Analyzing an Electronic Cash Protocol Using Applied Pi Calculus[⋆]

Zhengqin Luo[1], Xiaojuan Cai[1], Jun Pang[2], and Yuxin Deng[1]

[1] BASICS, Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, 200240, China
{martyluo, cxj, yuxindeng}@sjtu.edu.cn
[2] Carl von Ossietzky Universität Oldenburg
Department für Informatik, 26111 Oldenburg, Germany
jun.pang@informatik.uni-oldenburg.de

Abstract. *Untraceability* and *unreuseability* are essential security properties for electronic cash protocols. Many protocols have been proposed to meet these two properties. However, most of them have not been formally proved to be untraceable and unreuseable. In this paper we propose to use the applied pi calculus as a framework for describing and analyzing electronic cash protocols, and we analyze Ferguson's electronic cash protocol as a case study. We believe that this approach is suitable for many different electronic cash protocols.

1 Introduction

Security protocols for on-line payments play an important role in today's electronic commerce. These protocols can be applied in a myriad of circumstances, from real time money transfer to selling soccer match tickets on the Internet. However, when transactions are monitored, some private information of customers, which should be kept secret, is recorded too. In order to protect users' privacy, researchers have proposed a set of untraceable electronic cash protocols [14, 19, 11, 12, 29]. Similar to physical cash, electronic cash is also portable, recognizable, transferable, and untraceable (anonymous).

As exemplified in Figure 1, an electronic cash protocol usually consists of three sub-protocols — the *withdraw* protocol, the *payment* protocol, and the *deposit* protocol. It also has three types of principals — a *bank*, a *payer*, and a *shop*. A typical flow of the protocol is given as follows:

1. By executing a withdraw stage protocol with the bank over an authenticated channel, the payer can obtain electronic coins issued by the bank;
2. The payer spends these coins in the shop, by performing the payment stage protocol, in which the shop can be convinced that these coins are not faked;

[⋆] The work is supported by The National Grand Fundamental Research 973 Program of China (2003CB317005), and The National Nature Science Foundation of China (60473006) and (60573002).

3. The shop can initiate the deposit stage protocol with the bank, to verify and deposit these coins into its account.

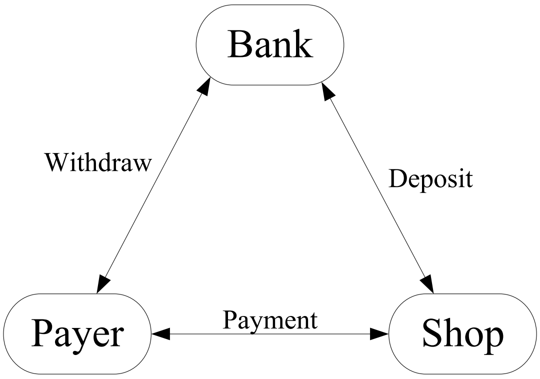

Fig. 1. Typical electronic cash protocol

In an on-line electronic cash protocol, the bank is required to stay in a standby condition to verify the coins for the shop in the payment stage protocol, while in an off-line one, its payment stage protocol does not require interactions between the shop and the bank. That is to say, the shop verifies the validity of the electronic coins without the bank's assistance. In this paper, we only consider off-line electronic cash protocols because they reduce the complexity of the bank systems. However this advantage also brings the danger of the payer's using the same coin more than once. So two main security properties are required for any off-line electronic cash protocol:

- *Anonymity (Untraceability).* The bank should not be able to determine if a certain payment is made by a particular payer, even with the shop's cooperation.
- *Detection-of-double-spending (Unreuseability).* If a dishonest payer spends a coin more than once, the bank should be able to detect the payer's identity with an overwhelming probability.

To our knowledge, most electronic cash protocols have not been proved to be untraceable and unreuseable. Recently, the need for applying formal methods to security protocols has been widely recognized and there have been several attempts to develop a formal framework for specifying and reasoning about security properties. For example, CSP [22] and the spi calculus [5] have been used to analyze security protocols [28, 6]. The applied pi calculus [4], which is a variant of the pi calculus [26, 27] extended with value passing, function symbol, and equational theory over terms and functions, has been successfully applied to verify some security protocols [2]. The protocol verifier ProVerif [8] provides a set of proof techniques which can be used directly in proving the equivalence between processes of the applied pi calculus.

The main contributions of this paper are summarized as follows.

- We model Ferguson's electronic cash protocol in the applied pi calculus by defining an appropriate signature and equational theory.
- We demonstrate that Ferguson's electronic cash protocol fulfills untraceability and unreuseability. The proofs are partly done by ProVerif.

Related work. Anonymity (untraceability) was first proposed by Chaum [13] to solve the Dinning Cryptographer Problem. After that, a great deal of research has been carried out on this topic and various formal definitions and frameworks for analyzing anonymity have been developed in the literature. For example, Schneider and Sidiropoulos analyzed anonymity with CSP [28]. They used substitution and observable equivalence to define anonymity in CSP. In their framework, the automatic tool FDR [24] was used to check the equivalence of two processes. In [23] Kremer and Ryan analyzed the FOO92 voting protocol with the applied pi calculus and proved that it satisfies anonymity. Chothia [15] used bisimulation in the pi calculus to test the anonymity of an anonymous file-sharing system. Chothia *et al.* [16] proposed a general framework based on the process algebraic verification tool ^{1}CRL [10] for checking anonymity and applied it to several protocols, including the Dinning Cryptographer Problem and the FOO92 voting protocol. Our framework is similar to [23], but our proofs are partly done by ProVerif when in [23] all the proofs are done manually.

