Research Article

Key-Insulated Undetachable Digital Signature Scheme and Solution for Secure Mobile Agents in Electronic Commerce

Yang Shi,¹ Jie Lin,² Guoyue Xiong,² Xiaoping Wang,³ and Hongfei Fan¹

¹*School of Software Engineering, Tongji University, 4800 Cao'an Highway, Shanghai 201804, China*

²*School of Economics and Management, Tongji University, 4800 Cao'an Highway, Shanghai 201804, China*

³*Department of Computer Science and Technology, Tongji University, 4800 Cao'an Highway, Shanghai 201804, China*

Correspondence should be addressed to Hongfei Fan; fanhongfei@tongji.edu.cn

Received 17 September 2015; Revised 30 March 2016; Accepted 11 April 2016

Academic Editor: Laurence T. Yang

Copyright © 2016 Yang Shi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Considering the security of both the customers' hosts and the eShops' servers, we introduce the idea of a key-insulated undetachable digital signature, enabling mobile agents to generate undetachable digital signatures on remote hosts with the key-insulated property of the original signer's signing key. From the theoretical perspective, we provide the formal definition and security notion of a key-insulated undetachable digital signature. From the practical perspective, we propose a concrete scheme to secure mobile agents in electronic commerce. The scheme is mainly focused on protecting the signing key from leakage and preventing the misuse of the signature algorithm on malicious servers. Agents do not carry the signing key when they generate digital signatures on behalf of the original signer, so the key is protected on remote servers. Furthermore, if a hacker gains the signing key of the original signer, the hacker is still unable to forge a signature for any time period other than the key being accessed. In addition, the encrypted function is combined with the original signer's requirement to prevent the misuse of signing algorithm. The scheme is constructed on gap Diffie-Hellman groups with provable security, and the performance testing indicates that the scheme is efficient.

1. Introduction

Agents are a type of computer program that acts autonomously on behalf of a person or an organization. Mobile agents can easily transport themselves from one system in a network to another. They can also automatically suspend execution on one platform and migrate to another to resume their computations. Compared with traditional computing models (e.g., client/server), mobile agent technology has several significant advantages in electronic commerce applications, including autonomy and fault tolerance [1, 2].

However, these benefits cannot be achieved without suitable security and trust technologies, which are critical for ensuring that all business data has been appropriately protected and business partners can collaborate with integrity and confidence. One significant threat is that malicious hosts might endanger passing agents because the attacks from

owners of hosts are too strong to defeat with traditional security countermeasures. As a typical example, a mobile agent needs to sign a contract on behalf of the original signer (the customer) when an appropriate product has been found, while an attacker controlling the host can extract the secret signing key from the agent's code or even generate a signature on the contract by simply forcefully calling the signing function that is "carried" on the agent. Furthermore, the risk of leakage of the signing key on the customer's PC may stem from a variety of threats such as the "SSL Heartbleed."

Therefore, in the research field of mobile agent security, it is a challenge to protect the digital signature functionality against attacks from remote malicious hosts and meanwhile control the security risk of signing key leakage on the original signer's host. Motivated by this security challenge, we propose a category of digital signature schemes for

mobile agents, namely, the *key-insulated undetachable digital signature* (KIUDS). Major contributions of this work are previewed as follows:

- (i) We contributed the novel idea of key-insulated undetachable digital signature, which enables mobile agents to generate undetachable digital signatures on remote hosts with the key-insulated property of the original signer's signing key.
- (ii) Theoretically, we provided the formal definition and security notion of the KIUDS schemes.
- (iii) Practically, we proposed a concrete KIUDS scheme for securing mobile agents in electronic commerce.
- (iv) We provided the security proof and efficiency analysis of the proposed scheme and conducted the performance testing which has indicated the good efficiency of the scheme.

The rest of this paper is organized as follows. Section 2 presents backgrounds and preliminaries of this work, including the mobile agent systems and applications in electronic commerce, the undetachable digital signature schemes and the key-insulated signature schemes, and the security analysis of threats on mobile agents from malicious hosts. The formal definition and security notion of KIUDS schemes are provided in Section 3. A concrete KIUDS scheme is described in Section 4, along with the theoretical analysis in terms of security, correctness, and computational costs, as well as a set of experimental results. Comparisons with related works are presented in Section 5. Finally, the paper concludes with a discussion on the findings.

2. Backgrounds and Preliminaries

2.1. Mobile Agent Systems and Applications in Electronic Commerce. An agent is a software entity that acts autonomously on behalf of a person or organization. Each agent has its own thread of execution, so tasks can be performed on its own initiative. A mobile agent is not bound to the system where it begins execution. It has the capability of transporting itself from one host to another in a network. We briefly introduce the backgrounds of mobile agents and the related system architecture [11] as follows.

During an agent's travel process, the agent state (including the execution state and the attributes) and code are being transported. The agent's authority identifies the person or organization for which the agent acts. The names of the agents are usually required for identification, management, and locating. Commonly, agents are named by their authority, identity, and agent system type, whose combination can be mapped into a unique value for identifying a particular agent instance within the scope of the authority.

By definition, an agent system is an integrated platform that can create, interpret, execute, transfer, and terminate agents. An agent system is usually associated with an authority that identifies the person or organization for which the agent system acts. An agent system is uniquely identified by its name and address, and a host can contain one or more

agent systems. An instance of the architecture of mobile agent system is illustrated in Figure 1.

Agents transfer themselves between *places* over the network, where a *place* hereby is a context in which an agent executes. An agent is associated with a location, which consists of the place name and the address of the agent system where the place resides. An agent system may contain one or multiple places while a place can also host a number of agents. If an agent system does not support places, then it acts as a default place. When a client requests for the location of an agent, it retrieves the address of the place where the agent is executing.

The agent technology seems an attractive paradigm to support e-commerce applications [12, 13], because agents are capable of acting on behalf of customers to reduce the effort required for performing transactions. Agents are autonomous by nature and therefore can be easily personalized to embody customers' preferences. In addition, they are adaptive in terms of the capabilities of learning from both past actions and their environment, coping with changing network conditions and evolving user requirements. The above features permit the agent technology to add value to three primary e-commerce dimensions: information filtering, information gathering and retrieval, and dynamic and flexible execution of transactions. As typical applications, agent-based electronic commerce, product recommendation, and decision making have been demonstrated in the recent years [14–16].

