmpG1, Q_B);

// computing e(H(c||R||ID), Q_L) * e(R, P_B)
len = 0;
len += element_to_bytes(data+len, c);
len += element_to_bytes(data+len, R);
strcpy(data+len, ID_BV); len += strlen(ID_BV);
element_from_hash(in1[0], data, len);
element_mul_zn(in1[1], R,      tmpr);
element_mul_zn(in2[0], Q_L, tmpr);
element_mul_zn(in2[1], P_B, tmpr);
element_prod_pairing(temp2, in1, in2, (int)2);
element_printf("e(S, Q_B) = %B\n", temp1);
element_printf("e(H(c||R||ID), Q_L) * e(R, P_B) = %B\n", temp2);
if(!element_cmp(temp1, temp2)) {
        printf("** e-cash signature verifies! ** \n");
} else    {
        printf("* BUG * e-cash signature does not verify *BUG* \n");
}


printf("\n*****************************************");
printf("\n*             E-Cash Payment Phase                    *");
printf("\n*****************************************\n");

element_init_Zr(k, pairing);
element_init_G2(K, pairing);
element_init_G2(SK, pairing);
element_init_G2(Sm, pairing);

printf("\n----------------------------------------------------------------------\n");
```

```
printf("User -> Merchant:\n");
element_printf("c = %B", c);

len = element_length_in_bytes_compressed(S);
element_to_bytes_compressed(data, S);
        printf("S (%d bytes compressed in HEX) = ", len);
for(i=0; i<len; i++) printf("%02X", data[i]);
printf("\n");

len = element_length_in_bytes_compressed(R);
element_to_bytes_compressed(data, R);
        printf("R (%d bytes compressed in HEX) = ", len);
for(i=0; i<len; i++) printf("%02X", data[i]);
printf("\n");

len = element_length_in_bytes_compressed(SIG_T);
element_to_bytes_compressed(data, SIG_T);
        printf("SIG_T (%d bytes compressed in HEX) = ", len);
for(i=0; i<len; i++) printf("%02X", data[i]);
printf("\n");

printf("ID_BV = %s\n", ID_BV);

len = element_length_in_bytes_compressed(P_B);
element_to_bytes_compressed(data, P_B);
        printf("P_B (%d bytes compressed in HEX) = ", len);
for(i=0; i<len; i++) printf("%02X", data[i]);
printf("\n");
printf("----------------------------------------------------------------------\n\n");

element_random(k);
element_printf("Merchant selects a random integer k = %B\n", k);
element_mul_zn(K, P_B, k);
element_printf("Merchant computes challange K = k * P_B = %B\n", K);

printf("----------------------------------------------------------------------\n");
printf("Mechant -> User :\n");
len = element_length_in_bytes_compressed(K);
```

```
element_to_bytes_compressed(data, K);
        printf("K (%d bytes compressed in HEX) = ", len);
for(i=0; i<len; i++) printf("%02X", data[i]);
printf("\n");
printf("------------------------------------------------------------------------\n");

element_mul_zn(SK, K, w);
element_printf("Merchant computes SK = w * K = %B\n", SK);
time(&curtime);
strcpy(datetime, ctime(&curtime));
printf("current date-time t = %s, money to pay m = 35 dollars\n", datetime);
// computing H_M2G1 = H(SK||t||m)    & Sm = w * H_M2G1
len = 0;
len += element_to_bytes(data+len, SK);
strcpy(data+len, datetime); len += strlen(datetime);
strcpy(data+len, "35"); len += 2;
element_from_hash(H_M2G1, data, len);
element_mul_zn(Sm, H_M2G1, w);

printf("------------------------------------------------------------------------\n");
printf("User -> Merchant :\n");
printf("t = %sm = 35\n", datetime);
len = element_length_in_bytes_compressed(Sm);
element_to_bytes_compressed(data, Sm);
        printf("Sm (%d bytes compressed in HEX) = ", len);
for(i=0; i<len; i++) printf("%02X", data[i]);
printf("\n");
printf("------------------------------------------------------------------------\n");

//computing SK
element_mul_zn(SK, Q_L, k);
element_printf("Mechant computes SK = k * Q_L ＝ %B\n", SK);
////////////    VERIFY Sm    //////////
printf("Merchant verifies Owner-Sig Sm...\n");
//computing e(Sm, P_B)
element_set1(tmpr);
element_mul_zn(tmpG1, Sm, tmpr);
element_pairing(temp1, tmpG1, P_B);
```

```
	//computing e(H(SK||t||m), Q_L)
	element_pairing(temp2, H_M2G1, Q_L);
	element_printf("e(Sm, P_B) = %B\n", temp1);
	element_printf("e(H(SK||t||m), Q_L) =   = %B\n", temp2);
	if(!element_cmp(temp1, temp2)) {
		printf("** Owner-Sig Sm verifies! ** \n");
	} else   {
		printf("* BUG * Owner-Sig Sm does not verify *BUG* \n");
	}

