Secure Mobile Agents in eCommerce with Forward-Secure Undetachable Digital Signatures

Yang Shi, Qinpei Zhao, and Qin Liu

We introduce the idea of a forward-secure undetachable digital signature (FS-UDS) in this paper, which enables mobile agents to generate undetachable digital signatures with forward security of the original signer's signing key. The definition and security notion of an FS-UDS scheme are given. Then, the construction of a concrete FS-UDS scheme is proposed; and the proof of security for the proposed scheme is also provided. In the proposed scheme, mobile agents need not carry the signing key when they generate digital signatures on behalf of the original signer, so the signing key will not be compromised. At the same time, the encrypted function is combined with the original signer's requirement; therefore, misuse of the signing algorithm can be prevented. Furthermore, in the case where a hacker has accessed the signing key of the original signer, he/she is not able to forge a signature for any time period prior to when the key was obtained.

Keywords: Mobile agents, forward-secure, undetachable digital signatures.

Manuscript received June 3, 2014; revised Mar. 7, 2015; accepted Mar. 19, 2015.

Yang Shi (shiyang@tongji.edu.cn), Qinpei Zhao (qinpeizhao@tongji.edu.cn), and Qin Liu (corresponding author, sse508lab@126.com) are with the School of Software Engineering, Tongji University, Shanghai, China.

I. Introduction

With the fast development of distributed computing technologies, mobile agent systems and related technologies have attracted great interest. A mobile agent system consists primarily of *mobile agents* and *platforms*. An *agent* is a type of computer software that acts autonomously on behalf of a person or an organization [1].

Platforms that can create, execute, interpret, transfer, and terminate agents are *agent systems*. Mobile agents have a high mobility to transport themselves from one platform in a network to another and can automatically suspend execution on one platform and migrate to another to resume their computations. The ability to travel enables a mobile agent to move to a destination agent system that contains an entity in which the agent wants to interact.

The advent of electronic commerce practices has increased the demand for interoperable applications that allow for the flexible exchange of data across enterprise borders, different IT-platforms, and different information systems. For example, an intelligent trade agent (ITA) that roams a network [2] buying and selling goods through three different merchant's hosts within a network is shown in Fig. 1. Moreover, many mobile agent–based technologies have been developed and put into practice, such as those in [3]–[8].

The aforementioned applications cannot be realized securely without suitable security technologies. However, the fact is that mobile agents are exposed to serious security threats; for example, malicious hosts might endanger passing mobile agents. As security threats have become a bottleneck in the development and maintenance of mobile agent–based applications [9], there has become an urgent calling for efficient and practical security solutions that can achieve a good

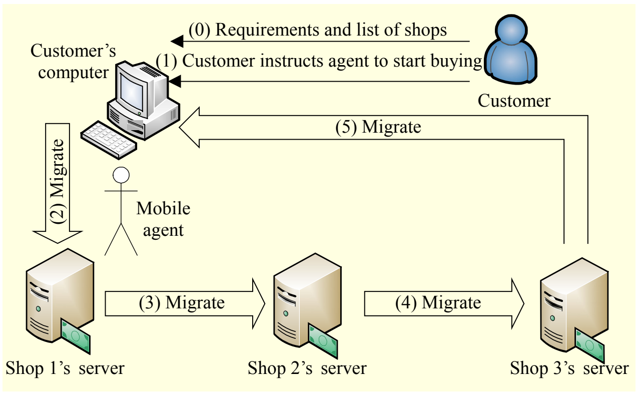
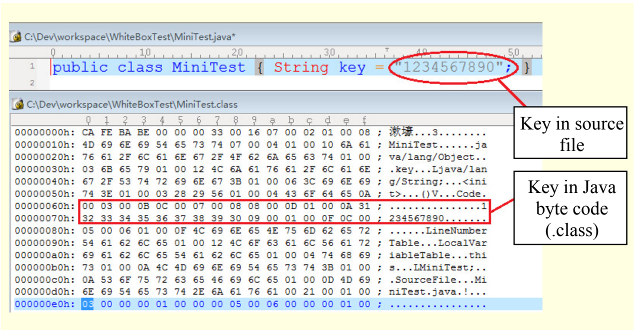


Fig. 1. ITA roams network buying and selling goods.



C:\Dev\workspace\WhiteBoxTest\MiniTest.java*

```
public class MiniTest { String key = "1234567890"; }
```

C:\Dev\workspace\WhiteBoxTest\MiniTest.class

```
            0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: CA FE BA BE 00 00 00 33 00 16 07 00 02 01 00 08 ; 漱壕...3........
00000010h: 4D 69 6E 69 54 65 73 74 07 00 04 01 00 10 6A 61 ; MiniTest......ja
00000020h: 76 61 2F 6C 61 6E 67 2F 4F 62 6A 65 63 74 01 00 ; va/lang/Object..
00000030h: 03 6B 65 79 01 00 12 4C 6A 61 76 61 2F 6C 61 6E ; .key...Ljava/lan
00000040h: 67 2F 53 74 72 69 6E 67 3B 01 00 06 3C 69 6E 69 ; g/String;...<ini
00000050h: 74 3E 01 00 03 28 29 56 01 00 04 43 6F 64 65 0A ; t>...()V...Code.
00000060h: 00 03 00 0B 0C 00 07 00 08 08 00 0D 01 00 0A 31 ; ...............1
00000070h: 32 33 34 35 36 37 38 39 30 09 00 01 00 0F 0C 00 ; 234567890.......
00000080h: 05 00 06 01 00 0F 4C 69 6E 65 4E 75 6D 62 65 72 ; ......LineNumber
00000090h: 54 61 62 6C 65 01 00 12 4C 6F 63 61 6C 56 61 72 ; Table...LocalVar
000000a0h: 69 61 62 6C 65 54 61 62 6C 65 01 00 04 74 68 69 ; iableTable...thi
000000b0h: 73 01 00 0A 4C 4D 69 6E 69 54 65 73 74 3B 01 00 ; s...LMiniTest;..
000000c0h: 0A 53 6F 75 72 63 65 46 69 6C 65 01 00 0D 4D 69 ; .SourceFile...Mi
000000d0h: 6E 69 54 65 73 74 2E 6A 61 76 61 00 21 00 01 00 ; niTest.java.!...
000000e0h: 03 00 00 00 01 00 00 00 05 00 06 00 00 00 01 00 ; ................
```

Fig. 2. Segment of Java byte code of agent.

balance between security and mobility in mobile agent–related studies.

Security problems of mobile agent systems stemming from the mobility of mobile agents can be categorized into the following two subjects [10]: agent-side security and platform-side security. This study focuses on agent-side security and aims to provide a practical security solution under mobility. Furthermore, the study addresses the worst situation of agent-side threats when the agents run in an untrusting environment or, more specifically, the agents are subject to attacks from malicious platforms. Because mobile agents frequently migrate among various hosts (platforms[1)]) due to their mobility, they run the risk of exposing themselves to untrusting environments more often than static agents. In such environments, the proposed forward-secure undetachable digital signature technique can provide the integrity and validity of business contracts, as well as safeguarding the confidentiality of the user's private signing key.

II. Mobile Agent Security and Undetachable Digital Signatures

The protection of a mobile agent from possible attacks by malicious hosts is referred to as the problem of malicious hosts. Hohl [11] identified a considerable number of such attacks including spying out code, data, and control flow; manipulation of code, data, and control flow and so on. Thus, threats from potentially malicious hosts have become a great obstacle to the widespread deployment of mobile agent technology–based applications in electronic commerce.

In traditional digital signature schemes, mobile agents need to carry the implementation of the signing algorithm with the private key to generate signatures on behalf of the original user, so an adversary can misuse the signing algorithm or even extract the private signing key from the agent. For example, an attacker who has control of a malicious host can find the signing key from Java byte code, as illustrated in Fig. 2 (that is, a screen snapshot of a Hex file editor).

Hence, Sander and Tschudin proposed the idea of undetachable digital signatures [12], which allows a mobile agent to effectively produce a digital signature inside a remote and possibly malicious host without the host being able to deduce the agent's secret (for example, the signing key) or reuse the implementation of the signing algorithm for arbitrary documents.