Furthermore, mobile agents not only can autonomously act and negotiate on behalf of their creators in one host, but also can autonomously decide to move itself from one host to another as necessary [13]. Such mobility achieves better network utilization and allows mobile users to disconnect with their agents roaming in the network, thus reducing connection costs. In the context of electronic marketplaces, for example, the mobility permits the agent to perform all required operations locally in the involved marketplace without reliable connection or bandwidth engagement.

Some eShop and e-marketplace applications based on mobile agent technology have been recently proposed. A typical application of mobile agents in electronic commerce is shown in Figure 2. The figure illustrates an intelligent trade agent (ITA) that roams the Internet buying goods or services from the servers of three eShops in the network. Other applications include a silicon intellectual property automatic trading platform [17], an agent-based English auction protocol using an elliptic curve cryptosystem for mobile commerce [18], a novel multiagent system architecture to cope with the flexibility requirements of virtual enterprises [19], and an approach aiming to deploy the mobile agent paradigm for mobile business applications [20].

2.2. Digital Signature Schemes and Attack Models. Digital signature schemes allow a signer who has established a public key to sign a message such that any other party can verify that the message originated from the signer and was not modified in any way. A digital signature scheme typically consists of three algorithms: the key generation algorithm, the signing algorithm, and the verification algorithm.

FIGURE 1: Mobile agent systems and architecture.

FIGURE 2: ITA roaming a network.

There are three main attack models according to the capabilities of an adversary to attack cryptosystems [21], for example, a digital signature scheme. The first is the black-box model. It is a traditional attack model where an adversary only has access to the functionality of a cryptosystem. This limited access increases the effort and is time required to start attacks. The second is the grey-box model, which refers to a model where a leakage function is present. In such an attack context, the adversary can deploy side-channel cryptanalysis techniques. Owing to the large variety of leakage functions, the grey-box model can further be classified into several subgroups. The third is the white-box model, where the adversary has total visibility of the software implementation of the cryptosystem and full control over its execution platform.

As shown in Figure 3, the white-box model is the worst-case model. The white-box model is used to analyze algorithms that are running in a nontrustable environment, that is, an environment in which applications are subject to attacks from the execution platform.

Secure computing in a white-box model is a challenge because the model assumes that [22] (1) fully privileged attack

software shares a host with the cryptographic software, having complete access to the implementation of the algorithms, (2) dynamic execution (with instantiated cryptographic keys) can be observed, and (3) internal details of cryptographic algorithms are both completely visible and alterable.

2.3. Attacks and Threats against Mobile Agents and Their Signing Routines. While mobile agent-based technologies are already in use, mobile agents are still exposed to serious security threats. Mobile agent-based applications cannot be successfully implemented without suitable security technologies to ensure that the sensitive business data are appropriately protected and business partners can work together with integrity and confidence. One of the most challenging threats to mobile agent security is attacks from platforms (malicious hosts). In general, there are two main kinds of attacks from malicious hosts [23], eavesdropping attacks and manipulation attacks. A substantial number of attacks are identified in detail in [24], including spying out code, spying out data, spying out control flow, manipulation of code, manipulation of data, manipulation of control flow, incorrect execution of code, masquerading of the host, denial of execution, spying

FIGURE 3: Attack models.

out interaction with other agents, manipulation of interaction with other agents, and returning wrong system call results issued by the agent.

Clearly, mobile agents executed on a malicious host are in a white-box attack context. In traditional digital signature schemes, mobile agents must carry the private key when they generate digital signatures on behalf of the original user. Possible attacks from malicious remote hosts endanger the digital signature functionality provided by a mobile agent because the signing algorithm may be misused and the signing key may be compromised. Furthermore, many mobile agent systems are implemented in Java because of the language's suitable performance across various platforms. Unfortunately, the key in a class file (the Java byte code) is extremely vulnerable, as illustrated in Figure 4. Hence, traditional digital signature schemes are not suitable for mobile agents, while new signature techniques are required.

Meanwhile, digital signature functionalities also face a significant key leakage problem on the local host of the original signer. A serious security issue is presented in the event that the original signer's local host is under the control of a hacker because the original private signing key may be compromised.

The severity of the potential threats increases with the rapid development of advanced persistent threats (APTs) such as "Operation Aurora" against Gmail accounts, the "Stuxnet Worm" against the control system of nuclear devices in Iran, "Operation Shady RAT" against more than 70 organizations (including several defense contractors), and the recently discovered "Havex" against industrial control

systems. An APT is a set of stealthy and continuous hacking processes often orchestrated by people targeting a specific entity. Additionally, security vulnerabilities such as the "SSL Heartbleed," which was disclosed in April 2014 in the OpenSSL cryptography library, could also enable hackers to extract information from remote hosts.

2.4. Undetachable Digital Signatures for Mobile Agents. The idea of the undetachable digital signature was proposed by Sander and Tschudin in [25] based on the reasoning that mobile agents do not have to be executed in clear text form. The undetachable digital signature technique allows a mobile agent to effectively produce a digital signature inside a remote and possibly malicious host without allowing the host to deduce the agent's secret or to reuse the signature routine for arbitrary documents. A brief introduction to this idea is presented as follows.