	pairing_clear(pairing);
	return 0;
}
```

II Program Running Result

```
yalin@Yalin-S400CA:~/libpbc/e-cash$ run
** E_CASH START **

***************************************
*           Initialization Phase              *
***************************************
G1/G2 generator P =
[4876661194828077595500551795928866267332321065143007465175187173643598753950812681398421984278851553440529132364591339616854692438237717238195246093226019,
7624907113024290572467337074154080607095683938978765939494393164432692743011377835652656612487785814687995814386366195712840421517576597589227053641209074]
Trustee private key y = 635903116473564397461234714926838056598580820189
Trustee public Y =
[8659934005503108359150461449264549172229948421417604481525227654193579660382198959996788104951254034352419133318479413259322456192541331853064127702087429,
4737476795126686606930651616397368656111543063696999107676356768343861131690575939629520305034473183513855579537027342426785674147411312327717643997119129]
```

Bank private key x = 548436541016353765443383790905715704719289819248
Bank ID||VDate = FCBKTWTP20141231
Bank public Q_B =
[713029847271779280323096769403147641409348123244330220332641765374
787060032525655389005574220397918056737922028311685128725323777322
8531910893969880976921,
442831748101359447326640981315911324001950087098091553568178840019
241452928311577339135739527723121704771946208198544459979858731996
2364069382982313463494]
Bank public P_B =
[822579153778586875959118761346659997832837576288619285659826445979
450265255664699622841092742947745555179792643231596178410772234546
8676572551348750794328,
198216487274692478141821197943884179224692312715956417792806291146
068907216708180251602998915397722330558992819366719961513360280704
4341549020317275727872]

****
*   License Issuing Phase   *
****

User -> Trustee :
w = 246256710187041570016323985253510015150319455623
BankName = FIRST BANK

Trustee fetches bank's identity FCBKTWTP20141231, Q_B, P_B
Trustee Computes license Q_L...
Trustee signs Q_L...
computing the hash of Q_L first

Trustee -> User:
Q_L(65 bytes compressed in HEX) =
60DC89487321F22D9A7B188322ADD18FB8467642F120E8F9EF27050CD63D93B0D
16A4D85568F8AFF4B8C2EA066049178CB71B2DF9E33F558D60329FB9CB2ED0801
SIG_T(H(Q_L)) (65 bytes compressed in HEX) =

152D3B136E57932659A1A7DC9312FE3EC131EFFB4DB3DB381F784E826B9379869
3068C5C253148CCE62D597D31D50DFC825A430F60FDC43359D9A5E51F75B79A01
ID_BV = FCBKTWTP20141231
P_B(65 bytes compressed in HEX) =
9D0ED4DD2B0F976C1FD6EF823A092C0767EDE18E080D8BF20029BDA71DEEA987
AA95CAACBF629A9605E9556CFBB8D59EB08261F0C3DDFF8B25BE5EA917863E580
0

User verifies SIG_T...
computing e(SIG_T, P) =
[5344251982763255380745283452976920523843805259243574112622633341621248519223026989526136812361255794571280582837667029039791287433272715368156679759462034,
5179032795553564461910975973605014829250629840786125312595230884330671578917025773946110422629512995470909563900233954127133153681223030543595678289308716]
computing e(H(Q_L), Y) =
[5344251982763255380745283452976920523843805259243574112622633341621248519223026989526136812361255794571280582837667029039791287433272715368156679759462034,
5179032795553564461910975973605014829250629840786125312595230884330671578917025773946110422629512995470909563900233954127133153681223030543595678289308716]
** Signature SIG_T verifies **
User stores w, Q_L, SIG_T, ID_BV, P_B

****
*   E-Cash Withdrawal Phase   *
****
User generates a random coin c =
2298597864356987293464195975179982282794785496 16
User generates a random blind factor b =