The general mathematical description of an undetachable signature function is as follows. Suppose that Sig is the signing algorithm used by C (a customer) to produce the digital signature z = Sig(m) for an arbitrary message m. The undetachable signing algorithm f_{Signed} should subject to (1), where f is an auxiliary binding function.

$$f_{\text{Signed}} = \text{Sig} \circ f. \qquad (1)$$

To create an undetachable signature on a shop's server, (2) is computed by calling f_{Signed} from the customer's mobile agent.

$$z = f_{\text{Signed}}(m). \qquad (2)$$

Although the signature function Sig is not known by others, everyone can verify the validity of message m by using a specialized verification algorithm.

Since the first concrete construction of an undetachable digital signature based on RSA [13] was published in 2001, a variety of undetachable signature schemes have been proposed [13]–[17]. The latest one to date, [18], is a threshold version using conic curves, which was proposed in 2013.

At the same time, digital signature schemes also face a significant key leakage problem on the original signer's local host, while undetachable signatures only protect the signing key from leakage on potentially malicious remote hosts. In the event that the customer's host is infiltrated by an attacker, there would be a serious security problem because the original

1) In the study, host and platform are used interchangeably to refer a place where mobile agents operate.

private signing key would be compromised.

The possibility of the occurrence of this security problem increases with the fast development of advanced persistent threats (APTs), such as "Operation Aurora" and "Stuxnet Worm."

Intuitively, a forward-secure signature technique could be used to control this risk. The principles behind forward-secure signature schemes are discussed briefly in [19]–[21]. In these schemes, the whole lifetime of a cryptosystem is divided into a number of time periods. At the end of each time period, the user computes a new signing key for the next time period using a one-way function and the old key is then erased. In this way, the user's singing key changes for different time periods, but the public key (verification key) is fixed during the whole lifetime. Each signature is associated with one time period, and the validity of a time period needs to be verified during signature verification. As a result, an adversary cannot forge any signature of any time period prior to the time when they obtained the signing key (for example, via cracking the host of the signer).

Traditional undetachable signature techniques (for example, [13]–[18]) mainly focus on protecting the signing key from being compromised, as well as preventing the misuse of the signature function/algorithm on malicious shop servers. On the other hand, normal forward-secure signature schemes do not fulfill the undetachable property, which is required to sign a contract securely by using a mobile agent on remote and potentially malicious hosts.

To fix this gap, we introduce the idea of a forward-secure undetachable digital signature (FS-UDS). The definition and security notion of an FS-UDS are given. Then, the construction of a concrete FS-UDS is proposed. The proposed FS-UDS enables mobile agents to generate undetachable digital signatures with forward security of the original signer's signing key. The main novelties of the scheme are as follows:

- It takes into account both the security of the customer's host and the security of shop servers.
- It is the first undetachable digital signature scheme that supports forward-secure key evolvement. Although there are many published forward-secure signature schemes, it is not a trivial work to design a scheme that fulfills the undetachable property given by equation (1).
- It is carefully designed such that the security of the scheme can be reduced to that of solving a hard mathematical problem (that is, the computational co-Diffie–Hellman problem on a co-GDH group pair).

To introduce the concrete scheme, we first give the formal definition and security notions of undetachable signature schemes in the next section. These are also contributions to theoretical aspects of mobile agent security.

III. Definition and Security Notions

1. Definition

In this section, the definition of an FS-UDS is proposed. Furthermore, the security notions of FS-UDS are given. Before the definition, we first list frequently used symbols in Table 1.

Definition 1. An FS-UDS FSUSig = (*KGen*, *KUpd*, *Sign*, *Vrfy*, *UndSigFunGen*, *UndSig*, *UndVrfy*) consists of the following seven algorithms:

- The key generation algorithm, *KGen*, takes as input a security parameter, 1^k, where $k \in \mathbb{N}$; the total number of periods T over which the scheme will operate; and possibly other parameters to return a base public key, U_0, and corresponding base secret key (signing key), s_0. The algorithm is probabilistic.
- The secret key update algorithm, *KUpd*, takes as input the secret signing key of the previous period, s_{j-1}, to return the secret signing key of the current period, s_j. The algorithm is usually deterministic.
- The undetachable signing function generation algorithm, *UndSigFunGen*, is a probabilistic polynomial-time algorithm, which takes the requirement of a customer, *REQ_C*; the customer's identity, ID_C; and the customer's public key and private key of the current period as inputs. The algorithm outputs a pair of functions, $f(\cdot)$ and $f_{\text{Signed}_j}(\cdot)$.
- The undetachable signing algorithm, *FSUndSig*, is a polynomial-time algorithm, which takes the contract (or a relative hash value) as input. The algorithm outputs an undetachable digital signature $z = \zeta_j = \langle \zeta, j \rangle$.
- The undetachable signature verification algorithm, *UndVrfy*, is a polynomial-time algorithm, which takes the contract and an undetachable signature z as input. The algorithm outputs either "Accept" or "Reject," simply "1" or "0."
- The signing algorithm, *Sign*, takes the secret signing key of the current period, s_j, and a message, M, to return a signature of M for period j. We write this as $\sigma_j \leftarrow Sign_{s_j}(M)$. The algorithm might be probabilistic. The signature is always a pair consisting of the value j of the current period and a tag σ.
- The verification algorithm, *Vrfy*, takes the public key (*PK*), message (*M*), and candidate signature $(\langle j, \sigma \rangle)$ to return a bit, with "1" meaning accept and "0" meaning reject. We write this as $b \leftarrow Vrfy_{U_0}(M, \langle j, \sigma \rangle)$.

2. Security Notions

For the formalization of the security definition, we set up

Table 1. Frequently used symbols.

Symbol	Description
1^k	Security parameter, $k \in \mathbb{N}$
U_0	Base public key
s_0	Base secret key (signing key)
j	Index of the time period
s_j / s_{j-1}	Secret signing key of the time period j / $j-1$
$CERT_j$	Certification (of underlying public key) of the period j
$REQ\_C \Vert ID_C$	Customer C's requirement and identity
f_{Signed_j}	Implementation of undetachable signing function of time period j
f	Auxiliary function of f_{Signed_j}
Msg	Message (usually a contract)
Z	Undetachable signature
σ, σ_j	Normal signatures

a fixed key-evolving undetachable signature scheme *FSUSig* = (*KGen*, *KUpd*, *Sign*, *Vrfy*, *UndSigFunGen*, *UndSig*, *UndVrfy*). An adversary, *F*, is viewed as functioning in three stages — the chosen message attack and chosen restriction attack phase; the break-in phase (breakin); and the forgery phase (forge). Its success probability in breaking the forward-security of the undetachable signature scheme is evaluated by the following two experiments. Note that we write $1^k, \ldots, T$ as the arguments to the key generation algorithm to indicate the possible presence of extra arguments.

```
Experiment FU-ForgeSig$(FSUndSig, F, 1^k, \ldots, T)$
$(U_0, s_0) \xleftarrow{\$} KGen(1^k, \ldots, T)$; $j \leftarrow 0$
Repeat
    $j \leftarrow j+1$; $s_j \leftarrow KUpd(s_{j-1})$
    $d \leftarrow F^{Sign_{s_j}(\cdot), UndSigFunGen_{s_j}(\cdot)}(CMA-CRA, U_0)$
Until$((d = breakin) \vee (j = T))$
If$((d \neq breakin) \wedge (j = T))$    $j \leftarrow T+1$
$(\langle \sigma, b \rangle, M) \leftarrow F(forge, s_j)$
If$\left(\begin{array}{l}(Vrfy_{U_0}(\langle \sigma, b \rangle, M) = 1) \wedge (1 \le b < j) \\ \wedge (M \notin L_b^{Sign}) \wedge (M \notin L_b^{UndSigFunGen})\end{array}\right)$   return 1
Else   return 0
```