Let Sig be a rational function used by C (a customer) to produce the digital signature $\text{Sig}(m)$ of an arbitrary message m . Furthermore, suppose the message m is the result of a rational function f applied to some input data x . Finally, the verification function Ver that C publishes to allow others to check the validity of the digital signature z is regarded to be a valid signature of m if and only if

$$z = \text{Sig}(m). \quad (1)$$

To allow the customer's mobile agent to create "undetachable" signatures, the following is computed:

$$f_{\text{Signed}} = \text{Sig} \circ f, \quad (2)$$

FIGURE 4: Segment of the Java byte code of an agent.

where f is an auxiliary binding function that binds a constraint (e.g., a restriction or a limitation) on the signing function. For example, in the case of electronic commerce, a typical constraint could be “an iPhone 6 costing no more than 916 Dollars.” f_{Signed} and f are then migrated to S (the eShop) with the mobile agent. S evaluates

$$\begin{aligned} m &= f(x) \\ z &= f_{\text{Signed}}(x). \end{aligned} \quad (3)$$

However, Sander and Tschudin did not provide a concrete implementation of the undetachable digital signature scheme. The first undetachable digital signature scheme was proposed by Kotzanikolaou et al. in the form of an RSA implementation [3]. This was improved by Lee et al. [4] with bilateral security, whose scheme provides server’s nonrepudiation because it contains server’s signature at the same time. Han et al. proposed a security scheme for e-transactions using mobile agents with an agent broker [5], while they gave an undetachable signature function pair but without presenting the signing function Sig subject to (2). Another undetachable signature scheme based on pairings was proposed in prior work [8], which is based on the short signature scheme proposed in prior work [26]. To solve the problem in which a host may force an agent to commit to a suboptimal transaction, Borselius et al. [6] introduced the notion of undetachable threshold signatures and proposed an RSA-based implementation. In prior work [9], an implementation of undetachable threshold digital signature based on conic curves was proposed, and computational studies indicated that the implementation in [9] was superior to the RSA-based implementation in [6]. The latest published undetachable signature scheme was presented in prior work [10], which provides forward security by following the BLS short signature [26]. However, except for the scheme in [10], none of the proposed undetachable digital signature schemes can simultaneously protect digital signature functionalities against attacks from remote malicious hosts and mitigate the security risk of signing key leakage on the original signer’s host. Moreover, even with the scheme in [10], an adversary can forge any signature during any period after he/she has got the signing key (e.g., via cracking the host of the original signer). Therefore, stronger undetachable digital signature schemes are demanded, which motivate us to develop undetachable signature schemes with the key-insulated property.

2.5. Key-Insulated Digital Signatures. The exposure of secret keys is perhaps the most devastating attack on a cryptosystem because it typically leads to a full loss of security. This problem is probably the greatest threat to cryptography in the real world: in practice, it is typically easier for an adversary to obtain a secret key from a naive user than to break the computational assumption on which the system is based. This threat is currently increasing as new APTs are quickly developed.

Complete prevention of key exposure—even for the original signer on his local host—usually requires some degree of physical security, which can be expensive and inconvenient. Thus, some security countermeasures assume that key exposure will inevitably occur and therefore focus on minimizing the damage which results when keys are obtained by an attacker.

A category of such damage-minimization security countermeasures is key-insulated cryptography [27, 28]. In the security model of key-insulated cryptography, physical security (and hence the secrecy of stored data) is guaranteed for a single device that holds a “master” secret key corresponding to a fixed public key. Day-to-day cryptographic operations such as signing a message, however, are performed by an insecure device (e.g., the customer’s PC) which “refreshes” its private key periodically by interacting with the physically secure device. In a (t, N) -key-insulated cryptosystem, an attacker who compromises the insecure device and obtains secret keys for up to t time periods is unable to violate the security of the cryptosystem for any of the remaining $N-t$ periods. The principle of key-insulated signature schemes is illustrated in Figure 5.

3. Key-Insulated Undetachable Digital Signature

In this section, we propose a novel category of digital signature schemes for mobile agents, the KIUDS scheme. This scheme simultaneously protects the digital signature functionality against attacks from remote malicious hosts and mitigates the security risk of signing key leakage on the original signer’s host. We begin by defining the KIUDS scheme and then provide its security model and security notion.

3.1. Definition. A KIUDS scheme $KIUDS = (KGen, Upd^*, Upd, UndSigFunGen, UndSig, UndVrfy)$ consists of six algorithms as follows:

FIGURE 5: Principle of the key-insulated signature.

- (1) $KGen$, the key generation algorithm, is a probabilistic algorithm taking as input a security parameter 1^k ($k \in \mathbb{N}$) and the total number of time periods N . It returns a public key P_{pub} , a master secret signing key s , and an initial key D_0 .
- (2) Upd^* , the device key update algorithm, is a probabilistic algorithm taking as input indices i, j for time periods (throughout, we assume $1 \leq i, j \leq N$) and the master key D . It returns a partial secret key $D'_{i,j}$.
- (3) Upd , the user key update algorithm, is a deterministic algorithm taking as input indices i, j , a secret key D_i , and a partial secret key $D'_{i,j}$. It returns the secret key D_j for time period j .
- (4) $UndSigFunGen$, the undetachable signing function generation algorithm, is a probabilistic polynomial time algorithm which takes the requirement of a customer REQ_C , the customer's identity ID_C , and the index of the current time period j as inputs. The algorithm outputs a function $f_{Signed,j}(\cdot)$.
- (5) $UndSig$, the undetachable signing algorithm, is a polynomial time algorithm which takes the contract (or its hash value) as input. The algorithm outputs an undetachable signature z .
- (6) $UndVrfy$, the undetachable signature verification algorithm, is a polynomial time algorithm which takes the contract (or its hash value) and an undetachable signature z as input. The algorithm outputs either "Accept" or "Reject," simply 1 or 0.

To aid in further discussions, the frequently used symbols are listed as follows:

- 1^k : a security parameter, ($k \in \mathbb{N}$),
- P_{pub} : the public key,
- s : the master secret signing key,
- i, j : indices of the time period,

D_0 : the initial key,

D_j : the secret key for time period j ,

$D'_{i,j}$: a partial secret key,

$BID_S \parallel ID_S$: shop S 's bid information and identity,

$REQ_C \parallel ID_C$: customer C 's requirement and identity,

$f_{Signed,j}$: an implementation of the undetachable signing function for time period j ,

f : the auxiliary function of $f_{Signed,j}$,

Msg : message (usually a contract),

z : an undetachable signature,

q : a prime number,

G_1, G_2 : two cyclic groups whose orders are both q ,

P : a generator of G_1 ,

$e(\cdot, \cdot)$: a bilinear pairing from $G_1 \times G_1$ to G_2 .