3501476210306942294377509459388831582400746341 76
User computes blind factor a = w/b (mod r) =
6722213378227378641241673210397112973331821577 84
User generates a random point R' =
[20833915025122507318086117068835282290495856985018048488722478777 9

4548008228980142063757018436163847097306880904120868005281678497334936244624736380088883,
7005377134480426632423706208387024354090633242067546345120545931929845559008831049201493591013723396039032592114620562835173304292382402142507065117036327]
User computes R = w * R' =
[5520193842251445996926007246143866879587052720025025890720803917011106533182971323714761365001462716228007637850976981834021658815206433605079045627524619,
888081159811895433390970648851634982964706155198386326136796705173026103462060352186779676181762902923917039556573749718857061177779626988911962722605917]
User concatenates c||R||ID_BV, result lengh = 164
H(c||R||ID_BV) =
[7683826385886991767728709157467336469021097368795618652829383870175467156647787581228463547503361828907890201729227121048258190518641710184809118822059561,
7190226600300523181167860568498011651119875804909005736829824687928987400756534174060329737943659047031799049393999878027845390529771516059139128091770895]
User computes blind message M = b * (H(c||R||ID_BV)+R') ...

User -> Bank:
Blind message M (65 bytes compressed in HEX) =
4FBDAF5D643C46B1B02A889EE49AD5A847FCD64A4D7965F32E1A8C7625C07FB8B2CC4D0761CBA816ECA19CFACC2575BF76852E880C9DFC0822C46A266EC0310B00

Bank blindly signs on M using its private key x, generating S'...

Bank -> User:
Blind signature S' (65 bytes compressed in HEX) =
33F429B709506EE625CBD9C4526F85AA7F2FEC6879C59E16148984B8CB898D263FC005A0B23913FB2BC43D9FC50A6D610E1C26D9AB7436768C6D62EC3332E43501

User unblind S' to get S =
[944697364895033223248888705842014480473916566985460977823776808364
209959291656405963349991286962686276524130942811432606948110037700
065076792317476132931,
711927088636478628811160727761905351862137478401698085056450474481
812272267504424271835119388068630223507040902571858540309640977374
0758073932590254169601]
e(S, Q_B) =
[845032566366455626836352445298271610137426121678762566870859020042
023267847307130676739867197366396591422081613052181016310854872260
6076145501165710012942,
549370360600825769328763026291678160945237288831374727209105826036
313128606946226250844247186498005170694393578263779410147953909513
4827387542368343456411]
e(H(c||R||ID), Q_L) * e(R, P_B) =
[845032566366455626836352445298271610137426121678762566870859020042
023267847307130676739867197366396591422081613052181016310854872260
6076145501165710012942,
549370360600825769328763026291678160945237288831374727209105826036
313128606946226250844247186498005170694393578263779410147953909513
4827387542368343456411]
** e-cash signature verifies! **

****

*   E-Cash Payment Phase   *

****

User -> Merchant:
c = 229859786435698729346419597517998228279478549616S (65 bytes
compressed in HEX) =
120995A3E4298933CBF2CC52C323CE63EDDFCE062116785939A40221D853C7B03
0065F376909D3B67BC84501D962CB540B76AEE6072DD83B59734B03E788D84301
R (65 bytes compressed in HEX) =
6966253BCAF9DCC9A5376A153D370E731E771BFBA2BEDFBAA402602B1557DDFA
78087858DC4A67274A7A5731AF81F112B8C243C1BB849FBA6AB71F720B8D360B0
1
SIG_T (65 bytes compressed in HEX) =

152D3B136E57932659A1A7DC9312FE3EC131EFFB4DB3DB381F784E826B9379869
3068C5C253148CCE62D597D31D50DFC825A430F60FDC43359D9A5E51F75B79A01
ID_BV = FCBKTWTP20141231
P_B (65 bytes compressed in HEX) =
9D0ED4DD2B0F976C1FD6EF823A092C0767EDE18E080D8BF20029BDA71DEEA987