In the experiment FU-ForgeSig, L_b^{Sign} and $L_b^{UndSigFunGen}$ are the query lists coming from the signing query $Sign_{s_b}(\bullet)$ of period b and the undetachable signature function generation query $UndSigFunGen_{SK_b}(\bullet)$ of period b, respectively.

```
Experiment FU-ForgeUndSig$(FSUndSig, F, 1^k, \ldots, T)$
$(U_0, s_0) \xleftarrow{\$} KGen(1^k, \ldots, T)$; $j \leftarrow 0$
Repeat
    $j \leftarrow j+1$; $s_j \leftarrow KUpd(s_{j-1})$
    $d \leftarrow F^{Sign_{s_j}(\cdot), UndSigFunGen_{s_j}(\cdot)}(CMA-CRA, U_0)$
Until$((d = breakin) \vee (j = T))$
If$((d \neq breakin) \wedge (j = T))$    $j \leftarrow T+1$
$(\langle \zeta, b \rangle, RES, M) \leftarrow F(forge, s_j)$
If$\left(\begin{array}{l}(UndVrfy_{U_0}(\langle \zeta, b \rangle, RES, M) = 1) \wedge (1 \le b < j) \\ \wedge (RES \notin T_b^{Sign}) \wedge (RES \notin T_b^{UndSigFunGen})\end{array}\right)$ return 1
Else   return 0
```

In the experiment FU-ForgeUndSig, T_b^{Sign} and $T_b^{UndSigFunGen}$ are the query lists coming from the signing query $Sign_{SK_b}(\bullet)$ of period b and the undetachable signature function generation query $UndSigFunGen_{SK_b}(\bullet)$ of period b, respectively.

Remark 1. *RES* is a restriction. It is usually in the form of *REQ*‖ID_C, where *REQ* is the requirement of a customer (*C*) and ID_C is the customer's identity.

Definition 2. The insecurity function of *FSUndSig* is defined as

$$\text{InSec}(FSUndSig, 1^k, \ldots, T; \tau, q_s, q_u) = \underset{F=\langle F_1, F_2 \rangle}{\text{Max}}$$

$$\left\{ \Pr\left[\begin{array}{l}\left(1 = \text{FU-ForgeSig}\left(FSUndSig, F_1, 1^k, \ldots, T\right)\right) \vee \\ \left(1 = \text{FU-ForgeUndSig}\left(FSUndSig, F_2, 1^k, \ldots, T\right)\right)\end{array}\right]\right\}. \quad (3)$$

The maximum here is over all $F = \langle F_1, F_2 \rangle$ for which the following are true: the execution time of the above two experiments, plus the size of the code of *F*, is at most *t*; *F* makes a total of at most q_s queries to the signing oracles and at most q_u queries to the undetachable signature function generation oracles across all the stages in all experiments.

Remark 2. The execution time is that of the entire experiment, not just that of *F*. So, it includes, in particular, the time to compute answers to oracle queries.

IV. Concrete Construction

1. Public Setting

The pubic setting $\Omega = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}(\cdot,\cdot), q, P, H_1(\cdot), H_2(\cdot))$ consists of the following components and algorithms:

- $\mathbb{G}_1$ is a multiplicative cyclic group of prime order *q*, $\mathbb{G}_2$ is also a multiplicative cyclic group of the same order.
- *G* and *P* are fixed generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

- $\hat{e}:\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map.
- $H_1:\{0,1\}^* \rightarrow Z_q^*$ and $H_2:\{0,1\}^* \rightarrow \mathbb{G}_1$ are two specialized hash maps.

Remark 3. We suppose that there exists an isomorphism $\psi:\mathbb{G}_2 \rightarrow \mathbb{G}_1$ with $\psi(P)=G$.

2. Mathematical Problems and Computational Assumptions

Definition 3. Decision co-Diffie–Hellman (co-DDH) problem on $(\mathbb{G}_1,\mathbb{G}_2)$: Given $P, P^a \in G_2$ and $Y, Y^b \in G_1$ as input, the solution of the problem is "yes" if $a = b$, otherwise it is "no." Note that when the solution is "yes," we say that (P, P^a, Y, Y^b) is a co-Diffie–Hellman tuple (co-DHT).

Assumption 1. We suppose that $\hat{e}$ can be effectively computed, so the co-DDH problem on $(\mathbb{G}_1,\mathbb{G}_2)$ is easy to solve.

Definition 4. Computational co-Diffie–Hellman (co-CDH) problem on $(\mathbb{G}_1,\mathbb{G}_2)$: Given $g_2,\ g_2{}^a \in \mathbb{G}_2$ and $h \in \mathbb{G}_1$ as input, the solution of the problem is (to compute) $h^a \in \mathbb{G}_1$.

Definition 5. We say that an algorithm (t',ε')-breaks co-CDH on $(\mathbb{G}_1,\mathbb{G}_2)$ if the algorithm runs in time (that is, within t') and successfully solves the co-CDH problem on $(\mathbb{G}_1,\mathbb{G}_2)$ with a probability of at least ε'.

Definition 6. Two groups, $(\mathbb{G}_1,\mathbb{G}_2)$, are a $(\lambda,t',\varepsilon')$ gap Diffie–Hellman group pair (co-GDH) if they satisfy the following properties:

- The group operation on both $\mathbb{G}_1$ and $\mathbb{G}_2$ and the map ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ can be computed in at most time λ.
- The co-DDH problem on $(\mathbb{G}_1,\mathbb{G}_2)$ can be solved in at most time λ.
- No algorithm (t',ε') breaks the co-CDH problem on $(\mathbb{G}_1,\mathbb{G}_2)$.

Assumption 2. The two order-q groups $(\mathbb{G}_1,\mathbb{G}_2)$ in the public setting are a $(\lambda,t',\varepsilon')$-bilinear group pair; that is, they satisfy the following properties:

- The group operation on both $\mathbb{G}_1$ and $\mathbb{G}_2$ and the map ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ can be computed in at most time λ.
- A group $\mathbb{G}_T$ of order q and a bilinear map $\hat{e}:\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ exist, and $\hat{e}$ is computable in at most time λ.
- No algorithm (t',ε')-breaks the co-CDH problem on $(\mathbb{G}_1,\mathbb{G}_2)$.

Because we can efficiently solve the co-DDH problem by computing two bilinear maps, if two groups $(\mathbb{G}_1,\mathbb{G}_2)$ are a $(\lambda,t',\varepsilon')$-bilinear group pair, then they are also a $(2\lambda,t',\varepsilon')$-co-GDH group pair.

Based on the above computational assumption, we propose a concrete and effective FS-UDS. The short signature scheme in [22] and the forward-secure signatures generation technique in [23] are used as building blocks in the proposed scheme.