3.2. Workflow of Using a KIUDS Scheme. First, a trusted authority, for example, a certification authority or a key distribution center, should publish all public parameters of the cryptosystem to all participants. The trusted authority then generates cryptographic keys for all participants by running the algorithm $KGen(1^k)$; and the private key (i.e., the master key and the initial key) of each participant is sent via a correspondingly secure communication channel. The customer stores the private signing key in a physically secure device. Day-to-day cryptographic operations are performed by an insecure device (e.g., the customer's PC) which periodically refreshes its key by interacting with the secure device using key update algorithms (Upd^* and Upd).

When a customer wants a mobile agent to do the shopping, the customer runs the $UndSigFunGen$ algorithm to prepare the mobile agent before it starts migrating. The mobile agent then begins migrating to search for shops that are willing to satisfy the customer's requirement.

FIGURE 6: Overall workflow of a KIUDS scheme.

Finally, anyone can check the validity of a contract by using the *UndVrfy* algorithm.

Figure 6 illustrates the workflow of using the algorithms in the proposed scheme.

3.3. Security Model and Security Notion. A KIUDS scheme differs from conventional digital signature schemes, as do its security model and security notion. However, the starting point of the discussion of the security model of KIUDS schemes is still the classical security model of conventional digital signature schemes.

There are four subclasses of known-message attacks against digital signature schemes [29]: the plain known-message attack, the generic chosen-message attack, the oriented chosen-message attack, and the adaptively chosen-message attack (ACMA). ACMA is the most dangerous scenario because the adversary is supposed to have such privilege, asking the signer to sign any message that he wants and then adapting his queries according to previous message-signature pairs.

The expected results of an attack are classified as follows [30]: (1) disclosing the secret key of the signer; (2) constructing an efficient algorithm which is able to sign any message;

and (3) providing a new message-signature pair. The third category is called existential forgery. In many cases this attack is not dangerous because the output message is likely to be meaningless. Nevertheless, a signature scheme which is not existentially unforgeable (and thus that admits existential forgeries) cannot be used to certify random-looking elements such as keys.

Besides the attack model of ACMA, the following three security threats must be included in the attack model of KIUDS schemes.

- (1) *Implementation Exposures.* A shop owner S is capable of obtaining the implementation of $f_{signed,j}$. This corresponds to an attack from malicious hosts. Note that an implementation exposure indicates ACMA because S can run the implementation of the signing algorithm.
- (2) *Key Exposures.* In some time periods, an attacker may break the defense system of a customer's computer and extract the corresponding signing key from the disk or memory.
- (3) *Adaptively Chosen-Restriction Attack.* This attack is an extension of the first threat. Consider the case where

a friend (F) of customer C asks C to purchase an item. The restriction of an undetachable signature is probably set up by F instead of C . Then, if F colludes with a shop owner, F is also capable of obtaining the implementation of $f_{\text{signed},j}$.

The first threat is easy to model by giving the adversary the description of the implementation of $f_{\text{signed},j}$.

To model the second threat, we give the adversary access to a key exposure oracle $O_{s,D_0}^{\text{Exposure}}(i)$ that performs the following operations on input i . The oracle first checks whether period i has been “activated”; if so, the oracle returns the value already stored for D_i . Otherwise, the key exposure oracle runs $d'_{0,i} \leftarrow \text{Upd}^*(0, i, s)$ followed by $D_i \leftarrow \text{Upd}(0, i, D_0, D'_{0,i})$, returns and stores the value D_i , and labels period i as “activated.”

To model the third threat, we give the adversary access to a chosen-restriction oracle $O_{s,D_0}^{\text{Chosen-restriction}}(i, \text{REQ}_C)$ that performs the following operations on input (i, REQ_C) . The

oracle first checks whether period i has been “activated”; if so, the oracle returns the output of $\text{UndSigFunGen}(D_j, \text{REQ}_C \parallel \text{ID}_C)$. Otherwise, the key exposure oracle runs $d'_{0,i} \leftarrow \text{Upd}^*(0, i, s)$ followed by $D_i \leftarrow \text{Upd}(0, i, D_0, D'_{0,i})$, stores the value D_i , labels period i as “activated,” and returns the output of $\text{UndSigFunGen}(D_j, \text{REQ}_C \parallel \text{ID}_C)$. Moreover, the first threat is also covered by the chosen-restriction oracle because the chosen-restriction oracle models a stronger adversary rather than a simple malicious eShop owner.

Note that storing the values of the secret signing keys for activated time periods is only necessary when the algorithm Upd^* is probabilistic; when Upd^* is deterministic, the oracles ($O_{s,D_0}^{\text{Exposure}}$ and $O_{s,D_0}^{\text{Chosen-restriction}}$) may simply run Upd^* “from scratch” whenever needed to answer a query.

Definition 1 ($((t, N)$ -KIUDS scheme). Let $\text{KIUDS} = (\text{KGen}, \text{Upd}^*, \text{Upd}, \text{UndSigFunGen}, \text{UndSig}, \text{UndVrfy})$ be a KIUDS scheme. For any adversary $\mathcal{A}$, the probability of a successful attack is defined as follows:

$$\text{Succ}_{\mathcal{A}}^{\text{KIUDS}}(k) = \Pr \left[\begin{array}{l} 1 \leftarrow \text{KIUDS.UndVrfy}(z, \text{Msg}) : \\ \langle P_{\text{pub}}, s, D_0 \rangle \leftarrow \text{KIUDS.KGen}(1^k) \\ (z, \text{Msg}) \leftarrow \mathcal{A}^{O_{s,D_0}^{\text{Exposure}}(\cdot), O_{s,D_0}^{\text{Chosen-restriction}}(\cdot, \cdot)}(P_{\text{pub}}) \\ \text{Msg} = \text{ID}_C \parallel \text{REQ}_C \parallel \text{ID}_S \parallel \text{BID}_S \\ O_{s,D_0}^{\text{Chosen-restriction}}(z, j, \text{REQ}_C) \text{ and } O_{s,D_0}^{\text{Exposure}}(j) \text{ has not been queried} \end{array} \right]. \quad (4)$$

We say that KIUDS is a (t, N) -KIUDS scheme if $\text{Succ}_{\mathcal{A}}^{\text{KIUDS}}(k)$ is negligible for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ who submits at most t queries to the key exposure oracle $O_{s,D_0}^{\text{Exposure}}$. Moreover, following the convention of key-insulated cryptography, we say KIUDS is perfectly key-insulated if t equals $N - 1$.