AA95CAACBF629A9605E9556CFBB8D59EB08261F0C3DDFF8B25BE5EA917863E580
0

Merchant selects a random integer k =
18649282338605649679143610364743564084513103988
Merchant computes challange K = k * P_B =
[41291354627236316746497648304457611961321258522683543631360665660049382257103419739030813390888399332640549082312155143676705498360180576992643301162861,
53465530175350991030437257259642110227774295984573560087034691980694107355857895954144087764719546828682061842964380707446284461287915398933206245111 25702]

Mechant -> User :
K (65 bytes compressed in HEX) =
07E24784EC17F4DA0C934A4E8867DCA2FE4867850F50BCB11DC2CA8E7568FE57D
FDD59144BDB6DFD4C3BCF0F38D6A6D7BA5C46DC7428AF3200578228634ABF6D0
0

Merchant computes SK = w * K =
[26423321038851296823293693775505587084992503538395200533791885507119179496502794642900535908811629124277046883435659186809120453010004847302515352192 21754,
21684449207916135997362414930259726724242711100497532478618062271450317103530365234998387174104045221832825649148688684556514870840837938864435551367 90277]
current date-time t = Sat Aug 30 15:50:50 2014
money to pay m = 35 dollars

User -> Merchant :
t = Sat Aug 30 15:50:50 2014
m = 35
Sm (65 bytes compressed in HEX) =
A76BC9261FC0B7A5E4F2F145A57B0D8A5256A40FF85B4E9C18EA7DAA5C5611CC5
D64926337DB15D63F3FB3D2EDFC169DFAB3A6DED69660EFBCC95ABC1F9403440
0

Mechant computes SK = k * Q_L =
[264233210388512968232936937755055870849925035383952005337918855071
191794965027946429005359088116291242770468834356591868091204530100
0484730251535219221754,
216844492079161359973624149302597267242427111004975324786180622714
503171035303652349983871741040452218328256491486886845565148708408
3793886443555136790277]
Merchant verifies Owner-Sig Sm...
e(Sm, P_B) =
[425068860357981089596962294169322312370887301815531313712574904540
986324376205445741372728372636055241228170806326010130356638150382
3723879076875469041752,
101108532973727731818722120810025248298489972424833633257428130925
788151386421385521321301487875887982634053133179843308035316947615
4408678199591543736626]
e(H(SK||t||m), Q_L) = =
[425068860357981089596962294169322312370887301815531313712574904540
986324376205445741372728372636055241228170806326010130356638150382
3723879076875469041752,
101108532973727731818722120810025248298489972424833633257428130925
788151386421385521321301487875887982634053133179843308035316947615
4408678199591543736626]
** Owner-Sig Sm verifies! **
Merchant verifies E-Cash...
e(S, Q_B) =
[845032566366455626836352445298271610137426121678762566870859020042
023267847307130676739867197366396591422081613052181016310854872260
6076145501165710012942,
549370360600825769328763026291678160945237288831374727209105826036

313128606946226250844247186498005170694393578263779410147953909513
4827387542368343456411]
e(H(c||R||ID), Q_L) * e(R, P_B) =
[845032566366455626836352445298271610137426121678762566870859020042
023267847307130676739867197366396591422081613052181016310854872260
6076145501165710012942,
549370360600825769328763026291678160945237288831374727209105826036
313128606946226250844247186498005170694393578263779410147953909513
4827387542368343456411]
** E-Cash verifies! **


Keynote Speaker | D
sdiwc.net/conferences/2014/digitsec2014/keynote-speaker/
The International Conference on Digital Security and Forensics (DigitalSec2014)
June 24-26, 2014 | VSB-Technical University of Ostrava, Czech Republic
A post conference tour to Poland.
All registered papers will be included in SDIWC digital library.
DigitalSec
SDIWC
HOME
IMPORTANT DATES