3. Algorithms

The proposed FS-UDS consists of seven algorithms; that is, *KGen*, *KUpd*, *UndSigFunGen*, *UndSig*, *UndVrfy*, *Sig*, and *Vrfy*, as follows:

```
**Algorithm.** KGen (1^k)
$s_0 \xleftarrow{\$} Z_q^*; U_0 \leftarrow P^{s_0}$
For$(j=1; j \le T; j++)$ do
    $(s_j) \xleftarrow{\$} H_1(s_{j-1});\ U_j \leftarrow P^{s_j}$
    $CERT_j \leftarrow \left\langle U_0, j, U_j, H_2\left(U_0, j, U_j\right)^{s_0} \right\rangle$ EndFor
$erase\ \ s_j,\ j=1,\ldots,T;\ store\ \ CERT_j,\ j=1,\ldots,T$
$return\ \ \langle U_0, s_0 \rangle$
```

```
**Algorithm.** KUpd $(s_{j-1}, CERT_j, j, U_0)$
$\langle U_0', j', U_j', \Lambda_j \rangle \leftarrow CERT_j;\ s_j \leftarrow H_1(s_{j-1});\ U_j \leftarrow P^{s_j}$
If$\left(\hat{e}(\Lambda_j, P) \ne \hat{e}\left(H_2(U_0, j, U_j), U_0\right)\right)$ $return\ \ \perp$
$erase\ \ s_{j-1}$
$return\ \ s_j$
```

```
**Algorithm.** UndSigFunGen $\left(REQ\_C \,\|\, ID_C, sk_j, CERT_j\right)$
$H \leftarrow H_2(REQ\_C \,\|\, ID_C);\ K \leftarrow H^{s_j}$
$\sigma \leftarrow \langle CERT_j, \sigma' \rangle;\ \sigma_j \leftarrow \langle \sigma, j \rangle$
$setup\ \ f(x) = H^x\ \ \ f_{\mathrm{Signed}_j}(x) = \left\langle \langle CERT_j, K^x \rangle, j \right\rangle$
$return\ \ f(\cdot), f_{\mathrm{Signed}_j}(\cdot)$
```

```
**Algorithm.** UndSig (Msg)
$h = H_1(Msg);\ \ return\ \ f_{\mathrm{Signed}_j}(h)$
```

```
**Algorithm.** UndVrfy $\left(U_0, Z, j, Msg, REQ\_C \,\|\, ID_C\right)$
$\left\langle \langle CERT_j, Z' \rangle, j \right\rangle \leftarrow Z;\ \langle U_0', j', U_j', \Lambda_j \rangle \leftarrow CERT_j$
If$\left((U_0 \ne U_0') \vee (j \ne j')\right)$ $return\ \ 0$
If$\left(Msg\ does\ not\ satisfy\ REQ\_C\right)$ $return\ \ 0$
If$\left(\hat{e}(\Lambda_j, P) \ne \hat{e}\left(H_2(U_0', j', U_j'), U_0\right)\right)$ $return\ \ 0$
If$\left(\hat{e}(Z', P) \ne \hat{e}\left(H_2(REQ\_C \,\|\, ID_C)^{H_1(Msg)}, U_j\right)\right)$ $return$ 0
$else\ \ return$ 1
```

```
**Algorithm.** Sign $(s_j, j, Msg)$
$\sigma' \leftarrow H_2(Msg)^{s_j};\ \sigma \leftarrow \langle CERT_j, \sigma' \rangle;\ \sigma_j \leftarrow \langle \sigma, j \rangle$
$return\ \ \sigma_j$
```

```
**Algorithm.** $Vrfy\,(U_0,\sigma,j,Msg)$
$\langle CERT_j,\sigma'\rangle \leftarrow \sigma;\ \langle U_0',j',U_j',\Lambda_j\rangle \leftarrow CERT_j$
If $((U_0 \neq U_0') \vee (j \neq j'))$  return 0
If $(\hat{e}(\Lambda_j,P) \neq \hat{e}(H_2(U_0,j',U_j'),U_0))$  return 0
If $(\hat{e}(\sigma',P) \neq \hat{e}(H_2(Msg),U_j))$  return 0
Else  return 1
```

4. Proof of Correctness

The following proposition shows that the proposed scheme is a "correct" FS-UDS scheme.

Proposition 1. Let $Sig_{s_j,j}(x)=\langle\langle CERT_j,x^{s_j}\rangle,j\rangle$, for each $j\in\{1,2,\ldots,T\}$, then the function pair f_{Signed_j} and f generated by *UndSigFunGen* satisfies (1).

Proof. For any $j\in\{1,2,\ldots,T\}$, we have

$$(Sig_{s_j,j}\circ f)(x)=Sig_{s_j,j}(H^x)=\langle\langle CERT_j,(H^x)^{s_j}\rangle,j\rangle$$
$$=\langle\langle CERT_j,(H^{s_j})^x\rangle,j\rangle=\langle\langle CERT_j,K^x\rangle,j\rangle=f_{Signed_j}(x).$$

It is not difficult to validate the correctness of *UndVrfy* and *Vrfy*, so we omit the proofs. ■

5. Experimental Performance Analysis

We have implemented the algorithms in Java. Java has been used instead of C/C++ because many mobile agent platforms are developed in Java. An open-source Java Pairing-Based Cryptography Library (JPBC) [24] is used in our implementation. The performance is tested on both a portable PC and a server. The configurations of these platforms are listed in Table 2.

The performance results of the proposed algorithms on the platforms are listed in Table 3. The input and output sizes of the algorithms are also provided. Moreover, results of another undetectable digital signature scheme [16] without forward security is appended at the end of the table for comparison.

According to Table 3, all the algorithms in both schemes work fine on either a PC or a server. For algorithms that are frequently used in both schemes (that is, *Sign*, *Vrfy*, *UndSigFunGen*, *UndSig*, and *UndVrfy*), although the time and space cost of the proposed algorithms are a little more than the ones in [16], the largest difference is only 45 ms on time cost and 387 bytes on space cost. Importantly, the sacrifice of these small costs brings the forward-security. The new forward-secure signature scheme is capable of protecting the signing key in both the original signer's host and the remote servers. As for the *KeyGen* algorithm, it is only used once for each user in the initialization stage. Therefore, it brings little burden to the mobile agent system, even though the *KeyGen* algorithm needs more time than the one in [16].

6. Theoretical Security Analysis

In this subsection, we prove the security of the proposed scheme based on Assumption 2. The co-CDH problem is computationally infeasible in many algebra structures, such as Weil pairings; so, Assumption 2 is widely used in security schemes. We use this assumption as a mathematical basis for the proposed FS-UDS scheme.

Theorem 1. Suppose that Assumption 2 is true, then for the proposed FS-UDS scheme, and for any $\tau < t'$, the insecurity function that is defined in (3) satisfies (4).

$$\begin{aligned}&\text{InSec}(FSUndSig,1^k,\ldots,T;\tau,q_s,q_u)\\&\quad\le e(q_s^{(1)}+q_s^{(2)}+q_u^{(1)}+q_u^{(2)}+1)\cdot\varepsilon'.\end{aligned}\tag{3}$$

Proof. The first step of this proof is to adapt the forgery experiments to the proposed concrete construction as follows:

Table 2. Configurations of testing platforms.

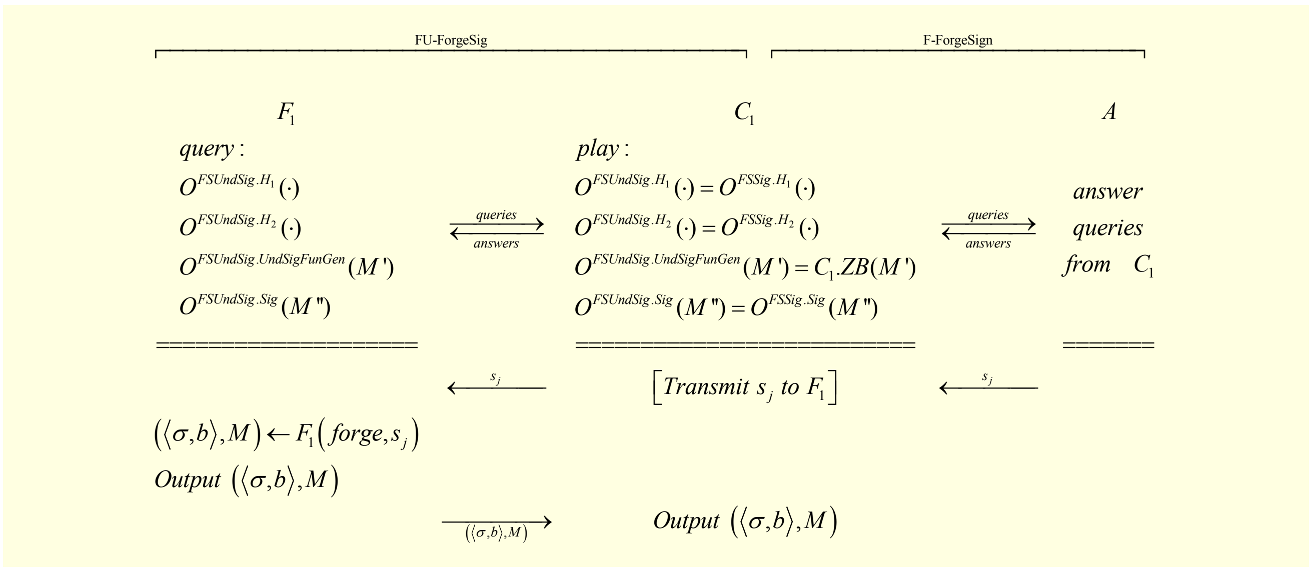
Platform	CPU	RAM	OS	JDK
PC	1.7 G 2Cores	4 GB	Win7	7.0
Server	2.5 G 6Cores × 2	96 GB	WinServ2008	7.0

Table 3. Results of performance tests.