4. A Concrete Scheme

In this section, we propose a construction of a KIUDS scheme and provide the proofs of correctness and security. In addition, we present the results of the complexity analysis and performance testing. Note that the proposed scheme utilizes the signature scheme in [31] as a building block.

4.1. The Algorithms. Let G_1 be a cyclic group generated by a generator P , whose order is a prime number q , and let G_2 be a cyclic multiplicative group of the same order q . Suppose that discrete logarithm problems in both G_1 and G_2 are hard. Let $e : G_1 \times G_1 \rightarrow G_2$ be a pairing that satisfies the following three conditions.

Bilinear. Equations (5) and (6) together or (7) are as follows:

$$e(P_1 + P_2, Q) = e(P_1, Q) e(P_2, Q), \quad (5)$$

$$e(P, Q_1 + Q_2) = e(P, Q_1) e(P, Q_2), \quad (6)$$

$$e(aP, bQ) = e(P, Q)^{ab}, \quad (7)$$

where $\{a, b\} \subseteq \mathbb{Z}_q^*$.

Nondegenerate. There exist $P \in G_1$ and $Q \in G_1$ subject to

$$e(P, Q) \neq 1. \quad (8)$$

Computability. There is an efficient algorithm to compute $e(P, Q)$ for all $\{P, Q\} \subseteq G_1$.

We note that the Weil and Tate pairings associated with supersingular elliptic curves or abelian varieties can be modified to create such bilinear maps.

Suppose that G is an additive group. Four well-known mathematical problems are defined as follows.

- (1) **Discrete Logarithm Problem (DLP).** Given two group elements P and Q , and an integer n , (9) is satisfied whenever such an integer exists:

$$Q = nP. \quad (9)$$

- (2) **Decision Diffie-Hellman Problem (DDHP).** For $\{a, b, c\} \subseteq \mathbb{Z}_q^*$, given P, aP, bP, cP decide whether

$$c \equiv ab \pmod{q}. \quad (10)$$

ALGORITHM 1: $KGen(1^k)$. ALGORITHM 2: $Upd^*(s, j)$.

- (3) *Computational Diffie–Hellman Problem (CDHP)*. For $\{a, b\} \subseteq Z_q^*$, given P, aP, bP compute abP without any knowledge about the value of either a or b .

- (4) *Gap Diffie–Hellman Problem*.

We assume throughout this paper that CDHP and DLP are intractable, meaning that there is no polynomial time algorithm to solve CDHP or DLP with nonnegligible probability. When the DDHP is easy but the CDHP is hard on group G_1 , G_1 is called a gap Diffie–Hellman (GDH) group. Our scheme can be built on any GDH group. Further mathematical background can be found in [32, 33].

We now define some system parameters. Let P be a generator of G_1 . Suppose that the total number of time periods N is a shared public value. Two secure hash functions are given here: $H_1 : \{0, 1\}^* \rightarrow Z_q$ and $H_2 : \{0, 1\}^* \rightarrow G_1$. The implementations of these hash functions can be found in works such as [34–36].

The public parameters $\Omega = (G_1, G_2, e(\cdot, \cdot), q, P, H_1, H_2)$ should be published to all participants by the trusted authority. Moreover, a public function ValidTuple that outputs 1 when the input is a valid Diffie–Hellman tuple and outputs 0 otherwise is shared among all parties.

- (i) *The Key Generation Algorithm $KGen$* . It takes a security parameter 1^k where $k \in \mathbb{N}$ as input and returns the public key P_{pub} and the master secret key s as shown in Algorithm 1.

Because the total number of time periods N is already a system-wide public value and the initial key D_0 is not demanded in the proposed scheme, we omit them in the input and output, respectively.

- (ii) *The Device Key Update Algorithm Upd^** . In the proposed scheme, the device key update algorithm Upd^* updates the signing key directly as shown in Algorithm 2.

- (iii) *The User Key Update Algorithm Upd* . The user key update algorithm Upd can be defined as an identical transformation to fulfill the definition as shown in Algorithm 3.

ALGORITHM 3: $Upd(D'_j)$.

- (iv) *The Undetachable Signing Function Generation Algorithm $UndSigFunGen$* . First, the customer C generates the description of his/her requirement REQ_C . Next, the customer should run the $UndSigFunGen$ algorithm to prepare his/her mobile agent before the agent starts migrating. The process is illustrated in Figure 7 and the algorithm is presented as shown in Algorithm 4.

- (v) *The Undetachable Signing Algorithm $UndSign$* . If a shop owner is going to make a deal with the customer, he/she should generate bid information BID_S satisfying REQ_C and run the $UndSign$ algorithm on input $Msg = ID_C \parallel REQ_C \parallel ID_S \parallel BID_S$ to sign a contract (see Algorithm 5).

- (vi) *The Undetachable Signature Verification Algorithm $UndVrfy$* . Anyone can verify a contract by running the $UndVrfy$ algorithm that works as shown in Algorithm 6.

4.2. *Correctness of the Scheme*. First, we prove the correctness of the verification algorithm.

Proposition 2. *If $z = \langle \langle U, V, R' \rangle, Z' \rangle, j \rangle$ is generated by the undetachable signing algorithm $UndSign$ on $Msg = ID_C \parallel REQ_C \parallel ID_S \parallel BID_S$ and j , and BID_S satisfies REQ_C , then the verification algorithm $UndVrfy(z, Msg)$ outputs 1.*

Proof. Since P is a generator of G_1 , $\exists k (k \in Z_q^* \wedge kP = H_2(j))$.