REGISTRATION
SPONSORS
WORKSHOPS
DOWNLOAD A POSTER
CONTACT US
Home
Important Dates
Paper Submission
Camera Ready
Program Committees
Paper Review System
Become a Reviewer
Registration
Keynote Speaker
The last day for the registration and the Camera Ready has been extended to June 16,
Keynote Speaker
Ekaterina Pshehotskaya and Tamara Sokolova
InfoWatch Company, Russian Fedration
Keynote Title: Practical Issues of Clustering Relatively Small Text Data Sets for Business Purposes
EYalin Chen and Jeu-Sam Chou
Nanhua University and C&C Information Security Laboratory, Taiwan
Keynote Title: ID-Based Certificateless Electronic Cash on Smart Card against Identity Theft and Financial Card Fraud
下午 12:41

Jue-Sam Chou & Yalin Chen

Info. Mgmt., Nanhau University

C & C Info. Security LAB.

ChiaYi, Taiwan

Anonymous Electronic Cash Against CNP & Identity Theft

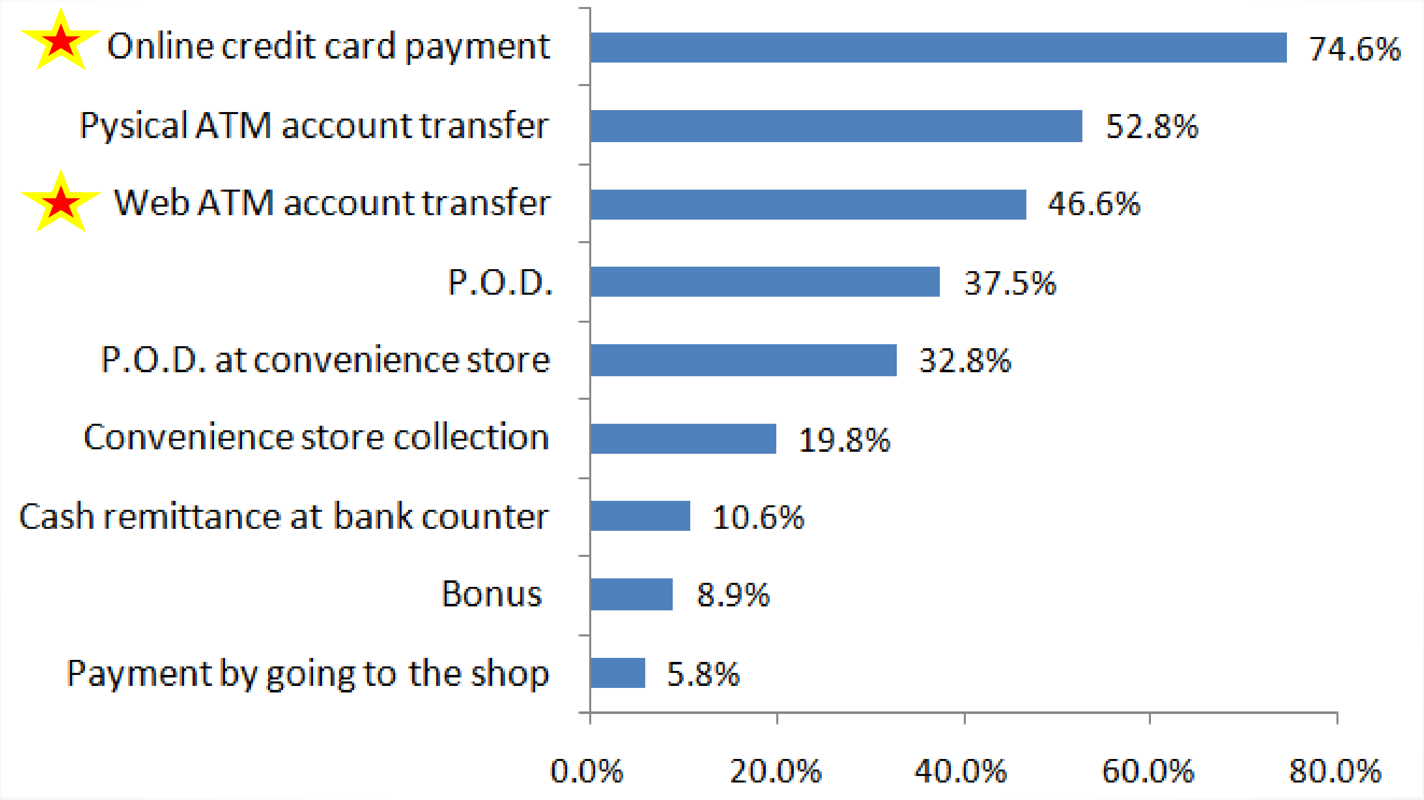
Yalin Chen and Jue-Sam Chou

DigitalSec2014

Agenda

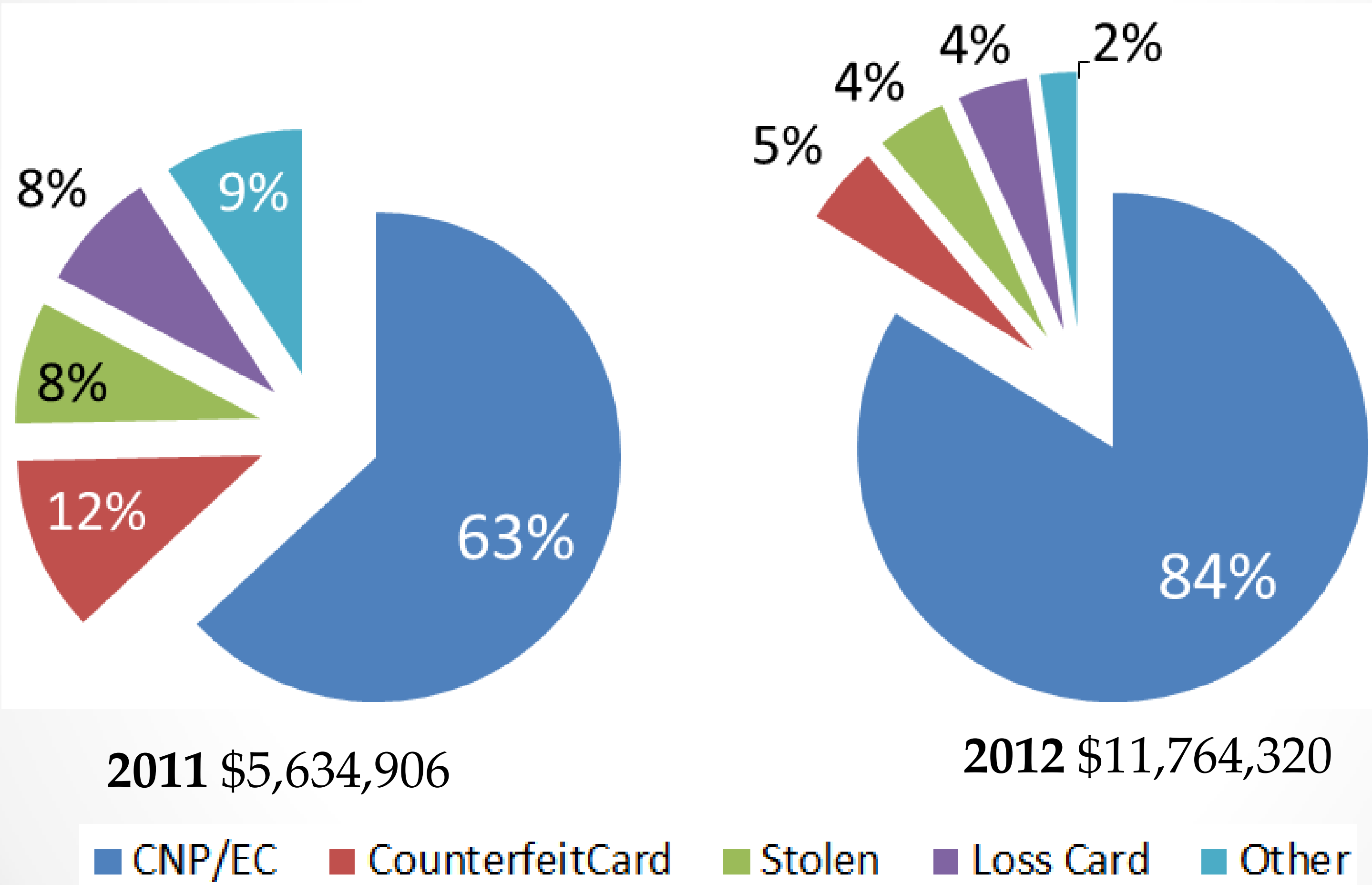
- Part 1 Background and Motivations
- Part II The Proposed Scheme

Taiwanese Payment Methods for Online Shopping, 2012



Data from Taiwan Joint Credit Information Center.

Taiwan Credit Card Frauds



33%

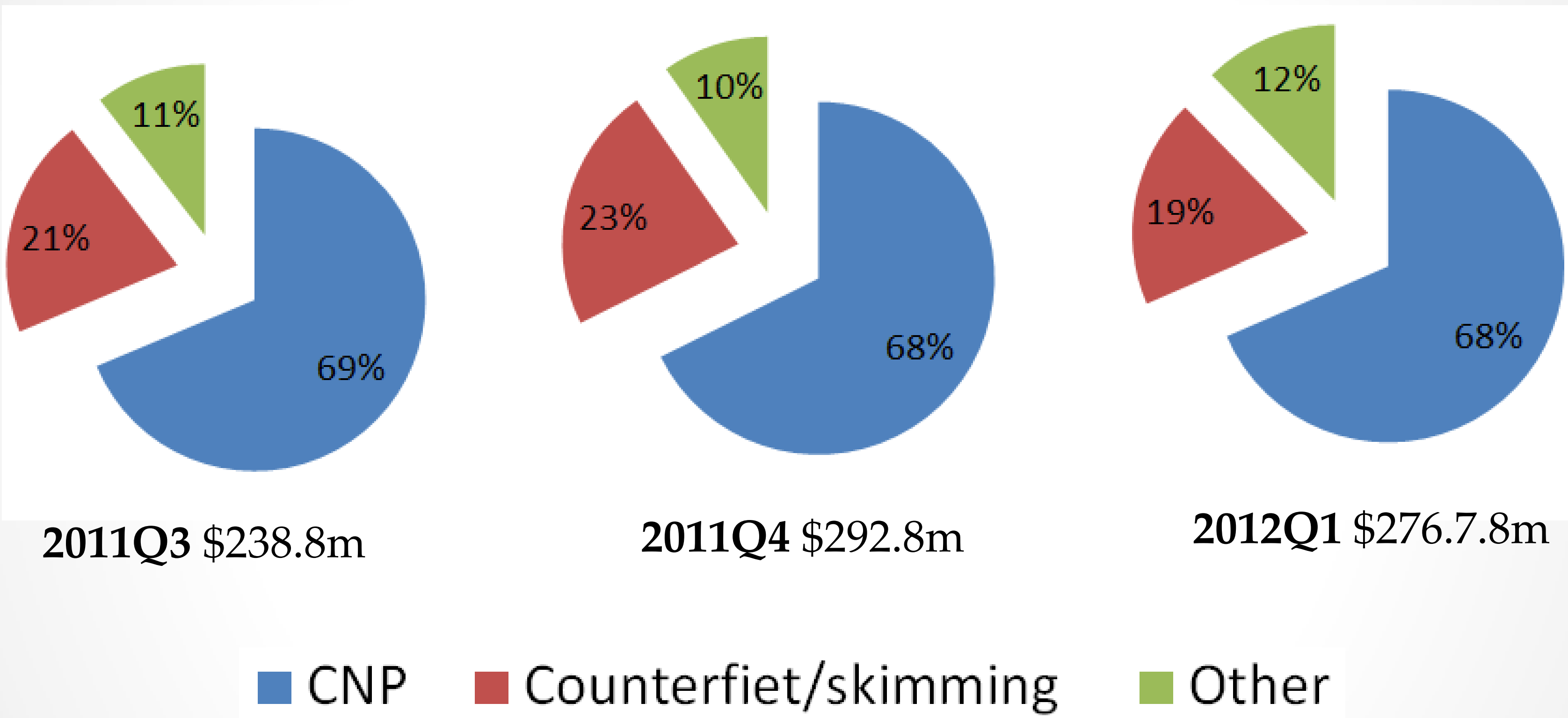
CNP

Data from Taiwan Joint Credit Information Center.

CNP/EC

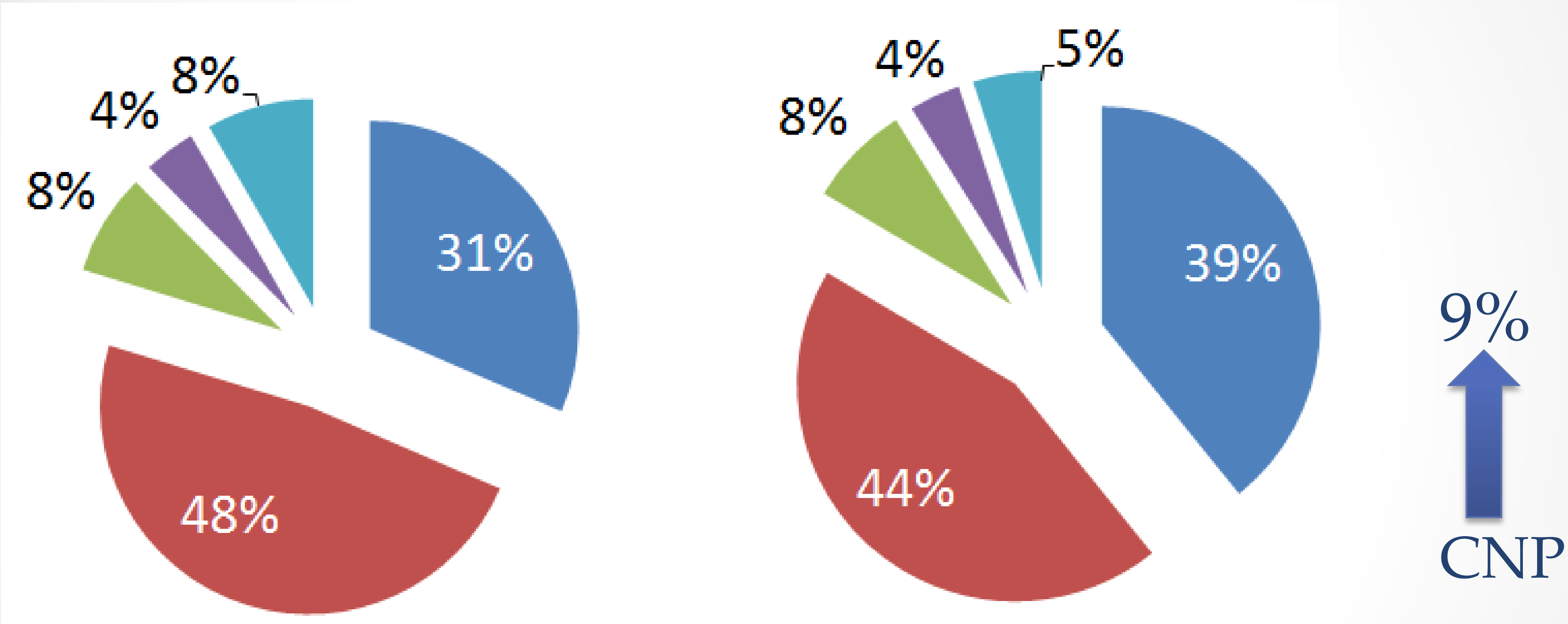
- In a typical **CNP** fraud, for example, a cheater offer just a victim's Card No, Card’s valid date, and CVC to an online merchant; then he can complete a payment transaction without any error alarms.
- **Not-Face-to-Face** payment usually happened in the cyber world.
- This is because these card information are easily be collected by the criminals through
    - Trojan Horse, skimming, hacker’s hacking merchant's web site, DB or illegal deal

Australia Plastic Card Frauds



Data from Australian Payment Clearing Association.

Canada Credit Card Frauds



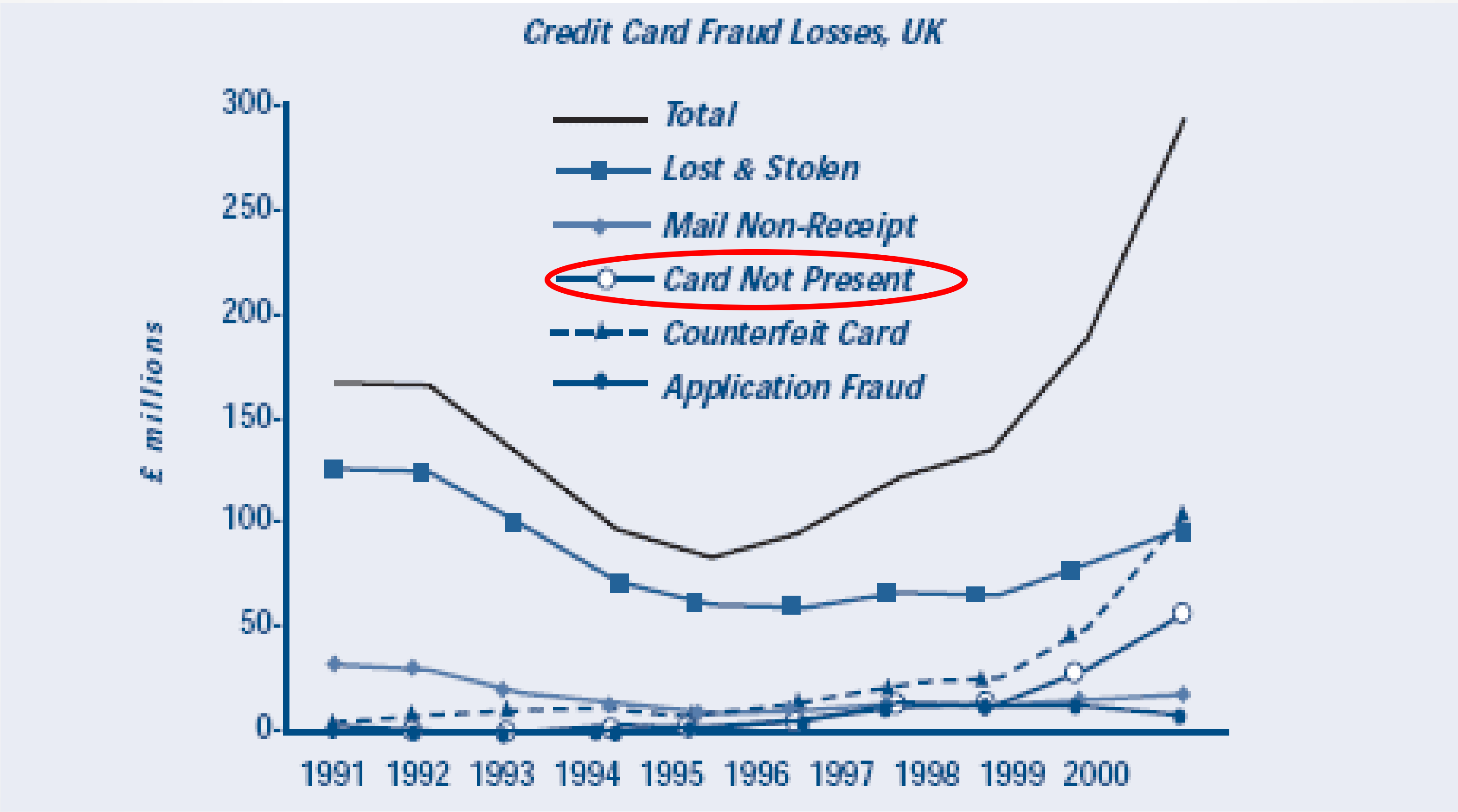
2008 CAD$408m

2009 CAD$358m

CNP Counterfeit Stolen Lost Other

Data from http://www.kubera.cc

UK Credit Card Frauds



http://www.popcenter.org/learning/60steps/index.cfm?stepNum=11

Our Countermeasure

↓

NOT to Present Card No. or Identity

↓

Anonymous Electronic Cash

Anonymous Electronic Cash

- Like paper cash which itself is identifiable
  - For E-Cash, one can identify it by verifying the issuer's digital signature
- Untraceability
  - No one can link a presented e-cash to any particular person; this protects individuals' privacy
- E-Cash is typically a series of meaningless bits which link to nothing,
- Not like a card number which always links to a personal identity.

E-Cash Types

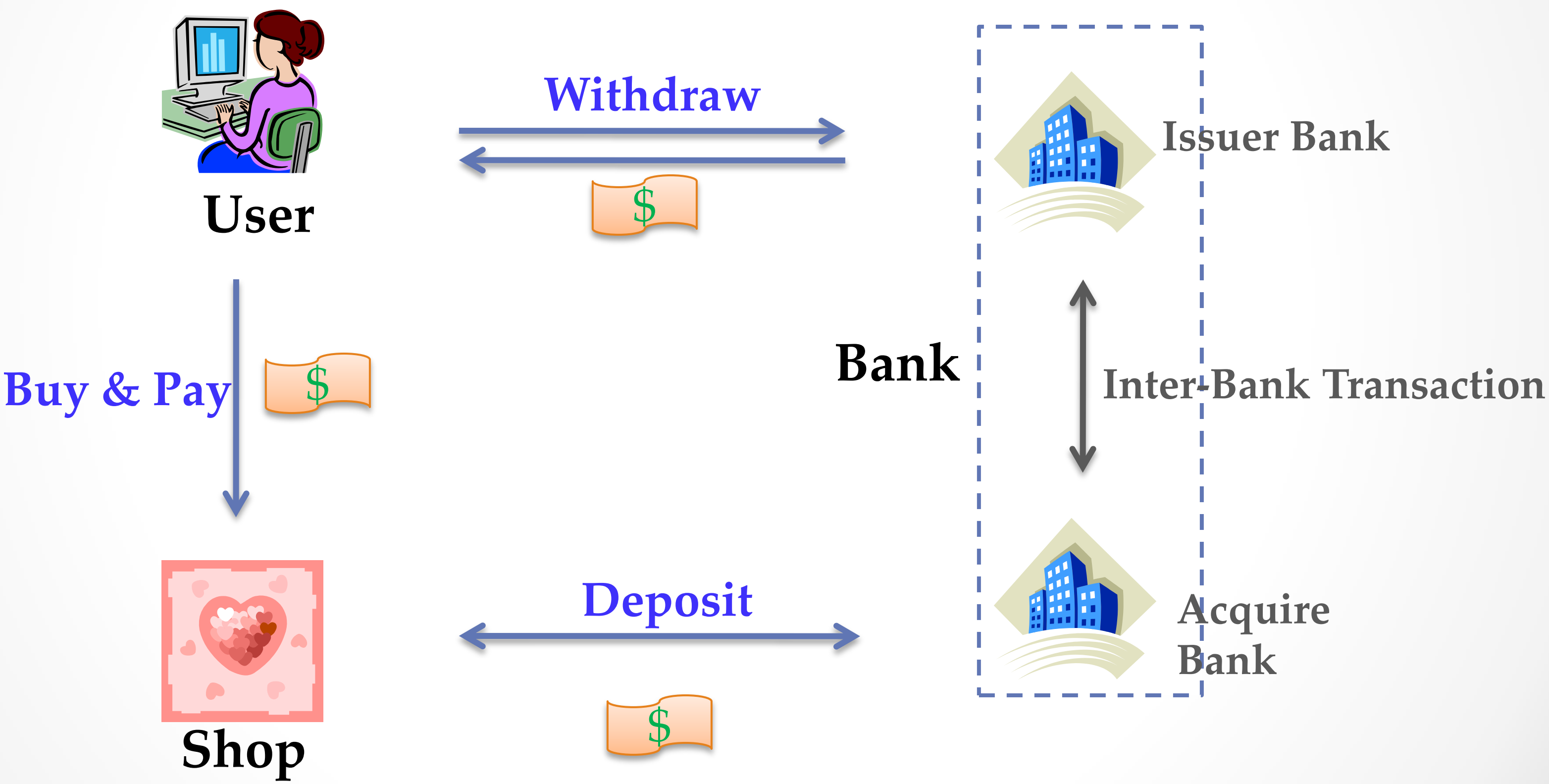
Type I: Bank-controlled E-Cash

- Mondex

Type II: P2P- distributed (Bank-free) E-Cash

- Bitcoin

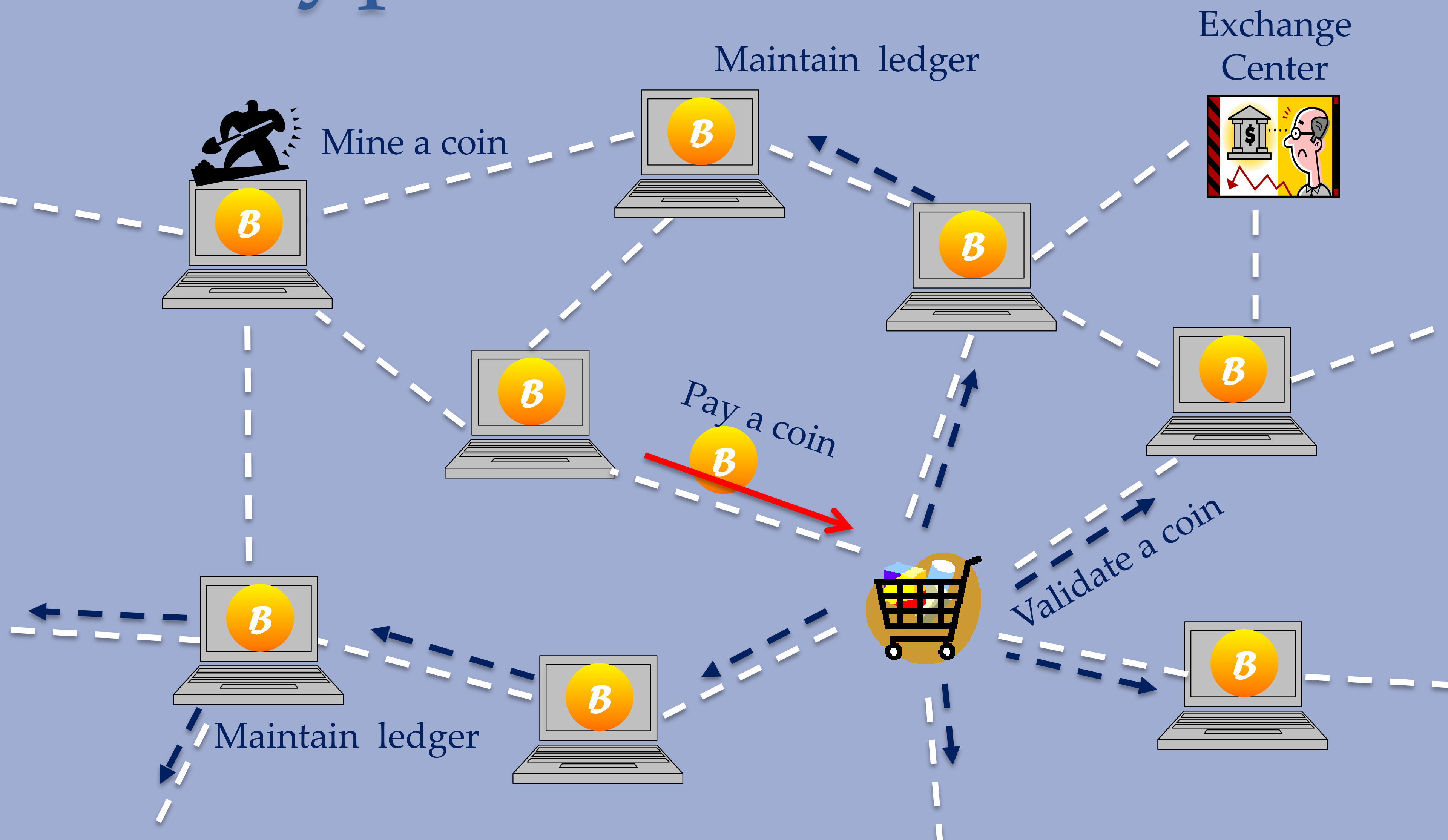
Type-I: Bank-Controlled E-Cash



Type-I: Bank-Controlled E-Cash

- Trust payment tool, transactions guaranteed by the banks
- High transaction process fee
- MONDEX
  - by National Westminster Bank in the U. K.
  - great success in 1990s, absolute anonymity
  - But open a perfect channel for criminals to untraceably transfer their illegal funds

Type-II: P2P BitCoin



Type II: P2P BitCoin

- Low transaction fee
- Dramatic price → Big risk for the holders
- Hacker attacks → Bigger risk
  - 2014/2 the biggest exchange Mt.gox closed due to 850,000 Bitcoins (about $4.8b) stolen by hackers
- Privacy issue
  - one may trace sensitive transactions or de-anonymous social network data through using network topology