Other works, such as [7, 17, 18], considered probabilistic anonymity. Bhargava and Palamidessi [7] formulated their notions of probabilistic anonymity in terms of observables for processes in the probabilistic pi calculus [25]. Deng, Palamidessi and Pang [17] extended the work of [7] and defined the notion of weak probabilistic anonymity and used a probabilistic model checker [21] to automatically analyze the Dining Cryptographers Problem. In [18] Deng, Pang and Wu used the notion of relative entropy from information theory to measure the degree of anonymity a protocol can guarantee, and they proposed a probabilistic process calculus to describe protocols. They considered the scenario with nondeterministic and probabilistic users, which is more realistic. However, the expressiveness of the process calculi they used are less powerful than that of the applied pi calculus used here, which can express all the computations by functions. All of these works on probabilistic anonymity have not been applied to electronic cash protocols.

Organization of the paper. In next section, we briefly introduce the applied pi calculus. In Section 3, we model a simplified version of Ferguson's electronic cash protocol. Two crucial properties of the protocol, untraceability and unreuseability, are analyzed in Sections 4 and 5, respectively. Finally, we conclude the paper and discuss some future work in Section 6.

2 The applied pi calculus

In this section, we give a brief overview of this calculus. The reader is referred to [4] for more details.

2.1 Syntax

To describe a process in the applied pi calculus, one should first define a signature § which consists of some function symbols. Given a signature §, an infinite set of names, and an infinite set of variables, the set of *terms* are defined below:

$L; M; N; T; U; V$ ::=		terms
	$a; b; c; : : : ; k; : : : ; m; n$	name
	$x; y; z$	variable
	$f(M_1; : : : ; M_l)$	function application

Equational theories play an important role in security protocol analysis. An equational theory over a signature usually consists of a set of equations asserting the equality of cryptographic primitives. For example, in order to model the symmetry cryptography, one can use the equation

$$\mathsf{dec}(\mathsf{enc}(x; y); y) = x:$$

Here x represents a plaintext and y is a key. The binary function symbols enc and dec denote encryption and decryption operation, respectively.

We usually use E to denote an equational theory. The notation § $\vdash M =_E N$ means the equation $M = N$ is in the theory E associated with §.

The definition of plain processes is similar to the one in the pi calculus, except that messages can contain terms rather than names.

$P; Q; R ::= \mathbf{0}$	null process
$P \mid Q$	parallel composition
$!P$	replication
$^o n{:}P$	name restriction ("new")
if $M = N$ *then* P *else* Q	conditional
$u(x){:}P$	message input
$\bar{u}\langle N\rangle{:}P$	message output

Extended processes introduce active substitutions and variable restrictions.

$A; B; C ::= P$	plain process
$A \mid B$	parallel composition
$^o n{:}A$	name restriction
$^o x{:}A$	variable restriction
$\{M{=}x\}$	active substitution

Here $\{M{=}x\}$ is an active substitution which replaces the variable x with the term M, just like "*let* $x = M$ *in* $: : :$". The active substitution $\{M{=}x\}$ typically appears when the term M has been sent to the environment. The variable restriction $º x$ restricts the scope of active substitutions. We write $fv(A)$, $bv(A)$, $fn(A)$, and $bn(A)$ for free and bound variables and free and bound names of A.

A *closed* extended process A can be rewritten into a form which consists of a substitution and a plain process with some restricted names:

$$A \equiv º\, \tilde{n}\text{:}\{\widetilde{M}\text{=}\tilde{x}\} \mid P$$

where $fv(P) = \emptyset; fv(\widetilde{M}) = \emptyset$, and $\{\tilde{n}\} \subseteq fn(\widetilde{M})$.

Every extended process can be mapped to a *frame* $Á(A)$ which contains only restriction and parallel composition of active substitutions, by replacing every plain process in A with $\mathbf{0}$. The frame $Á(A)$ can be viewed as static knowledge exposed by A to its environment, but not for A's dynamic behavior. We write $dom(\,')$ for the domain of $\,'$, a set of variables which appear in active substitutions in $\,'$ but not under a variable restriction.

We write $Á \vdash M$ to mean M can be deduced from $Á$. This relation is called *deduction* which is axiomatized by the following rules.

Subst $\dfrac{}{º\, \tilde{n}\text{:}¾ \vdash M}$ $if\ \exists x \in dom(¾)\ s\text{:}t\text{:}\ x¾ = M$

Nonce $\dfrac{}{º\, \tilde{n}\text{:}¾ \vdash s}$ $if\ s \notin \tilde{n}$

Funct $\dfrac{Á \vdash M_1 \ \cdots \ Á \vdash M_k}{Á \vdash f(M_1; \cdots ; M_k)}$ $f \in §$

Equiv $\dfrac{Á \vdash M \quad M =_E N}{Á \vdash N}$

An *evaluation context*, denote by $C[\_]$, is a context whose hole is not under a replication, a conditional, an input, or an output. An evaluation context $C[\_]$ closes A when $C[A]$ is closed, and $C[\_]$ is called a *closing evaluation context.*

2.2 Semantics

Structural equivalence, written $A \equiv B$, is the smallest equivalence relation on extended processes that is closed under ®-conversion on both names and variables, and under evaluation contexts.