Although it is difficult to calculate the value of k , the following equations hold:

$$P_{\text{pub}} = sP \quad (11)$$

$$\begin{aligned} U + H_1(REQ_C \parallel ID_C, U) H_2(j) \\ = rH_2(j) + H_1(REQ_C \parallel ID_C, U) H_2(j) \\ = (r + h)kP \end{aligned} \quad (12)$$

$$V = (r + h) D_j = (r + h) sH_2(j), \quad (13)$$

where

$$h = H_1(REQ_C \parallel ID_C, U). \quad (14)$$

Thus, according to (11), (12), and (13), we have

$$\begin{aligned} 1 = \text{ValidTuple}(P, P_{\text{pub}}, U \\ + H_1(REQ_C \parallel ID_C, U) H_2(j), V). \end{aligned} \quad (15)$$

FIGURE 7: Preparation of a mobile agent for migration.

```

Begin
   $r \xleftarrow{\$} Z_q^*$ 
   $U \leftarrow rH_2(j)$ 
   $h \leftarrow H_1(\text{REQ\_C} \parallel ID_C, U)$ 
   $V \leftarrow (r + h)D_j$ 
   $Q \leftarrow U + H_1(\text{REQ\_C} \parallel ID_C, U)H_2(j)$ 
  return a function  $f_{\text{signed},j}(x)$  as follows.
  Begin
     $t \xleftarrow{\$} Z_q$ 
     $R' \leftarrow tQ$ 
     $W' \leftarrow tV$ 
    return  $\langle U, V, R', xV + W' \rangle$ 
  End
End

```

ALGORITHM 4: $\text{UndSigFunGen}(D_j, \text{REQ_C} \parallel ID_C)$.

Moreover,

$$\begin{aligned}
 R' + \bar{x} \cdot \bar{Q} &= R' + H_1(\text{Msg}) \\
 &\quad \cdot (U + H_1(\text{REQ_C} \parallel ID_C, U) H_2(j)) \\
 &= t(U + H_1(\text{REQ_C} \parallel ID_C, U) H_2(j)) \\
 &\quad + H_1(\text{Msg}) \\
 &\quad \cdot (U + H_1(\text{REQ_C} \parallel ID_C, U) H_2(j)) \\
 &= (t + H_1(\text{Msg})) \cdot (U + H_1(\text{REQ_C} \parallel ID_C, U) H_2(j)) \\
 &= (t + H_1(\text{Msg})) \cdot (r + h) \cdot H_2(j) \\
 Z' &= H_1(\text{Msg}) V + W' = H_1(\text{Msg}) V + tV \\
 &= (H_1(\text{Msg}) + t)(r + h) D_j \\
 &= (H_1(\text{Msg}) + t)(r + h) s H_2(j).
 \end{aligned} \tag{16}$$

```

Begin
   $x \leftarrow H_1(\text{Msg})$ 
  return  $z = \langle f_{\text{signed},j}(x), j \rangle$ 
End

```

ALGORITHM 5: $\text{UndSign}(\text{Msg}, j)$.

Because of (11) and (16), we have

$$\text{ValidTuple}(P, P_{\text{pub}}, R' + \bar{x} \cdot \bar{Q}, Z') = 1. \tag{17}$$

Hence, the verification algorithm UndVrfy outputs 1. This completes the proof. $\square$

The next proposition indicates that the proposed scheme satisfies (2) in Section 2.3.

Proposition 3. Let $\text{Sig}_j(x) = \langle R', xD_j + W' \rangle$ and $f(x) = (r + H_1(\text{REQ_C} \parallel ID_C))x$; then $f_{\text{Signed},j}(x) = (\text{Sig}_j \circ f)(x)$.

Proof. Consider the following:

$$\begin{aligned}
 f_{\text{Signed}}(x) &= \langle R', xV + W' \rangle \\
 &= \langle R', x(r + H_1(\text{REQ_C} \parallel ID_C))D_j + W' \rangle \\
 &= \langle R', f(x)D_j + W' \rangle = (\text{Sig} \circ f)(x).
 \end{aligned} \tag{18}$$

$\square$

4.3. Security of the Scheme

Theorem 4. The proposed scheme is a perfectly KIUDS scheme.

Proof. First, we construct a security game which consists of two subgames (Game 1 and Game 2), as shown in Figure 8. In this security game, there are three players: $\mathcal{A}$,

```

Begin
   $ID_C \parallel REQ\_C \parallel BID\_S \leftarrow Msg$ 
   $\langle \langle U, V, R' \rangle, Z' \rangle, j \rangle \leftarrow z$ 
  If ( $BID\_S$  does NOT satisfy  $REQ\_C$ )
    return 0
  If ( $0 = \text{ValidTuple}(P, P_{\text{pub}}, U + H_1(REQ\_C \parallel ID_C, U)H_2(j), V)$ )
    return 0
   $\bar{Q} \leftarrow U + H_1(REQ\_C \parallel ID_C, U)H_2(j)$ 
   $\bar{x} \leftarrow H_1(Msg)$ 
  If ( $1 = \text{ValidTuple}(P, P_{\text{pub}}, R' + \bar{x} \cdot \bar{Q}, Z')$ )
    return 1
  Else
    return 0
End

```

ALGORITHM 6: $UndVrfy(z, Msg)$.

```

Begin
  return  $CCS.Extract(i)$ 
End

```

ALGORITHM 7: $\mathcal{F}.OraExpo(i)$.

$\mathcal{F}$, and a simulator of the signature scheme in [31] called $CCS(Sim)$. The game proceeds in the random oracle model (ROM). $\mathcal{A}$ is a PPT adversary against the proposed key-insulated undetachable signature scheme. The message is in the form of $Msg = (m_1, m_2)$. The forger $\mathcal{F}$ plays between $\mathcal{A}$ and $CCS(Sim)$. $\mathcal{A}$ is capable of making queries $O_{s,D_0}^{\text{Exposure}}(\cdot)$, $O_{s,D_0}^{\text{Chosen-restriction}}(\cdot, \cdot)$, $O^{H_1}(\cdot)$, and $O^{H_2}(\cdot)$. $\mathcal{F}$ is responsible for answering these queries with the help of $CCS(Sim)$. $CCS(Sim)$ should answer queries $CCS.Extract(\cdot)$, $CCS.Sign(\cdot, \cdot)$, $CCS.H_1(\cdot)$, and $CCS.H_2(\cdot)$, from $\mathcal{F}$. Note that the input parameter of $CCS.KGen$ is the unitary form $l \in \mathbb{N}$, where l is the order of P .