Algorithm		Time (ms)		Size (byte)	
		PC	Server	Input	Output
Ours	*KGen* (*T*=10)	663	610	2	148
	KUpd	66	65	534	20
	Sign	46	44	149	514
	Vrfy	101	101	771	1
	UndSigFunGen	46	43	535	642
	UndSig	15	13	128	514
	UndVrfy	115	114	901	1
[16]	*KGen*	17	15	2	148
	Sign	52	50	148	128
	Vrfy	57	56	384	1
	UndSigFunGen	56	51	150	256
	UndSig	16	15	128	128
	UndVrfy	76	72	514	1

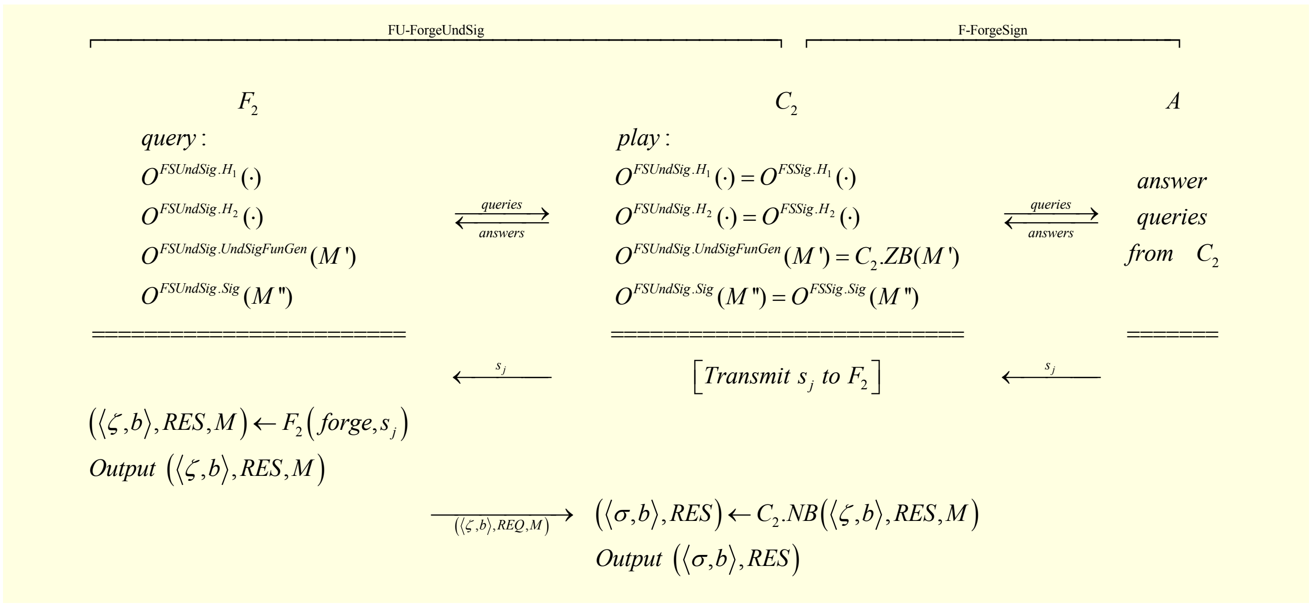
```
Game  FU-ForgeSig$\left(FSUndSig[1^k,r,T],F_1\right)$,
$\langle U_0,s_0\rangle \leftarrow KGen(1^k,r,T);\ j\leftarrow 0$
Repeat
    $j\leftarrow j+1;\ s_j\leftarrow KUpd(s_{j-1});$
    $d\leftarrow F_1\begin{bmatrix} O^{FSUndSig.Sign_{s_j}}(\cdot), \\ O^{FSUndSig.UndSigFunGen_{s_j}}(\cdot) \end{bmatrix}(CMA-CRA,U_0)$
Until$\left((d=breakin)\vee(j=T)\right)$
If $\left((d\neq breakin)\wedge(j=T)\right)$    $j\leftarrow T+1$
$\left(\langle\sigma,b\rangle,M\right)\leftarrow F_1\left(forge,s_j\right)$
If $\begin{pmatrix}\left(FSUndSig.Vrfy\left(U_0,\langle\sigma,b\rangle,M\right)=1\right) \\ \wedge(1\le b<j)\wedge\left(M\notin L_b^{Sign}\right)\wedge\left(M\notin L_b^{UndSigFunGen}\right)\end{pmatrix}$ return 1
Else   return 0
```

```
Game  FU-ForgeUndSig$\left(FSUndSig\left[1^k,r,T\right],F_2\right)$
$\langle U_0,s_0\rangle \leftarrow KGen\left(1^k,r,T\right);\ j\leftarrow 0$
Repeat
    $j\leftarrow j+1;\ s_j\leftarrow KUpd(s_{j-1});$
    $d\leftarrow F_1\begin{bmatrix} O^{FSUndSig.Sign_{s_j}}(\cdot), \\ O^{FSUndSig.UndSigFunGen_{s_j}}(\cdot) \end{bmatrix}(CMA-CRA,U_0)$
```



FU-ForgeSig
F-ForgeSign
F_1
C_1
A
$query:$
$O^{FSUndSig.H_1}(\cdot)$
$O^{FSUndSig.H_2}(\cdot)$
$O^{FSUndSig.UndSigFunGen}(M')$
$O^{FSUndSig.Sig}(M'')$
queries
answers
$play:$
$O^{FSUndSig.H_1}(\cdot)=O^{FSSig.H_1}(\cdot)$
$O^{FSUndSig.H_2}(\cdot)=O^{FSSig.H_2}(\cdot)$
$O^{FSUndSig.UndSigFunGen}(M')=C_1.ZB(M')$
$O^{FSUndSig.Sig}(M'')=O^{FSSig.Sig}(M'')$
queries
answers
answer queries from C_1
s_j
$\left[Transmit\ s_j\ to\ F_1\right]$
s_j
$\left(\langle\sigma,b\rangle,M\right)\leftarrow F_1\left(forge,s_j\right)$
$Output\ \left(\langle\sigma,b\rangle,M\right)$
$\left(\langle\sigma,b\rangle,M\right)$
$Output\ \left(\langle\sigma,b\rangle,M\right)$

Fig. 3. C_1 plays between F_1 and A.



FU-ForgeUndSig
F-ForgeSign
F_2
C_2
A
$query:$
$O^{FSUndSig.H_1}(\cdot)$
$O^{FSUndSig.H_2}(\cdot)$
$O^{FSUndSig.UndSigFunGen}(M')$
$O^{FSUndSig.Sig}(M'')$
queries
answers
$play:$
$O^{FSUndSig.H_1}(\cdot)=O^{FSSig.H_1}(\cdot)$
$O^{FSUndSig.H_2}(\cdot)=O^{FSSig.H_2}(\cdot)$
$O^{FSUndSig.UndSigFunGen}(M')=C_2.ZB(M')$
$O^{FSUndSig.Sig}(M'')=O^{FSSig.Sig}(M'')$
queries
answers
answer queries from C_2
s_j
$\left[Transmit\ s_j\ to\ F_2\right]$
s_j
$\left(\langle\zeta,b\rangle,RES,M\right)\leftarrow F_2\left(forge,s_j\right)$
$Output\ \left(\langle\zeta,b\rangle,RES,M\right)$
$\left(\langle\zeta,b\rangle,REQ,M\right)$
$\left(\langle\sigma,b\rangle,RES\right)\leftarrow C_2.NB\left(\langle\zeta,b\rangle,RES,M\right)$
$Output\ \left(\langle\sigma,b\rangle,RES\right)$

Fig. 4. C_2 plays between F_2 and A.

```
Until$\left(\left(d = breakin\right) \vee \left(j = T\right)\right)$
If $\left(\left(d \neq breakin\right) \wedge \left(j = T\right)\right)$   $j \leftarrow T+1$
$\left(\left\langle \zeta, b\right\rangle, RES, M\right) \leftarrow F_2\left(forge, s_j\right)$
If $\left(\begin{array}{l}\left(UndVrfy\left(U_0, \left\langle \zeta, b\right\rangle, RES, M\right)=1\right) \wedge \left(1 \le b < j\right)\\ \wedge\left(RES \notin T_b^{Sign}\right) \wedge \left(RES \notin T_b^{UndSigFunGen}\right)\end{array}\right)$ return 1
Else   return 0
```

The second step proceeds with the help of some intermediate players and transitivity security games. An intermediate player C_1 plays the game FU-ForgeSig with F_1 as shown in the left part of Fig. 3 and plays the game F-ForgeSig with another intermediate player A as shown in the right part of Fig. 3. Similarly, an intermediate C_2 plays the game FU-ForgeUndSig with F_1 as shown in the left part of Fig. 4 and plays the game F-ForgeSig with another intermediate player A as shown in the right part of Fig. 4.