Internal reduction $\to$ is the smallest relation on extended processes closed by structural equivalence and application of evaluation contexts:

Comm $\bar{a}\langle x\rangle\text{:}P \mid a(x)\text{:}Q \to P \mid Q$

Then *if* $M = M$ *then* P *else* $Q \to P$

Else *if* $M = N$ *then* P *else* $Q \to Q\ ;$ when $§ \nvdash M =_E N$

As usual, labeled operational semantics extends reduction semantics and enables us to reason about the interaction between processes and the environment.

We give a labeled operational semantics which defines transitions of the form $A \xrightarrow{\alpha} A'$ with ® being a label of input or output action. The following rules are adopted in the labeled operational semantics.

In	$a(x){:}P \xrightarrow{a(M)} P\{M{=}x\}$
Out-Atom	$\bar{a}\langle u\rangle{:}P \xrightarrow{\bar{a}\langle u\rangle} P$
Open-Atom	$\dfrac{A \xrightarrow{\bar{a}\langle u\rangle} A' \quad u \neq a}{\text{º}u{:}A \xrightarrow{\nu u.\bar{a}\langle u\rangle} A'}$
Scope	$\dfrac{A \xrightarrow{\alpha} A' \quad u \text{ does not occur in ®}}{\text{º}u{:}A \xrightarrow{\alpha} \text{º}u{:}A'}$
Par	$\dfrac{A \xrightarrow{\alpha} A' \;\; bv(\text{®}) \cap fv(B) = bn(\text{®}) \cap fn(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$
Struct	$\dfrac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad B' \equiv A'}{A \xrightarrow{\alpha} A'}$

2.3 Equivalence relations

We write $A \Downarrow a$ when A can send a message on channel a, i.e., $A \rightarrow^* C[\bar{a}\langle M\rangle{:}P]$ for some evaluation context $C[\_]$ that does not bind a.

Definition 1 (Observational equivalence). *Observational equivalence ($\approx$) is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A \;\mathcal{R}\; B$ implies:*

1. *if $A \Downarrow a$, then $B \Downarrow a$;*
2. *if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A' \;\mathcal{R}\; B'$ for some B';*
3. *$C[A] \;\mathcal{R}\; C[B]$ for all closing evaluation contexts $C[\_]$.*

Definition 1 says that two processes which cannot be distinguished in any context are observationally equivalent. The context usually denotes an attacker.

Definition 2. *We say that two terms M and N are equal in the frame Á, and write $(M = N)$Á, if and only if Á $\equiv$ º$\tilde{n}$:¾, M¾ $=$ N¾, and $\{\tilde{n}\} \cap (fn(M) \cup fn(N)) = \emptyset$ for some names $\tilde{n}$ and substitution ¾.*

Definition 3 (Static equivalence). *Two closed frames ' and Ã are statically equivalent, written ' $\approx_s$ Ã, for ' $\equiv$ º$\tilde{n}_1$¾$_1$ and Ã $\equiv$ º$\tilde{n}_2$¾$_2$, when dom(') $=$ dom(Ã) and when, for all terms M and N, we have $(M = N)$' if and only if $(M = N)$Ã.*
Two extended processes are static equivalent if and only if Á$(A) \approx_s$ Á(B).

Static equivalence only defines a relation on frames, which are static knowledge exposed to the environment by some processes. More details about static equivalence can be found in [3].

Definition 4 (Labeled bisimilarity). *Labeled bisimilarity ($\approx_l$) is the largest symmetric relation $\mathcal{R}$ on closed extended processes such that $A \ \mathcal{R}\ B$ implies:*

1. $A \approx_s B$;
2. *if* $A \to^* A'$, *then* $B \to^* B'$ *and* $A' \ \mathcal{R}\ B'$ *for some* B'
3. *if* $A \xrightarrow{\alpha} A'$ *and* $fv(\circledR) \subseteq dom(A)$ *and* $bn(\circledR) \cap fn(B) = \emptyset$, *then* $B \to^* \xrightarrow{\alpha} \to^* B'$

The following theorem was proved in [4].

Theorem 1. $\approx_l = \approx$ *and* $\approx \subseteq \approx_s$.

Theorem 1 shows that observational equivalence coincides with labeled bisimilarity and observational equivalence is finer than static equivalence. Labeled bisimilarity does not consider all the contexts, which leads to easy proofs in many occasions.

3 Modeling an electronic cash protocol

In this section, we first introduce a simplified version of Ferguson's electronic cash protocol [19], then we define an appropriate equational theory with reasonable abstraction for the blind signatures scheme in the protocol. Finally we model the protocol in the applied pi calculus from the viewpoint of three types of principals.

3.1 Ferguson's electronic cash protocol

As mentioned in the introduction, the protocol consists of three types of participants, the *bank*, the *payer*, and the *shop*. The protocol employs a randomized blind signatures scheme based on RSA-signature scheme to ensure that the payer can obtain the bank's signature on the coin, while the bank has no knowledge of the coin. A polynomial secret sharing scheme is used for double-spending detection. The whole protocol can be described by three sub-protocols:

Withdraw Stage

1. The payer chooses three numbers c, k, and g, where c represents a coin, k is a random number, and g is a blinding factor;
2. The payer blinds the coin c with k and g using blinding function $\mathsf{blind}(c; k; g)$, and then sends this blinded coin together with its identity U to the bank over an authenticated channel c_1;
3. The bank signs on the blinded coin using the randomized blind signatures scheme, generates two blinded signatures s_1 and s_2, and sends them back to the payer on c_2;
4. The payer unblinds these two signatures received from the bank using unblinding function $\mathsf{unblind}$ and blinding factor g, and obtains $\mathsf{sign}(C^k A; prvkey)$ and $\mathsf{sign}(C^U B; prvkey)$, which are two RSA signatures with the bank's private key $prvkey$ on $C^k A$ and $C^U B$ separately. Here $C = f_c(c); A = f_a(c); B = f_b(c)$, and $f_c; f_a; f_b$ are public suitable one-way functions.