Algorithms 7 and 8 are used by the forger $\mathcal{F}$ to answer queries from the adversary $\mathcal{A}$ in the security game.

At the end of the security game, the algorithm $\mathcal{F}.Forge$ is used to forge a signature of $CCS(Sim)$ from the output of $\mathcal{A}$. The algorithm $\mathcal{F}.Forge$ proceeds as shown in Algorithm 9.

Clearly, in the security game that is illustrated in Figure 8, we have

$$\begin{aligned}
 Succ_{\mathcal{A}}^{KIUS}(k) &= \Pr[1 \leftarrow KIUS.UndVrfy(z, Msg)] \\
 &= 1 - \Pr[\perp \leftarrow \mathcal{F}.Forge(z, Msg)].
 \end{aligned} \quad (19)$$

Suppose that the adversary $\mathcal{A}$ can win Game 1 that queries $O_{s,D_0}^{\text{Exposure}}(\cdot)$, $O_{s,D_0}^{\text{Chosen-restriction}}(\cdot, \cdot)$, $O^{H_1}(\cdot)$, and $O^{H_2}(\cdot)$ at most q_E , q_C , q_{O_1} , and q_{O_2} times, respectively, and has a running time of T_0 and an advantage ϵ_0 . Then, $\mathcal{F}$ can win Game 2 with queries $CCS.Extract(\cdot)$, $CCS.Sign(\cdot, \cdot)$, $H_1(\cdot)$, and $H_2(\cdot)$ at most q_E , q_C , $q_{O_1} + q_C + 3$, and $q_{O_2} + q_E + q_C + 2$ times, respectively, and has a running time of $T = T_0 + c_F$ and an advantage $\epsilon_0 = Succ_{\mathcal{A}}^{KIUS}(k)$. Recall that the simulator $CCS(Sim)$ simulates

the signature scheme in [31], and therefore Game 2 depicts attacks against the signature scheme in Section 3.1 of [31] in an equivalence form. Based on Lemma 4 in [30] (the Forking Lemma), Theorem 3 in [31] stated that “if there is an algorithm for an adaptively chosen message and ID attack to our scheme which queries H_1 , H_2 and Sign and Extract at most q_{H_1} , q_{H_2} , q_S , and q_E times, respectively, and has running time t_0 and advantage

$$\epsilon_0 \geq \frac{10(q_S + 1)(q_S + q_{H_1})q_{H_2}}{l - 1}, \quad (20)$$

then the CDHP can be solved with probability $\geq 1/9$ and within running time

$$\frac{23 \times q_{H_1} \times q_{H_2} \times t_0}{\epsilon_0(1 - (1/l))}. \quad (21)$$

By substituting $q_{H_1} = q_{O_1} + q_C + 3$, $q_{H_2} = q_{O_2} + q_E + q_C + 2$, $q_S = q_C$, $q_E = q_E$, and $t_0 = T_0 + c_F$ into (20) and (21), we have that if

$$\begin{aligned}
 \epsilon_0 &\geq \frac{10(q_C + 1)(2q_C + q_{O_1} + 3)(q_{O_2} + q_E + q_C + 2)}{l - 1},
 \end{aligned} \quad (22)$$

then the CDHP can be solved with probability no less than $1/9$ and within running time

$$\frac{23(q_{O_1} + q_C + 3)(q_{O_2} + q_E + q_C + 2)(T_0 + c_F)}{\epsilon_0(1 - 1/l)}. \quad (23)$$

According to the assumption that CDHP is hard, ϵ_0 is negligible with respect to the value of l , and so is $Succ_{\mathcal{A}}^{KIUS}(k)$. This completes the proof. $\square$

4.4. Complexity Analysis. The mathematical operations used in this scheme are mainly scalar addition (SA; $\langle Z_q^*, Z_q^* \rangle \rightarrow Z_q^*$), random selection (RS; $\xleftarrow{\$} Z_q^*$), scalar multiplication

FIGURE 8: The security game.

```

Begin
   $\langle U, V \rangle \leftarrow CCS.Sign(i, REQ\_C \parallel ID_C)$ 
   $Q \leftarrow U + CCS.H_1(REQ\_C \parallel ID_C, U) \parallel CCS.H_2(j)$ 
  return a function  $f_{signed,j}(x)$  as follows.
  Begin
     $t \xleftarrow{\$} Z_q$ 
     $R' \leftarrow tQ$ 
     $W' \leftarrow tV$ 
    return  $\langle U, V, R', xV + W' \rangle$ 
  End
End

```

ALGORITHM 8: $\mathcal{F}.OraCR(i, REQ_C)$.

(SM; $\langle Z_q^*, G_1 \rangle \rightarrow G_1$), point addition (PA; $\langle G_1, G_1 \rangle \rightarrow G_1$), bilinear map (BM; $e, G_1 \times G_1 \rightarrow G_2$), and the two hash functions H_1 and H_2 . All of these operations are polynomial-bounded and can be computed efficiently.

Let q be the size of an element in Z_q^* and let S_{G_1} be the size of an element in G_1 . Let $|res|$ be the length of $REQ_C \parallel ID_C$, $|msg|$ the length of the Msg , and $|impl|$ the size of the implementation of $f_{signed,j}$ without U, V , and Q . In Table 1, we show the numbers of operations needed for the algorithms proposed in Section 4.1.

5. Theoretical Comparison with Related Work

Since the first implementation of undetachable digital signatures was proposed by Kotzanikolaou et al. [3], several concrete constructions of undetachable digital signature schemes have been proposed from 2001 to 2015 [4, 5, 7–10]. Compared with these prior undetachable digital signature schemes, the most significant feature of our proposed scheme lies in its

key-insulation property. Different from most of prior studies, we have formally proved the security of the scheme. Moreover, the security of related schemes depends on a variety of diverse assumptions on the computational infeasibility of mathematical problems. Comprehensive comparisons in terms of the above three factors are summarized in Table 2.

The forward-secure undetachable digital signature (FSUDS) scheme in prior work [10] is currently the latest published study on undetachable signatures. In terms of the security, the “perfectly key-insulated” property implies the “forward-secure” property, but the inverse is not valid. Therefore, the proposed KIUDS scheme is stronger than the FSUDS scheme.