www.coindesk.com/price/
CoinDesk
The Voice of Digital Currency
CoinDesk Bitcoin Price Index
$642.40 £382.62 €474.77 ¥4058.81
Last updated: Jun 11, 2014 at 17:39 BST
NEWS News & analysis
INFORMATION What is bitcoin?
PRICE Current Bitcoin Price
DATA Bitcoin Stats
Bitcoin > Bitcoin Price Index
USD CNY
$ 642.40 -1.14%
Today's Open $649.81
Change $-7.41
Today's High $654.74
Market Cap $8.275B
Today's Low $634.18
Total BTC 12,882,075
www.xe.com/currencycharts/?from=XBT&to=USD&view=1Y
USD per 1 XBT
10 Jun 2014 06:45 UTC
XBT/USD close:653.06375
2,000
1,500
1,000
500
0
4 Dec 1230.7
July 90.3
16 Feb 260.4
Jul
Sep
Nov
Jan 14
Mar
May
12h
1D
1W
1M
1Y
2Y
5Y
10Y

Our Solution

ID-Based E-Cash

The system uses issuer's identity

URL "Amazon.com",

SWIFT/BIC "NWBKGB55",

…

As e-cash's public verification key.

ID-Based Cryptosystem

- A Public Key Cryptosystem
- Cheaper transaction handle fee
    - Doesn't need PKI building, maintenance
    - Doesn't need certificate issuing, maintenance, access
- RSA

Anonymity-evocable E-Cash

- Prevent from money laundry, illegal money transfer

- Legally using e-cash → anonymity maintained
- Illegally using e-cash → anonymity revoked

Our E-Cash System

Verify E-Cash Using
Public Identity

Security Bases

- Elliptic Curve Cryptosystem (ECC)
- G is an additive group over a properly chosen elliptic curve. When $|G| = q$ is sufficiently large. It can be used as an ECC.
- **DL problem**: given a group $G = \langle P \rangle$ and a random point Q in G, it is computationally infeasible to find the integer a such that $Q = aP$.
- **CDH problem**: given two random points aP and bP in G, calculating abP is computationally infeasible.
- A Secure One-Way Hash function H: $y = H(x)$.

p.36

Bilinear Pairing

- G_1, G_2, q, P, e:
- Assume that
  - (G_1, +) and (G_2, ·) are two cyclic groups of order q.
  - $e : G_1 X G_1 \rightarrow G_2$.
  - P is a generator of G_1, e(P, P) is the generator of G_2.
- Then
- Bi-linearity: $e(aP, bQ) = e(P, Q)^{ab}$.

Proposed E-Cash: Notations

x	**Bank's private key**
ID_{BV}	bank's pubic data indicates the bank's identity together with a valid period.
Q_B	$Q_B = H_1(ID_{BV})$ is bank's public data
P_B	$P_B = xQ_B$ is bank's public data
$S_T(.)$	Trustee 's signature on some message
w	A user-chosen license key
Q_L	$Q_L = wP_B$ is a license for e-cash
SK	a session key shared between the user and merchant in a payment transaction

Proposed E-Cash System Initialization

- Bank B registers its private key x and its identity with a valid date, $ID_{BV} = (ID_B, VD_B)$, to the trustee T.
- B then obtains $Q_B = H_1(ID_{BV})$, $P_B = xQ_B$ and $SIG_T(Q_B, P_B)$, where $SIG_T(Q_B, P_B)$ is the trustee's signature.
- B publish ID_{BV}, P_B and $SIG_T(Q_B, P_B)$.

Proposed E-Cash License Issuing Protocol

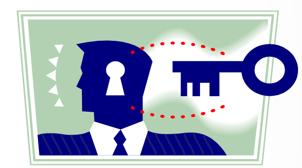
1\. Selects ***license key*** $w \in Z_q^*$.

w, *BankName* $\longrightarrow$

$\longleftarrow$

$Q_L, S_T(Q_L), ID_{BV}, P_B$

Trustee



2.Fetchs the bank's pubic data ID_{BV}, Q_B, P_B.

3.Computes

$\boldsymbol{Q_L} = \boldsymbol{wP_B}$,

signs on $\boldsymbol{Q_L} = S_T(Q_L)$.

License=$\{Q_L, S_T(Q_L), ID_{BV}, P_B\}$

Proposed E-Cash Withdraw E-Cash Protocol

1. Selects coin c,

Blind factor R', a, $b \in Z_q^*$
such that $ab = w \bmod q$
$R = wR'$,
$M = b(H(c, R, ID_{BV}) + R)$.

coin, license-key
$\downarrow$
blind factor $\rightarrow$ coin, license
blind message M

blind message M $\dashrightarrow$

$\dashleftarrow$ blind signature S'

2. Computes $S' = xM$