Payment Stage

1. The payer sends the coin c to the shop over a secret channel c_{pay};
2. The shop sends the payer a randomly chosen (non-zero) challenge x on c_3;
3. The payer computes a response $r = kx + U$ (the share of the payer's identity for double-spending detection) and a signature on $C^r A^x B$ which can be easily deduced from the two RSA-signatures obtained from the bank, and sends them on channel c_4;
4. The shop can verify the signature on $C^r A^x B$ with the bank's public key to ensure the validity of the payer's coin, and then completes the trade with the payer.

Deposit Stage

1. The shop sends the coin c, the challenge x and the response (both r and the signature) to the bank on channel c_5 to deposit the coin;
2. The bank can check whether the coin is valid by verifying the correctness of the signature, and then stores $(c; x; r)$ in the Used-coin Database and informs the shop on channel c_{payOK} if the coin is valid;
3. If the payer spends the coin c twice dishonestly, then there must be two entries $(c; x_1; r_1)$ and $(c; x_2; r_2)$ associated with c in the database. Since $(x_1; r_1)$ and $(x_2; r_2)$ are two different points on the line $r = kx + U$, the bank is able to determine the payer's identity U immediately.

3.2 Equational theory

To analyze Ferguson's electronic cash protocol, we define signature § for cryptography primitives in the following way:

$\mathsf{blind}(c; k; g)$	(* *blind a coin with k, and g* *)
$\mathsf{unblind}(c; g)$	(* *undo blinding* *)
$\mathsf{pk}(sk)$	(* *get public key from private key* *)
$\mathsf{blindsignA}(c; U; sk)$	(* *blind signature algorithm A* *)
$\mathsf{blindsignB}(c; U; sk)$	(* *blind signature algorithm B* *)
$\mathsf{sign}(m; sk)$	(* *RSA signature scheme**)
$\mathsf{checksign}(m; s; pk)$	(* *verify RSA signature* *)
$\mathsf{hash1}(c; k; U)$	(* *hash1(c,k,U)=*$C^k A$ *)
$\mathsf{hash2}(c; k; U)$	(* *hash2(c,k,U)=*$C^U B$ *)
$\mathsf{hash3}(c; x; r)$	(* *hash3(c,x,r)=*$C^r A^x B$ *)
$\mathsf{resp\text{-}r}(x; k; U)$	(* *polynomial secret sharing scheme* *)
$\mathsf{resp\text{-}s}(x; s_1; s_2)$	(* *compute the coin's signature from the two given by the bank* *)
$\mathsf{reveal}(x; y)$	(* *determine the identity of the double-spender* *)
$(\_ ; \_); (\_ ; \_ ; \_); : : :$	(* *constructors for combined messages* *)
$\mathsf{F}_1; : : : ; \mathsf{F}_i; : : :$	(* *projections for combined messages* *)

The equational theory on § is given as follows.

- For convenience, we introduce $(\_ ; \_)$, $(\_ ; \_ ; \_)$, ... to denote combinations of messages, and F_i extracts the i-th component of the combined messages. An example of the equation is:

$$\mathsf{F}_i((x_1; : : : ; x_i; : : : ; x_n)) = x_i$$

- For f_a, f_b, and f_c, which are three public suitable one-way functions used in the protocol, we introduce hash1, hash1, and hash3 for some kinds of combinations of f_a, f_b and f_c:

$$\begin{gathered}\mathsf{hash1}(coin; k; U) = C^k A\\ \mathsf{hash2}(coin; k; U) = C^U B\\ \mathsf{hash3}(coin; x; r) = C^r A^x B\end{gathered}$$

  There is no equation on hash1, hash2, and hash3, because these one-way functions are collision-free.
- In order to model the RSA signature scheme, sign and checksign are employed to denote corresponding signature and verification procedure, and the function symbol pk is used for deriving public-key from private-key. The equation is:

$$\mathsf{checksign}(M; \mathsf{sign}(M; sk); \mathsf{pk}(sk)) = true$$

- The randomized blind signature scheme in the protocol runs as follows: The payer first blinds the coin c with random number k and blinding factor g, and then sends it together with the payer's identity to the bank. The bank executes two special blind signature algorithms A and B with its private key and the payer's identity to generate blinded signatures, from which the payer can obtain two signatures over $C^k A$ and $C^U B$ (in RSA signature scheme) after unblinding operation. The equations are describe as follows:

$$\begin{gathered}\mathsf{unblind}(\mathsf{blindsignA}(\mathsf{blind}(coin; k; g); U; sk); g) = \mathsf{sign}(\mathsf{hash1}(coin; k; U); sk)\\ \mathsf{unblind}(\mathsf{blindsignB}(\mathsf{blind}(coin; k; g); U; sk); g) = \mathsf{sign}(\mathsf{hash2}(coin; k; U); sk)\end{gathered}$$