As shown in Figs. 3 and 4, the intermediate players C_1 and C_2 have some algorithms that help them to play the games. The descriptions of these algorithms are as follows:

```
**Algorithm.** $C_2.NB\left(\left\langle \zeta, b\right\rangle, RES, M\right)$
$\left\langle\left\langle CERT_j, \zeta'\right\rangle, j\right\rangle \leftarrow \zeta; h \leftarrow O^{H_1}(M);$
$\sigma \leftarrow \left\langle\left\langle CERT_j, h^{-1}\cdot\zeta'\right\rangle, j\right\rangle$
$return \quad \left(\left\langle \sigma, b\right\rangle, RES\right)$
```

```
**Algorithm.** $C_i.ZB(RES, sk_j, CERT_j) \qquad i = 1,2$
$H \leftarrow O^{FSSig.H_2}(RES);\ \Delta \leftarrow O^{FSSig.Sig}(RES);$
$\left\langle CERT_j, K, j\right\rangle \leftarrow \Delta$
$setup \quad f(x) = H^x \quad f_{\mathrm{Signed}}(x) = \left\langle\left\langle CERT_j, K^x\right\rangle, j\right\rangle$
$return \quad f(\cdot), f_{\mathrm{Signed}}(\cdot)$
```

Moreover, the game F-ForgeSign is the normal security game that is widely used in the research of forward-secure signature schemes. F-ForgeSign proceeds as follows:

```
Game  F-ForgeSign$\left(FSSig[1^k, r, T], C_i\right)$   $// i=1,2$
$\left\langle U_0, s_0\right\rangle \leftarrow FSSig.KGen(1^k, r, T);\ j \leftarrow 0$
Repeat
    $j \leftarrow j+1;\ s_j \leftarrow FSSig.KUpd(s_{j-1});$
    $d \leftarrow C_i\left[O^{FSSig.Sig_{s_j}(\cdot)}\right]\left(CMA, U_0\right)$
Until$\left(\left(d = breakin\right) \vee \left(j = T\right)\right)$
If $\left(\left(d \neq breakin\right) \wedge \left(j = T\right)\right)$   $j \leftarrow T+1$
$\left(\left\langle \sigma, b\right\rangle, M\right) \leftarrow C_i(forge, s_j)$
If $\left(\left(FSSig.Vrfy\left(U_0, \left\langle \sigma, b\right\rangle, M\right)=1\right) \wedge \left(1 \le b < j\right) \wedge \left(M \notin V_b^{Sig}\right)\right)$
        return 1
Else   return 0
```

Similar to the second step, the third step involves the intermediate player A playing the game F-ForgeSign with C_i (i = 1 or 2) as shown in the left part of Fig. 5 and playing the game CMA with another player S as shown in the right part of Fig. 5. CMA is a standard message attack game, so we omit the description of this game. As shown in Fig. 5, the intermediate player A has some algorithms to help him play the games. The descriptions of these algorithms are as follows:

```
**Algorithm.** $A.Init(T, 1^k)$
$t \xleftarrow{\$} \{1, \ldots, T\};\ b \xleftarrow{\$} \{1, \ldots, t\}$
$HashList \leftarrow \varnothing;\ SKeyList \leftarrow \varnothing;\ PKeyList \leftarrow \varnothing$
For$\left(i = 0, i \le t; i++\right)$  do
    $SKeyList[i] \xleftarrow{\$} Z_q^*;\ PKeyList[i] \leftarrow P^{SKeyList[i]}$
EndFor
$U_b \leftarrow U^*$
For$\left(i = t+1, i \le T; i++\right)$  do
    $SKeyList[i] \leftarrow B.ZB\_H(s_{i-1});\ PKeyList[i] \leftarrow P^{SKeyList[i]}$
EndFor
$store \quad all \quad SKeyList, PKeyList$
```

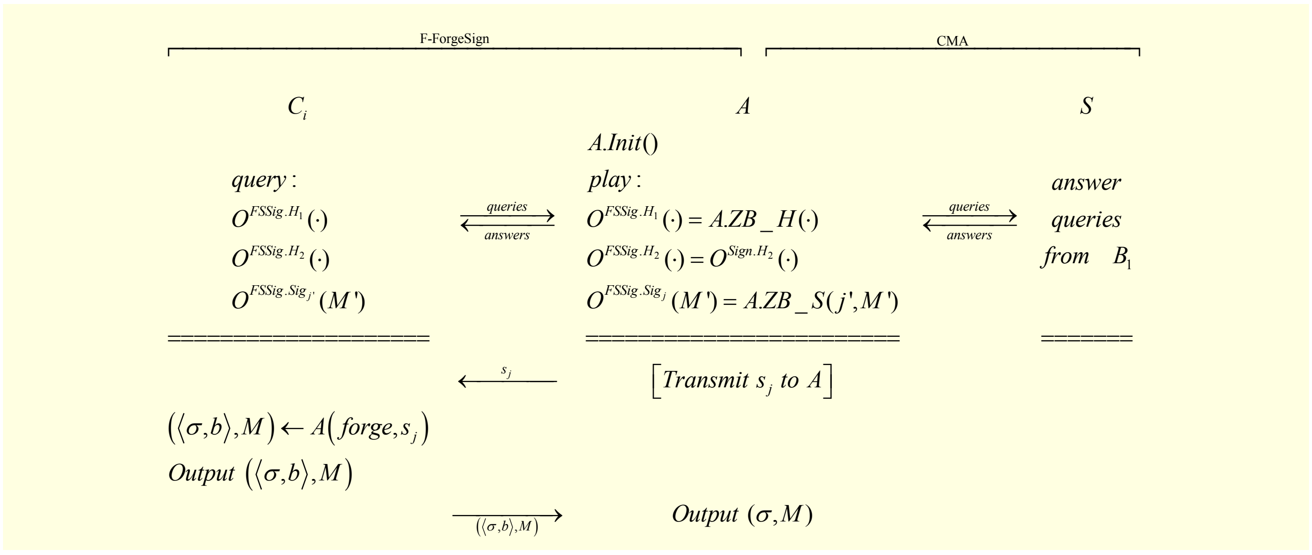
```
**Algorithm.** $A.ZB\_S(j, Msg)$
If$(b' = j)$  $return$  $O^{Sign.Sig}(Msg)$
Else  $return$  $H_2(Msg)^{SKeyList[j]}$
```

```
**Algorithm.** $A.ZB\_H\left(x\right)$
If$\left(\exists y, \left\langle x, y\right\rangle \in HashList\right)$  $return\ y$
$y \xleftarrow{\$} Z_q^*;\ HashList \leftarrow HashList \cup \left\{\left\langle x, y\right\rangle\right\};\ return \quad y$
```

After these steps, we can immediately find that a successful forgery of F can be reduced to a successful forgery of the BLS signature. Finally, we estimate the cost and probability of a successful forgery.