3. Sigature $S = aS'$

E-Cash= $\boxed{c, R, S, Q_L}$

coin, license
blindly

Proposed E-Cash Payment Protocol

paying \$ *m*

merchant

1\. c, R, S, Q_L $S_T(Q_L), ID_{BV}, P_B$

2\. Verifies license,
Generate a challenge
$K = \textcolor{red}{k} P_B$.

K

3.Computes $SK = wK$,
$S_m = wH(SK, t, m)$

t, m, S_m

4\. $Q_B = H_1(ID_{BV})$, $SK = kQ_L$ $(=kwP_B)$
Verifies the ownership of the **license** $e(P_B, S_m) =?$ $e(H(SK,t,m), Q_L)$.
Verifies e-cash $(Q_B, S) =?$ $e(Q_L, H(c,R,ID_{BV}))\, e(P_B,R)$.
5\. Write payment record.

$\{c, S, R\ Q_L, S_T(Q_L), ID_{BV}, P_B, \textcolor{red}{k}, t, m, S_m\}$

p36

Proposed E-Cash Verification Formula

The proposed e-cash verification equation is

$$\hat{e}(H_1(ID_{BV}), S) =? \ \hat{e}(Q_L, H(c||R||ID_{BV}))\,\hat{e}(P_B, R).$$

We show the proof below.

$$\begin{aligned}
&\hat{e}(H_1(ID_{BV}), S) \\
=\ & \hat{e}(Q_B, S) \\
=\ & \hat{e}(Q_B, aS') \\
=\ & \hat{e}(Q_B, axM) \\
=\ & \hat{e}(Q_B, axb(H(c||R||ID_{BV}) + R')) \\
=\ & \hat{e}(Q_B, wxH(c||R||ID_{BV}))\,\hat{e}(Q_B, wxR') \\
=\ & \hat{e}(wxQ_B, H(c||R||ID_{BV}))\,\hat{e}(xQ_B, R). \\
=\ & \hat{e}(wP_B, H(c||R||ID_{BV}))\,\hat{e}(P_B, R). \\
=\ & \hat{e}(Q_L, H(c||R||ID_{BV}))\,\hat{e}(P_B, R).
\end{aligned}$$

computing $SK = k \cdot Q_L = kwP_B$ and checking if $\hat{e}(P_B, S_m) = \hat{e}(H(SK||t||m), Q_L)$ holds.

Proposed E-Cash Deposit Protocol

Merchant

payment record:

$\{c, S, R\ Q_L, S_T(Q_L), ID_{BV}, P_B, \mathbf{k}, t, m, S_m\}$

(secure channel)

Bank

a. Verifies license Q_L, $S_T(Q_L)$.

b. Verifies the ownership of the **payment record** by checking ***k***. /* Bank computes $SK = k \cdot Q_L = kwP_B$ and checks (P_B, S_m) =? $(H(SK||t||m), Q_L)$ */

c. Verifies E-Cash and the ownership of the E-Cash.

If all are valid, it checks

d. **Payment record** duplicate?

e. E-Cash is over-spent?

p37

Accept / Reject

(secure channel)

Privacy and Security Analysis

Q 1. (Anonymity Issue) Can a bank link to a specific user by e-cash $\{c, S, R, Q_L\}$ between or after a withdrawal process?

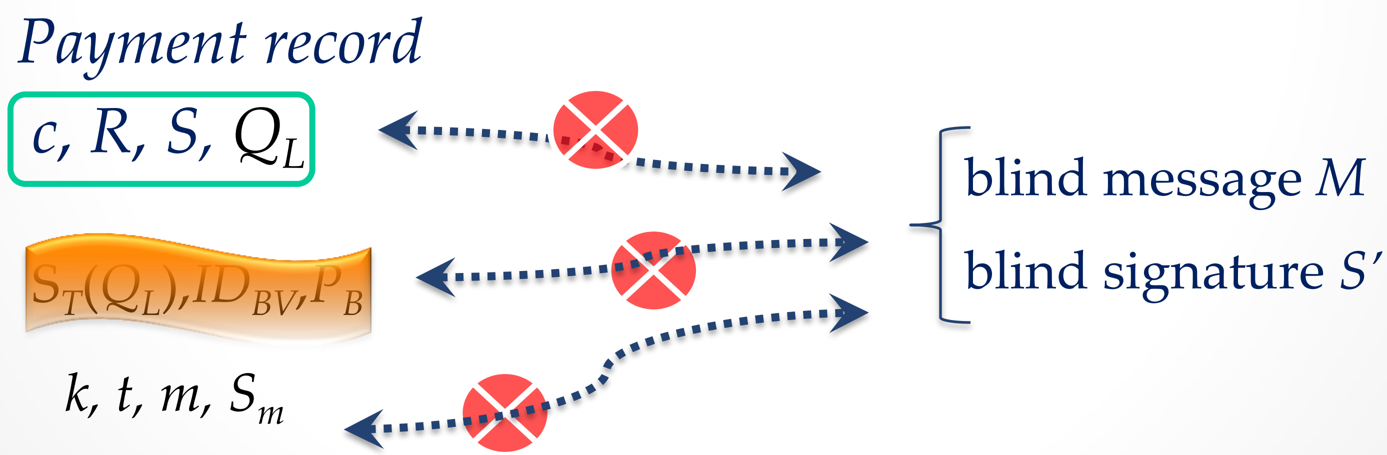
blind message M

blind signature S'

E-Cash= $\boxed{c, R, S, Q_L}$

Privacy and Security Analysis

Q 2. (Anonymity Issue) Is bank B able to link the returned e-cash, i.e. a payment record, $\{c, S, R\ Q_L, S_T(Q_L), ID_{BV}, P_B, k, t, m, S_m\}$*, to any previous withdrawal transcript,* $\{M, S'\}$*, and thus link it to the identity of the user?*



Privacy and Security Analysis

Q3. Can user U forge e-cash only by using his/her Q_L in the registered license without the bank's involvement?

➡ Without knowing the bank's private key *x*, *U* will meet a DL problem in computing *the e-cash component* $S = \boldsymbol{x} \cdot w \cdot H(c, R, ID_{BV})$.

Privacy and Security Analysis

Q4. (Unforgeability Issue) Can bank B make e-cash by itself ?

➡ Without knowing license key *w*,
B will meet a DL problem in computing
$S = \boldsymbol{w} \cdot x \cdot H(c, R, ID_{BV})$.

Privacy and Security Analysis

Q5. (Unforgeability Issue) Can an adversary forge valid e-cash ?

From Q 3 and 4, we can easily see that even a bank which only knows x or a user who only knows w, cannot successfully forge e-cash.

Privacy and Security Analysis

Q6. Can an adversary reuse an eavesdropped payment transcript to pay the e-cash?

He must compute $S_m = wH(SK, t, m)$
for a new challenge K.
BUT, without knowing the license key w,
even he knows the SK,
he will meet a CDH problem (p.22) .
That is, he is unable to compute S_m to satisfy the
license ownership verification(p.28).