- Note that we use symbolic abstraction of blind function for the protocol. Since there is no equation on blind functions, a blinded coin can be viewed as a fresh, opaque message, apparently unrelated to the coin c, when the blinding factor g is not revealed.

$$^o n{:}\bar{c}\langle n\rangle \approx {}^o k{:}{}^o g{:}\bar{c}\langle \mathsf{blind}(coin; k; g)\rangle$$

- Finally, the polynomial secret sharing scheme in the protocol is modeled with three function symbols. The first one, $\mathsf{resp\text{-}s}(x; s_1; s_2)$, denotes a response over challenge x and two signature s_1 and s_2 which are obtained from the bank by the payer; the second one, $\mathsf{resp\text{-}r}(x; k; U)$, denotes a response over challenge x on the line $r = kx + U$; the last one, reveal, is a special function symbol by which the bank can detect the double-spender's identity. The equations are defined below:

$$\begin{gathered}\mathsf{resp\text{-}s}(x; \mathsf{sign}(\mathsf{hash1}(coin; k; U); sk); \mathsf{sign}(\mathsf{hash2}(coin; k; U); sk))\\ = \mathsf{sign}(\mathsf{hash3}(c; x; \mathsf{resp\text{-}r}(x; k; U)); sk)\end{gathered}$$

$$\mathsf{reveal}(\mathsf{resp\text{-}r}(x_1; k; U); \mathsf{resp\text{-}r}(x_2; k; U)) = U; where\ x_1 \neq x_2$$

3.3 The payer process

The payer process models the role of a payer, as the one in the protocol which is mentioned in Section 3.1. First, the payer generates a fresh random number k, a fresh blind factor g, and a coin c. Next, the payer blinds the coin c with k and g, and then sends this blinded coin together with its identity U to the bank, expecting two corresponding signatures. Finally, the payer sends the coin c to the shop and accomplishes a challenge-response procedure with the shop.

It should be noticed that $c_1;:::;c_n$ are public channels, used only for synchronizing different protocol steps. For example, the payer process sends its first message on channel c_1, and then the bank process will receive this message on the same channel. Since these channels are public, the adversary will know what has been sent on these channels.

We write $let\ x = M\ \ in\ P$ instead of $P\{M{=}x\}$ for ease of understanding.

$$\begin{array}{ll}
P_{payer} ::= & {}^{o}k{:}{}^{o}g{:}{}^{o}c{:} \\
 & \overline{c_1}\langle(\mathsf{blind}(c;k;g);U)\rangle{:} \\
 & c_2(x_1){:} \\
 & let\ signature_1 = \mathsf{unblind}(\mathsf{F}_1(x_1);g)\ in \\
 & let\ signature_2 = \mathsf{unblind}(\mathsf{F}_2(x_1);g)\ in \\
 & if\ \mathsf{checksign}(\mathsf{hash1}(c;k;U);signature_1;Pub_{bank}) = true\ then \\
 & if\ \mathsf{checksign}(\mathsf{hash2}(c;k;U);signature_2;Pub_{bank}) = true\ then \\
 & \overline{c_{pay}}\langle c\rangle{:} \\
 & c_3(x_2){:} \\
 & \overline{c_4}\langle(\mathsf{resp\text{-}r}(x_2;k;U);\mathsf{resp\text{-}s}(x_2;signature_1;signature_2))\rangle
\end{array}$$

3.4 The bank process

The behavior of the bank is modeled by the process below. After receiving a blinded coin from the payer, the bank sends back two blind signatures, and debits one dollar from the payer's account at the same time. When the shop requests to verify the validity of a coin, the bank first verifies the correctness of signature, and then deposits one dollar to the shop's account.

$$\begin{array}{ll}
P'_{Bank} ::= & (c_1(x_1){:} \\
 & let\ blindcoin = \mathsf{F}_1(x_1)\ in \\
 & let\ U = \mathsf{F}_2(x_1)\ in \\
 & \overline{c_2}\langle(\mathsf{blindsignA}(blindcoin;U;Prv_{bank});\mathsf{blindsignB}(blindcoin;U;Prv_{bank}))\rangle) \mid \\
 & (c_5(x_2){:} \\
 & let\ coin = \mathsf{F}_1(x_2)\ in \\
 & let\ challenge = \mathsf{F}_2(x_2)\ in \\
 & let\ response = \mathsf{F}_3(x_2)\ in \\
 & let\ signature = \mathsf{F}_4(x_2)\ in \\
 & if\ \mathsf{checksign}(\mathsf{hash3}(coin;challenge;response);signature;Pub_{bank}) = true \\
 & then\ \overline{c_{payOK}}\langle\rangle)
\end{array}$$

The bank's public key is defined as a context, in which the private key remains secret and the public is exported.

$$Key_{bank}[\_] ::= {}^{o}Prv_{bank}\text{:}(\{\mathsf{pk}(Prv_{bank})\text{=}Pub_{bank}\} \mid [\_])$$

The bank process is defined as follows, and the replication operator enables the bank process to deal with multiple requests from payers and shops.