Suppose that F_i (i = 1 or 2) can win the game that queries $O^{FSUndSig.H_1}(\cdot)$, $O^{FSUndSig.H_2}(\cdot)$, $O^{IDSig}(\cdot,\cdot)$, $O^{H_3}(\cdot)$, and $O^{H_2}(\cdot)$ at most $q_{H_1}^{(i)}$, $q_{H_2}^{(i)}$, $q_u^{(i)}$, and $q_s^{(i)}$ times, respectively, and has a running time of τ_i and an advantage ε_i. That is, F_1 is a $\left\langle \tau_1, \varepsilon_1, q_{H_1}^{(1)}, q_{H_2}^{(1)}, q_s^{(1)}, q_u^{(1)}\right\rangle$ -forger of *FSUndSig.Sig* or F_2 is a $\left\langle \tau_2, \varepsilon_2, q_{H_1}^{(2)}, q_{H_2}^{(2)}, q_s^{(2)}, q_u^{(2)}\right\rangle$ -forger of *FSUndSig.UndSig*.

Let $c_{\exp}$ be the time of computing an exponentiation in $\mathbb{G}_1$ or $\mathbb{G}_2$, and let c_{rng} be the time of generating a random element of Z_q^*. We obtain the main result as follows, by calculating the


F-ForgeSign
CMA
C_i
A
S
$A.Init()$
$query:$
$play:$
$O^{FSSig.H_1}(\cdot)$
$O^{FSSig.H_2}(\cdot)$
$O^{FSSig.Sig_{j'}}(M')$
queries
answers
$O^{FSSig.H_1}(\cdot) = A.ZB\_H(\cdot)$
$O^{FSSig.H_2}(\cdot) = O^{Sign.H_2}(\cdot)$
$O^{FSSig.Sig_j}(M') = A.ZB\_S(j', M')$
answer queries from B_1
s_j
$[Transmit\ s_j\ to\ A]$
$(\langle\sigma,b\rangle, M) \leftarrow A(forge, s_j)$
$Output\ (\langle\sigma,b\rangle, M)$
$(\langle\sigma,b\rangle, M)$
$Output\ (\sigma, M)$
Fig. 5. A plays between C_i and S.

number of operations in the game:
For any τ , if

$$\tau \le \left(\begin{array}{l} t' - c_{\exp} \left(\begin{array}{l} 2\left(q_{H_2}^{(1)} + q_{H_2}^{(2)} + q_u^{(1)} + q_u^{(2)}\right) \\ + 3\left(q_s^{(1)} + q_s^{(2)} + q_u^{(1)} + q_u^{(2)}\right) + 2T \end{array} \right) \\ -\left(2T + q_{H_1}^{(1)} + q_{H_1}^{(2)}\right) \cdot c_{rng} \end{array} \right),$$

then

$$\mathrm{InSec}(FSUndSig, 1^k, \ldots, T; \tau, q_s, q_u) \\ \le e(q_s^{(1)} + q_s^{(2)} + q_u^{(1)} + q_u^{(2)} + 1) \cdot \varepsilon'. \qquad \blacksquare$$

7. Experimental Security Analysis

Besides the theoretical proof of security, experimental security analysis is provided to show that the security of the proposed scheme relies on the difficulty in solving the co-CDH problem. Actually, the analysis consists of two experiments. We firstly perform an experiment to show that if an attacker can break our scheme (that is, successfully forge an undetachable signature that can pass the verification of the undetachable signature verification algorithm *UndVrfy*), then the attacker is also capable of breaking the BLS signature scheme (that is, successfully forge a signature generated by the BLS signing algorithm). Secondly, we perform an experiment to show that a forgery of the BLS signature can support someone else to solve the co-CDH problem. The sketch of experimental security analysis is illustrated in Fig. 6.

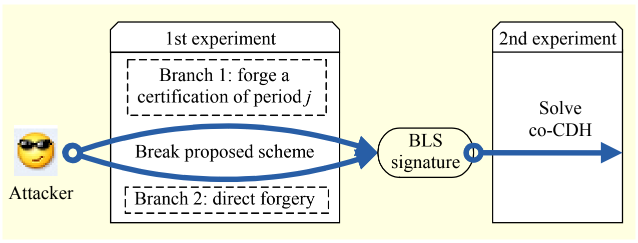
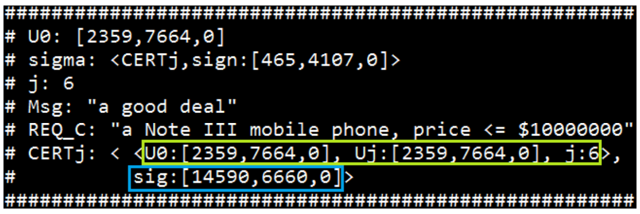


Fig. 6. Sketch of security experiments.


```
#################################################################
# U0: [2359,7664,0]
# sigma: <CERTj,sign:[465,4107,0]>
# j: 6
# Msg: "a good deal"
# REQ_C: "a Note III mobile phone, price <= $10000000"
# CERTj: < <U0:[2359,7664,0], Uj:[2359,7664,0], j:6>,
#         sig:[14590,6660,0]>
#################################################################
```
Fig. 7. Successful forgery of $CERT_j$.

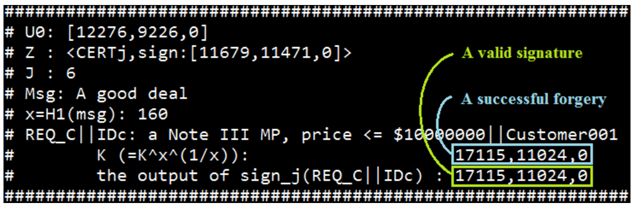

```
#################################################################
# U0: [12276,9226,0]
# Z : <CERTj,sign:[11679,11471,0]>
# J : 6
# Msg: A good deal
# x=H1(msg): 160
# REQ_C||IDc: a Note III MP, price <= $10000000||Customer001
#         K (=K^x^(1/x)):                       17115,11024,0
#         the output of sign_j(REQ_C||IDc) :    17115,11024,0
#################################################################
```

Fig. 8. Successful forgery of K.

In practical applications, sufficiently large security parameters should be used. In fact, in performance testing, we also use sufficiently large security parameters. However, to enable “an attacker” to successfully break the proposed scheme, very small security parameters (toy parameters) are used in the attacking experiments. Only in this way can the attacker break the scheme by brute-force guessing with limited computational capability.

The first experiment can be divided into two branches. One branch is where the attacker has forged a certification of the fake underlying public key of period j, then the attacker uses the fake signing key to generate a “valid” undetachable digital signature. A screen capture of the output is shown in Fig. 7. It is easy to verify that the value of "sig" in the blue rectangle is just a BLS signature on the hash value of the message in the upper rectangle. Another branch is where the attacker directly generates an undetachable digital signature. The certification of period j, $CERT_j$, is "real and valid." Thus, the attacker must generate a "valid" K that is used inv, otherwise the verification process would fail. Note that K is also a BLS signature on $REQ\_C \| ID_C$. A screen capture of the output is shown in Fig. 8.

According to the series of experiments, if an attacker is capable of breaking the proposed scheme, then the co-CDH problem could be solved. Hence, for sufficiently large security parameters, the proposed scheme is secure.

V. Comparison with Relative Works

The first implementation of an undetachable signature is the RSA-based version presented by Kotzanikolaou and others [13]. This was improved by Lee and others in [17] and by Shi and others in [15]. Han and others proposed a security scheme for e-Transactions using mobile agents with an agent broker [14]. They gave an undetachable signature function pair but did not present the signing function Sig subject to (1). Another undetachable signature scheme from pairings was proposed by Shi and others in [16], which is also based on the short signature scheme proposed in [22]. However, this proposed scheme provides forward security, which is not provided by the scheme in [16]. The latest published undetachable signature scheme was presented in 2013. The scheme [18] uses a cryptosystem based on conic curves and is in the form of a threshold digital signature.

As shown in Table 4, different schemes are based on different problems; however, most significantly, only the proposed scheme is forward-secure.

Table 4. Comparison of proposed scheme with other undetachable signature schemes.