i.e.,
Knowing vP_B ($=H(SK,t,m)$) for some v, and
$Q_L = wP_B$, but without knowing the license key w
Compute $S_m = wvP_B = wH(SK, t, m)$ is a CDH problem

Privacy and Security Analysis

Q7. Can an adversary deposit the eavesdropped e-cash from a payment transcript to his/her bank account?

Only the merchant who knows the discrete logarithm of challenge K in the transcript can let the bank successively verify the ownership of the **payment record** in the deposit protocol (p.30) .

Therefore, he will meet a DL problem in computing $SK = k \cdot Q_L$

To pass the verification

$$e(P_B, Sm) \ =?\ e(H(SK||t||m), Q_L) .$$

Conclusions

- Q1 to Q7 shows the security and privacy preservation of the proposed E-Cash.
- The proposed E-Cash is an ID-based system

  free from PKI building, maintenance, and access,

  and thus lower the transaction cost.
- The proposed E-Cash has anonymous property and thus links to nobody.

  Therefore is no card No., no identity…

  can be stolen.

Thank You

Q & A

科技部補助計畫衍生研發成果推廣資料表

日期:2014/07/21

科技部補助計畫	計畫名稱: 防帳卡號被盜之低成本免憑證電子錢網路付款可行性方案研究
	計畫主持人: 周志賢
	計畫編號: 102-2221-E-343-004-　　　　學門領域: 資訊安全

無研發成果推廣資料

102年度專題研究計畫研究成果彙整表

計畫主持人：周志賢			計畫編號：102-2221-E-343-004-				
計畫名稱：防帳卡號被盜之低成本免憑證電子錢網路付款可行性方案研究							
成果項目			量化			單位	備註（質化說明：如數個計畫共同成果、成果列為該期刊之封面故事...等）
			實際已達成數（被接受或已發表）	預期總達成數(含實際已達成數)	本計畫實際貢獻百分比		
國內	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	0	0	100%		
		專書	0	0	100%		
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（本國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	50%		
		專任助理	0	0	100%		
國外	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	1	1	100%		獲邀擔任 Plenary Speakers
		專書	0	0	100%	章/本	
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（外國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		

其他成果 (無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。)	獲得優秀論文獎狀一張。

	成果項目	量化	名稱或內容性質簡述
科教處計畫加填項目	測驗工具(含質性與量性)	0	
	課程/模組	0	
	電腦及網路系統或工具	0	
	教材	0	
	舉辦之活動/競賽	0	
	研討會/工作坊	0	
	電子報、網站	0	
	計畫成果推廣之參與（閱聽）人數	0	

科技部補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

■達成目標

□未達成目標（請說明，以 100 字為限）

□實驗失敗

□因故實驗中斷

□其他原因

說明：

2. 研究成果在學術期刊發表或申請專利等情形：

論文：■已發表 □未發表之文稿 □撰寫中 □無

專利：□已獲得 □申請中 ■無

技轉：□已技轉 □洽談中 ■無

其他：（以 100 字為限）

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）（以 500 字為限）

目前，安全的行動支付工具，已是零售電子商務業者和平台必爭的利器；而提供相關支援的軟硬體備廠商等也競相提出各種解決方案。一個電子支付的場景如下，使用者用手機掃瞄產品的商品 QRCode，之後在透過第三方支付工具扣款，或者利用智手機下載 QRCode 的 App 服務，並完成身份認證與鍵入信用卡號後，能隨時用手機行動消費。又例如，手機 Android 系統推出 TSM（Trusted Service Manager）平台，讓使用者將信用卡號是鍵入手機內 SWP-SIM 卡，作為行動支付運作的核心關鍵；蘋果 iOS 系統，在 iPhone 6 推出 App Pay 服務，準備和 Android TSM 相競抗衡。

在這樣的背景下，本研究提出的方法和實作正可以提供業者一個安全演算法的選擇。我們的方法除了和 RSA 公開金鑰密碼系統一樣安全外（基於計算複雜理論），它的金鑰長度只需要 RSA 金鑰長度的五分之一以下，而且我們的方法是 ID-Based 公開金鑰密碼系統，也就是說允許用身分識別作為公開金鑰，例如 Gmail 帳號，銀行的 SWIFT 代碼，業者的官方網址等等。所以，我們的方法承接了 ID-Based 公開金鑰密碼系統的好處，它不需要建置昂貴的“公開金鑰基礎建設”（Public Key Infrastrucute，PKI）來驗證不可讀的“公開金鑰”。

RSA 公開金鑰：

Public exponent:

0x10001

Modulus:

1350664108659952233496032162788059699388814756056670275244851438515265106048595338339402871505719094417982072821644715513736804197039641917430464965892742562393410208643832021103729587257623585096431105640735015081875106765946292055636855294752135008252879463773285339061097505443349998111500569772368909275623

ID-Based 公開金鑰：

Public key:

Alice@xxx.com

更進一步說，本研究允許全球知名的企業例如 Amazon、Westminster Bank、Citybank，eBay、拉里巴巴等，或地區知名的企業例如 PChome、Yahoo、義美、台灣銀行等，用其組織的 URL（例如網址）發行自己的匿名電子錢。也就是讓一些非銀行組織也可擔任第三支付的角色，以活化電子商務金流的快速流動並降低處理成本。本研究另一個結論是，比起交易成本最低的比特幣（Bitcoin），我們的方法提供更安全的電子支付，其處理成本比起現行的信用卡或透過銀行組織的各式帳卡低廉許多。