$$P_{Bank} ::= Key_{bank}[!P'_{Bank}]$$

3.5 The shop process

The shop is modeled as the process below. In order to determine whether a paid coin is valid, the shop initiates the challenge-response procedure with the payer. Then the shop sends the coin and the result of the challenge-response procedure to the bank to deposit this coin.

```
$P_{shop} ::= {}^{o}\ x\text{:}$
        $c_{pay}(x_1)\text{:}$
        $let\ coin_{pay} = x_1\ in$
        $\overline{c_3}\langle x\rangle\text{:}$
        $c_4(x_2)\text{:}$
        $let\ response = \mathsf{F}_1(x_2)\ in$
        $let\ signature = \mathsf{F}_2(x_2)\ in$
        $if\ \mathsf{checksign}(\mathsf{hash3}(coin_{pay}; x; response); signature; Pub_{bank}) = true$
        $then\ \overline{c_5}\langle(coin_{pay}; x; response; signature)\rangle\text{:}$
        $c_{payOK}()$
```

3.6 The system process

The whole system is obtained by putting in parallel the three components, the payer process, the bank process, and the shop process. Notice that the public key Pub_{bank} used in P_{payer} and P_{shop} is exported by P_{bank}, while the private key of the bank remains secret.

$$P_{system} ::= P_{bank} \mid P_{payer} \mid P_{shop}$$

4 Analysis of untraceability

A formal definition of untraceability was first proposed in [20]. We follow this formal definition and analyze Ferguson's protocol in the applied pi calculus. Here, we only consider passive attackers who eavesdrop on channels.

Definition 5 (Untraceability). *Let passive attacker $\mathcal{A}$ has access to all bank's views of withdraw, payment, and deposit protocols. Then for any two coins $C_i; C_j$ and two withdraws $W_0; W_1$ such that C_i and C_j are originated from W_0 and W_1, $\mathcal{A}$ cannot distinguish whether C_i comes from W_0 or W_1.*

For ease of understanding, we specialize the above general definition to a simple system with two payers P_1 and P_2. Suppose P_1 withdraws $coin_1$, and P_2 withdraws $coin_2$.

$$P_1 ::= P_{payer}\{coin_1{=}c; payer_1{=}U\}$$
$$P_2 ::= P_{payer}\{coin_2{=}c; payer_2{=}U\}$$

We say that an electronic cash protocol satisfies the requirement of untraceability, when process P_1 with $coin_1$ paralleled by process P_2 with $coin_2$ is observationally equivalent to process P_1 with $coin_2$ paralleled by process P_2 with $coin_1$, i.e., $P_1 \mid P_2 \approx P_1\{coin_2{=}coin_1\} \mid P_2\{coin_1{=}coin_2\}$.

Theorem 2 (Untraceability). *Ferguson's electronic cash protocol satisfies the requirement of untraceability.*

Proof. Proving equivalences of two processes which differ only in the choice of some terms is supported by ProVerif [8, 9]. We benefit from this feature of ProVerif, and the proof is partly done by this tool.

We use ProVerif to prove the following equivalence

$$P_{payer}\{coin_1{=}c\} \approx P_{payer}\{coin_2{=}c\}:$$

The code for proving this equivalence in included in the appendix. Based on the result of ProVerif, the process of withdraw stage is independent from the payment stage. Even if the bank and the shop cooperate, they cannot distinguish whether P_1 withdraws $coin_1$ or $coin_2$, so we have

$$P_1 \approx P_1\{coin_2{=}coin_1\}:$$

Thus, from the structural equivalence, the following equivalence is obvious.

$$P_1 \mid P_2 \approx P_1\{coin_2{=}coin_1\} \mid P_2\{coin_1{=}coin_2\}$$

□

5 Analysis of unreuseability

In this section we analyze the unreuseability property of the protocol modeled in Section 3. In an off-line electronic cash protocol, after receiving a coin, the shop does not deposit the coin immediately. Thus, the strategy adopted by the system is to detect the behavior of double-spending instead of preventing it. The formal definition of unreuseability from [20] is given below.

Definition 6 (Unreuseability). *If a coin is successfully deposited twice, then the identity of at least one misbehaving user can be efficiently computed and proved from the bank's view of the deposit.*

In Ferguson's protocol, the payer's identity is embedded in the payment stage by the polynomial secret sharing scheme. Repeated execution of the challenge-response procedure over a same coin will certainly reveal the identity of the payer. Since ProVerif cannot examine the knowledge of a particular participant, the proof of Theorem 3 is done manually.

Theorem 3 (Unreuseability). *Ferguson's electronic cash protocol can detect the identity of a* double-spender*, i.e. if a dishonest payer spends a coin twice*

$$P \xrightarrow{\nu y_1.\overline{c_{pay}}\langle y_1\rangle.c_3(x_2).\nu y_2.\overline{c_4}\langle y_2\rangle} \xrightarrow{\nu y_3.\overline{c_{pay}}\langle y_3\rangle.c_3(x_3).\nu y_4.\overline{c_4}\langle y_4\rangle} A'; \text{ and } (y_1 = y_3)\acute{A}(A')$$

then the payer's identity must be revealed by the bank,

$$\acute{A}(A') \vdash payer$$

Proof. Without loss of generality, we construct a process representing the misbehaving payer.