Scheme	Computational infeasible problem	Forward-secure
[13]	Factorization of big integer	No
[17]	Factorization of big integer	No
[14]	q-Strong Diffie-Hellman problem	No
[15]	Discrete logarithm on elliptic curves	No
[16]	co-CDH	No
[18]	Factorization of big integer and discrete logarithm on conic curves	No
Ours	co-CDH	Yes

VI. Conclusion

In this paper, we proposed the idea of a forward-secure undetachable digital signature (FS-UDS) so that mobile agents can perform forward-secure signing operations on behalf of the original signers securely, even while they are running in a malicious host. We gave the formal definition of an FS-UDS scheme. Then, we provided the security notion of forward-secure undetachable signature schemes as a theoretical basis. A concrete scheme with provable security was proposed for protecting mobile agents in electronic commerce. The proposed scheme is constructed on co-GDH group pairs, and its security relies on the infeasibility in solving computational co-Diffie–Hellman problems.

An implementation of the proposed undetachable signature algorithm can securely migrate with mobile agents from one host to another without the signing algorithm being at risk from misuse or the signing key being compromised. Moreover, even though a customer's signing key (in a given time period) has been compromised, any adversary cannot forge any contracts that were signed before the compromising because this scheme is forward-secure.

References

[1] D. Milojicic et al., "MASIF: The OMG Mobile Agent System Interoperability Facility," *Personal Technol.*, vol. 2, no. 2, Feb. 1998, pp. 117–129.

[2] S.H. von Solms, "Electronic Commerce with Secure Intelligent Trade Agents," *Comput. Security*, vol. 17, no. 5, May 1998, pp. 435–446.

[3] Y.-F. Chung et al., "An Agent-Based English Auction Protocol Using Elliptic Curve Cryptosystem for Mobile Commerce," *Expert Syst. Appl.*, vol. 38, no. 8, Aug. 2011, pp. 9900–9907.

[4] A.J.C. Trappey, C.V. Trappey, and F.T.L. Lin, "Automated Silicon Intellectual Property Trade Using Mobile Agent Technology," *Robot. Comput.-Integr. Manuf.*, vol. 22, no. 3, July 2006, pp. 189–202.

[5] G. Wang, T.N. Wong, and X.H. Wang, "A Hybrid Multi-agent Negotiation Protocol Supporting Agent Mobility in Virtual Enterprises," *Inf. Sci.*, vol. 282, no. 20, Oct. 2014, pp. 1–14.

[6] T.C. Du, E.Y. Li, and E. Wei, "Mobile Agents for a Brokering Service in the Electronic Marketplace," *Decision Support Syst.*, vol. 39, no. 3, May 2005, pp. 371–383.

[7] A. Aloui, O. Zerdoumi, and O. Kazar, "Architecture for Mobile Business Based on Mobile Agent," *Multimedia Comput. Syst.*, Tangier, Morocco, May 10–12, 2012, pp. 954–958.

[8] T.N. Wong and F. Fang, "A Multi-agent Protocol for Multilateral Negotiations in Supply Chain Management," *Int. J. Production Res.*, vol. 48, no. 1, Jan. 2010, pp. 271–299.

[9] S. Knight, S. Buffett, and P.C.K. Hung, "The International Journal of Information Security Special Issue on Privacy, Security and Trust

Technologies and E-Business Services - Guest Editors' Introduction," *Int. J. Inf. Security*, vol. 6, no. 5, Sept. 2007, pp. 285–286.

[10] Y. Jung et al., "A Survey of Security Issue in Multi-agent Systems," *Artif. Intell. Rev.*, vol. 37, no. 3, Mar. 2012, pp. 239–260.

[11] F. Hohl, "Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts," in *Mobile Agents Security*, Berlin, Germany: Springer, 1998, pp. 92–113.

[12] T. Sander and C. Tschudin, "Protecting Mobile Agents against Malicious Hosts," in *Mobile Agents Security*, Berlin, Germany: Springer, 1998, pp. 44–60.

[13] P. Kotzanikolaou, M. Burmester, and V. Chrissikopoulos, "Secure Transactions with Mobile Agents in Hostile Environments," in *Information Security Privacy*, Berlin, Germany: Springer, 2000, pp. 289–297.

[14] S. Han, E. Chang, and T. Dillon, "Secure E-Transactions Using Mobile Agents with Agent Broker," *Int. Conf. Services Syst. Services Manag.*, Chongqing, China, June 13–15, 2005, pp. 849–855.

[15] Y. Shi, L. Cao, and X. Wang, "A Security Scheme of Electronic Commerce for Mobile Agents Uses Undetachable Digital Signatures," *Int. Conf. Inf. Security*, Shanghai, China, Nov. 14–15, 2004, pp. 242–243.

[16] Y. Shi et al., "Secure Mobile Agents in Electronic Commerce by Using Undetachable Signatures from Pairings," *Int. Conf. Electron. Business*, Beijing, China, Dec. 5–9, 2004, pp. 1038–1043.

[17] B. Lee, H. Kim, and K. Kim, "Secure Mobile Agent Using Strong Non-designated Proxy Signature," in *Information Security Privacy*, Berlin, Germany: Springer, 2001, pp. 474–486.

[18] Y. Shi and G.Y. Xiong, "An Undetachable Threshold Digital Signature Scheme Based on Conic Curves," *Appl. Math. Inf. Sci.*, vol. 7, no. 2, Mar. 2013, pp. 823–828.

[19] J. Yu et al., "Forward-Secure Identity-Based Signature: Security Notions and Construction," *Inf. Sci.*, vol. 181, no. 3, Feb. 2011, pp. 648–660.

[20] Y.-C. Yu, T.-Y. Huang, and T.-W. Hou, "Forward Secure Digital Signature for Electronic Medical Records," *J. Med. Syst.*, vol. 36, no. 2, Apr. 2012, pp. 399–406.

[21] C.-I. Fan, Y.-H. Lin, and R.-H. Hsu, "Complete EAP Method: User Efficient and Forward Secure Authentication Protocol for IEEE 802.11 Wireless LANs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 4, Apr. 2013, pp. 672–680.

[22] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," *J. Cryptology*, vol. 17, no. 4, Sept. 2004, pp. 297–319.

[23] H. Krawczyk, "Simple Forward-Secure Signatures from Any Signature Scheme," *ACM Conf. Comput. Commun. Security*, Athens, Greece, Nov. 1–4, 2000, pp. 108–115.

[24] A. De Caro and V. Iovino, "JPBC: Java Pairing Based Cryptography," *IEEE Symp. Comput. Commun.*, Kerkyra, Greece, June 28–July 1, 2011, pp. 850–855.

Yang Shi received his BS degree in electronic engineering from Hefei University of Technology, China, in 1999 and his MS degree in pattern recognition and intelligence systems from Kunming University of Science and Technology, China, in 2002. He obtained his PhD degree in pattern recognition and intelligent systems from Tongji University, Shanghai, China, in 2005. From 2005 to 2011, he worked for Pudong CS&S Co. Ltd., Shanghai, China. Since 2012, he has been an associate professor with the School of Software Engineering, Tongji University, His research interests include information security, software privacy, and data engineering.

Qinpei Zhao received her BS degree in automation technology from Xi'dian University, China, in 2004. She received her MS degree in pattern recognition and image processing from Shanghai Jiaotong University, China, in 2007. She obtained her PhD degree in computer science from the School of Computing, University of Eastern Finland, Joensuu, Finland, in 2012. Since 2013, she has been a lecturer at the School of Software Engineering, Tongji University, Shanghai, China. Her current research interests include clustering algorithms and multimedia processing; location-based services; and security.

Qin Liu received her BS degree in industrial automation from Dalian University of Technology and Science, China, in 1998; her MS degree in software engineering from Southampton Solent University, UK, in 2001; and her PhD degree in software engineering from Northumbria University, Newcastle, UK, in 2006. Since 2007, she has been with the School of Software Engineering, Tongji University, Shanghai, China. Currently, she is both a professor and the executive dean of the School of Software Engineering, Tongji University. Her research interests include software measurement; information security and privacy; data mining; and data analysis.