$$\begin{aligned}
P_{double-spender} ::= \; & {}^{o}k{:}{}^{o}g{:}{}^{o}c{:} \\
& \overline{c_1}\langle(\mathsf{blind}(c;k;g);U)\rangle{:} \\
& c_2(x_1){:} \\
& let\ signature_1 = \mathsf{unblind}(\mathsf{F}_1(x_1);g)\ in{:} \\
& let\ signature_2 = \mathsf{unblind}(\mathsf{F}_2(x_1);g)\ in{:} \\
& if\ \mathsf{checksign}(\mathsf{hash1}(c;k;U);signature_1;Pub_{bank}) = true\ then \\
& if\ \mathsf{checksign}(\mathsf{hash2}(c;k;U);signature_2;Pub_{bank}) = true\ then \\
& \overline{c_{pay}}\langle c\rangle{:} \\
& c_3(x_2){:} \\
& \overline{c_4}\langle(\mathsf{resp\text{-}r}(x_2;k;U);\mathsf{resp\text{-}s}(x_2;signature_1;signature_2))\rangle{:} \\
& \overline{c_{pay}}\langle c\rangle{:} \\
& c_3(x_3){:} \\
& \overline{c_4}\langle(\mathsf{resp\text{-}r}(x_3;k;U);\mathsf{resp\text{-}s}(x_3;signature_1;signature_2))\rangle
\end{aligned}$$

$$P ::= P_{double-spender}\{payer{=}U\}$$

After withdrawing the coin from the bank, and spending the coin twice,

$$P \xrightarrow{\nu y.\overline{c_1}\langle y\rangle.c_2(x_1)} \xrightarrow{\nu y_1.\overline{c_{pay}}\langle y_1\rangle.c_3(x_2).\nu y_2.\overline{c_4}\langle y_2\rangle} \xrightarrow{\nu y_3.\overline{c_{pay}}\langle y_3\rangle.c_3(x_3).\nu y_4.\overline{c_4}\langle y_4\rangle} A'$$

The knowledge exposed by the double-spender to the bank and the shop is

$$\begin{aligned}
\acute{A}(A') = \; & {}^{o}k{:}{}^{o}g{:}{}^{o}c{:} \\
& (\{(\mathsf{blind}(c;k;g);payer){=}y\} \mid \{c{=}y_1\} \mid \{c{=}y_3\} \mid \\
& \{(\mathsf{resp\text{-}r}(x_2;k;payer);\mathsf{resp\text{-}s}(x_2;signature_1;signature_2){=}y_2\} \mid \\
& \{(\mathsf{resp\text{-}r}(x_3;k;payer);\mathsf{resp\text{-}s}(x_3;signature_1;signature_2){=}y_4\})
\end{aligned}$$

Notice that x_2 and x_3 are two challenges initiated by the shop in different sessions, so we can infer $x_2 \neq x_3$. From the equations on resp-r and reveal, we have

$$\frac{\acute{A}(A') \vdash \mathsf{resp\text{-}r}(x_2;k;payer);\acute{A}(A') \vdash \mathsf{resp\text{-}r}(x_3;k;payer)}{\acute{A}(A') \vdash \mathsf{reveal}(\mathsf{resp\text{-}r}(x_2;k;payer);\mathsf{resp\text{-}r}(x_3;k;payer)) = payer}$$

Clearly, this protocol satisfies *unreuseability.* □

6 Conclusion and future work

In this paper we have modeled Ferguson's electronic cash protocol in the applied pi calculus, and we have verified that it satisfies the security properties of untraceability and unreuseability. We believe that the approach used in this paper is suitable for analyzing many other similar electronic cash protocols as well.

As for the future work, it would be interesting to extend the framework of this paper to a probabilistic setting and use it to analyze anonymity of electronic cash protocols. We would also like to analyze other crucial properties of electronic cash protocols, such as divisibility and transferability, and to develop an automatic verification tool to verify security requirements of the protocols. Another future direction is to look at on-line electronic cash protocols, where the bank systems are much more complicated to model.

Appendix - The code for proving Theorem 2 in ProVerif

We use this piece of code to prove the following equivalence

$$P_{payer}\{coin_1{=}c\} \approx P_{payer}\{coin_2{=}c\}.$$

The P_{payer} is described by processP below.

```
(* Ferguson's E-cash protocol *)

(* constant and constructor *)
data true/0.
data U/0.
data con2/2.

(* signature *)
fun blind/3.
(* fun unblind/2. *)
fun pk/1.
fun blindsignA/3.
fun blindsignB/3.
fun sign/2.
fun checksign/3.
fun hash1/3.
fun hash2/3.
fun hash3/3.
fun respr/3.
(* fun resps/3. *)
(* fun reveal/2. *)

(* equational theory *)
equation checksign(m,sign(m,sk),pk(sk))=true.
```

```
reduc unblind(blindsignA(blind(c,k,g),U,sk),g)
      =sign(hash1(c,k,U),sk);
      unblind(blindsignB(blind(c,k,g),U,sk),g)
      =sign(hash2(c,k,U),sk).
reduc resps(x,sign(hash1(c,k,U),sk),sign(hash2(c,k,U),sk))
      =sign(hash3(c,x,respr(x,k,U)),sk).
reduc reveal(respr(x1,k,U),respr(x2,k,U))=U.

(* channel *)
free cpk.
free c1.
free c2.
free c3.
free c4.
private free cpay.

let processP = new k; new g; new coin1; new coin2;
           in (cpk, pubBank);
           let c = choice[coin1,coin2] in
           out (c1, con2(blind(c,k,g),U));
           in (c2, con2(bs1,bs2));
           let s1 = unblind(bs1,g) in
           let s2 = unblind(bs2,g) in
           if checksign(hash1(c,k,U),s1,pubBank) = true then
           if checksign(hash2(c,k,U),s2,pubBank) = true then
           out (cpay, coin);
           in (c3, x);
           out (c4, con2(respr(x,k,U),resps(x,s1,s2))).

process processP
```

References

1. Martín Abadi, Cédric Fournet: Private Authentication. Theoretical Computer Science **322** (2004) 427–476.
2. Martín Abadi, Bruno Blanchet, Cédric Fournet: Just Fast Keying in the Pi Calculus. Proceedings of ESOP 2004, Lecture Notes in Computer Science **2986** (2004) 340–354.
3. Martín Abadi, Véronique Cortier: Deciding Knowledge in Security Protocols under Equational Theories. Theoretical Computer Science **367** (2006) 2–32.
4. Martín Abadi, Cédric Fournet: Mobile Values, New Names, and Secure Communication. Proceedings of POPL 2001, 104-115.
5. Martín Abadi, Andrew D. Gordon: A Calculus for Cryptographic Protocols: The Spi Calculus. Information and Computation **148** (1999) 1–70.
6. Martín Abadi, Andrew D. Gordon: Reasoning about Cryptographic Protocols in the Spi Calculus. Proceedings of CONCUR 1997, Lecture Notes in Computer Science **1243** (1997) 59–73.

7. Mohit Bhargava, Catuscia Palamidessi: Probabilistic Anonymity. Proceedings of CONCUR 2005, Lecture Notes in Computer Science **3653** (2005) 65–75.
8. Bruno Blanchet: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. Proceedings of CSFW 2001, 82–96.
9. Bruno Blanchet, Martín Abadi, Cédric Fournet: Automated Verification of Selected Equivalences for Security Protocols. Proceedings of LICS 2005, 331–340.
10. Stefan Blom, Wan Fokkink, Jan Friso Groote, Izak van Langevelde, Bert Lisser, Jaco van de Pol: μCRL: A Toolset for Analysing Algebraic Specifications. Proceedings of CAV 2001, Lecture Notes in Computer Science **2102** (2001) 250–254.
11. Stefan A. Brands: Untraceable Off-line Cash in Wallets with Observers (Extended Abstract). Proceedings of CRYPTO 1993, Lecture Notes in Computer Science **773** (1994) 302–318.
12. Stefan A. Brands: An Efficient Off-line Electronic Cash System Based On The Representation Problem. CWI Technical Report, CS-R9323 (1993).
13. David Chaum: The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. Journal of Cryptology **1** (1988) 65–75.
14. David Chaum, Amos Fiat, Moni Naor: Untraceable Electronic Cash. Proceedings of CRYPTO 1988, Lecture Notes in Computer Science **403** (1990) 319–327.
15. Tom Chothia: Analysing the MUTE Anonymous File-Sharing System Using the Pi-Calculus. Proceedings of FORTE 2006, Lecture Notes in Computer Science **4229** (2006) 115–130.
16. Tom Chothia, Simona Orzan, Jun Pang, Mohammad Torabi Dashti: A Framework for Automatically Checking Anonymity with μCRL. Proceedings of TGC 2006, Lecture Notes in Computer Science (to appear).
17. Yuxin Deng, Catuscia Palamidessi, Jun Pang: Weak Probabilistic Anonymity. Proceedings of SECCO 2005, Electronic Notes in Theoretical Computer Science (to appear).
18. Yuxin Deng, Jun Pang, Peng Wu: Measuring Anonymity with Relative Entropy. Proceedings of FAST 2006, Lecture Notes in Computer Science (to appear).
19. Niels Ferguson: Single Term Off-Line Coins. Proceedings of EUROCRYPT 1993, Lecture Notes in Computer Science **765** (1993) 318–328.
20. Matthew K. Franklin, Moti Yung: Secure and Efficient Off-Line Digital Money (Extended Abstract). Proceedings of ICALP 1993, Lecture Notes in Computer Science **700** (1993) 265–276.
21. Andrew Hinton, Marta Kwiatkowska, Gethin Norman, David Parker: PRISM: A Tool for Automatic Verification of Probabilistic Systems. Proceedings of TACAS 2006, Lecture Notes in Computer Science **3920** (2006) 441–444.
22. Tony Hoare: Communicating Sequential Processes. Communications of the ACM **21** (1978) 666–677.
23. Steve Kremer, Mark Ryan: Analysis of an Electronic Voting Protocol in the Applied Pi Calculus. Proceedings of ESOP 2005, Lecture Notes in Computer Science **3444** (2005) 186–200.
24. Gavin Lowe: Breaking and Fixing the Needham-Schroeder Public-Key Proceedings of TACAS 1996, Lecture Notes in Computer Science **1055** (1996) 147–166.
25. Oltea Mihaela Herescu, Catuscia Palamidessi: Probabilistic Asynchronous Pi-Calculus. Proceedings of FoSSaCS 2001, Lecture Notes in Computer Science **1784** (2001) 146–160.
26. Robin Milner, Joachim Parrow, David Walker: A Calculus of Mobile Processes, I. Information and Computation **100** (1992) 1–40.
27. Robin Milner, Joachim Parrow, David Walker: A Calculus of Mobile Processes, II. Information and Computation **100** (1992) 41–77.

28. Steve Schneider, Abraham Sidiropoulos: CSP and Anonymity. Proceedings of ESORICS 1996, Lecture Notes in Computer Science **1146** (1996) 198–218.
29. Yiannis Tsiounis: Efficient Electronic Cash: New Notions and Techniques. PhD Thesis, Northeastern University, 1997.