Compared with other key-insulated signature schemes, the proposed scheme is undetachable. The private key of the original signer (i.e., the customer) will hardly be compromised, even in a white-box attack context (e.g., a malicious host). For attackers, the misuse of signing functions carried with mobile agents is also infeasible even in white-box attack contexts, because the related restriction is combined with the “encrypted” signing function.

We further present the comparison in the field of software obfuscation. Obfuscation is a process that transforms a program into an unintelligible one without changing the original functionalities. If the signing algorithm has been obfuscated, the attacker cannot extract the signing key from the obfuscated implementation. However, it is still an open research problem that whether there exists an obfuscator for the signing algorithm in a digital signature scheme. Some obfuscators for specialized encrypted signatures have been developed since Hada’s work was reported in Euro-Crypt 2010 [37]. For example, an obfuscator for encrypted verifiably encrypted signatures [38] and an obfuscator for encrypted group signatures [39] were presented in recent years. However, obfuscated implementation of the generation algorithm of encrypted signatures cannot prevent misusing

```

Begin
 $ID_C \parallel REQ\_C \parallel ID_S \parallel BID\_S \leftarrow \text{Msg}$ 
 $\langle \langle U, V, R' \rangle, Z' \rangle, j \rangle \leftarrow z$ 
If ( $BID\_S$  does NOT satisfy  $REQ\_C$ )
    retrun  $\perp$ 
If ( $0 = \text{ValidTuple}(P, P_{\text{pub}}, U + \text{CCS}.H_1(REQ\_C \parallel ID_C, U) \text{CCS}.H_2(j), V)$ )
    retrun  $\perp$ 
 $\bar{Q} \leftarrow U + \text{CCS}.H_1(REQ\_C \parallel ID_C, U) \text{CCS}.H_2(j)$ 
 $\bar{x} \leftarrow \text{CCS}.H_1(\text{Msg})$ 
If ( $1 = \text{ValidTuple}(P, P_{\text{pub}}, R' + \bar{x} \cdot \bar{Q}, Z')$ )
    return  $\langle j, REQ\_C \parallel ID_C, (U, V) \rangle$ 
Else
    return  $\perp$ 
End

```

ALGORITHM 9: $\mathcal{F}.\text{Forge}(z, \text{Msg})$.

TABLE 1: The number of operations and size of the input/output of the proposed algorithms.

Algorithm	Complexity	Size of input	Size of output
1	RS + SM	N/A	$q + S_{G1}$
2	H_2 + SM	$Zq + \log_2 N$	S_{G1}
3	N/A	N/A	N/A
4	$H_1 + H_2 + \text{SA} + \text{RS} + 3\text{SM} + \text{PA}$	$S_{G1} + \text{res} $	$3S_{G1} + \text{impl} $
5	$H_1 + \text{RS} + \text{SM} + \text{PA}$	$ \text{msg} + \log_2 N$	$4S_{G1} + \log_2 N$
6	$3H_1 + 2H_2 + 3\text{SM} + 3\text{PA} + 4\text{BM}$	$4S_{G1} + \log_2 N + \text{msg} $	1

attacks, (i.e., an attacker can call the algorithm to generate a signature). Therefore, compared with obfuscation approaches for encrypted signatures, the proposed scheme is capable of providing further protection against misuse of the signing algorithm.

6. Experimental Results and Comparison

6.1. Performance Testing. We have implemented the algorithms in Java. Java has been used instead of C/C++ because many mobile agent platforms are developed in Java, although C/C++ is known to be more efficient. We used an open source Java Pairing-Based Cryptography Library (JPBC) [40] in our implementation. The configurations of the testing platforms are listed in Table 3, and the experimental results are shown in Figures 9 and 10.

When the computing platform is a PC, we focus on the speed of the algorithms. In Figure 9, we show the speed of the algorithms on two different portable computers in single-thread mode. When the computing platform is a server, the most important index of performance is the number of transactions that can be processed in a short time span (e.g., a second). Hence, Figure 10 shows the number of operations of each algorithm on a PC server in multithread mode. The experimental results indicate that the algorithms in the proposed scheme are quite efficient.

6.2. Experimental Comparison with Related Work. With regard to the standard undetachable digital signature scheme

FIGURE 9: Running time (ms) on PC1 and PC2 for each algorithm in the proposed scheme.

based on bilinear pairings [8] and the FSUDS scheme [10], a group of experimental comparisons on computational costs has been performed on the server described in Table 3. The scheme in [8] has been selected for comparison because it is based on the same mathematical structure as the proposed scheme is. As the standard scheme does not include a key update algorithm, we use “0” to fill the data form. In Figure 11,

TABLE 2: Comparisons of undetachable digital signature schemes.

Scheme	Extra security feature	Formal proof of security	Computational infeasible problem
[3]	No	No	Factorization of big integer
[4]	Bilateral security	No	Factorization of big integer
[5]	No	No	q -Strong Diffie–Hellman problem
[6]	Threshold security	No	Factorization of big integer
[7]	No	No	Discrete logarithm on elliptic curves
[8]	No	No	Computational co-Diffie–Hellman problem
[9]	Threshold security	No	Discrete logarithm on conic curves
[10]	Forward-secure	Yes	Computational co-Diffie–Hellman problem
This work	Key-insulated	Yes	Computational Diffie–Hellman problem

TABLE 3: Configurations of testing platforms.

Platform	CPU	RAM	OS	JDK
PC1	Intel Core i5-4200M @ 2.50 GHz	8 GB	Win7	7.0
PC2	Intel Core i5-M480 @ 2.67 Hz	4 GB	Win8	7.0
Server	Intel E5 2.5 G 6Cores * 2	96 GB	WinServ2008	7.0

FIGURE 10: Number of operations per second on the server for each algorithm in the proposed scheme.

the result shows that the algorithms of the proposed scheme execute faster than those of the FSUDS scheme, except for the *UndSigFunGen* algorithm. In particular, the key generation algorithm *KGen* of the proposed scheme executes approximately 50 times faster than that of the FSUDS scheme. Furthermore, the performance of the proposed KIUDS scheme is similar to that of the standard scheme (denoted as *Std* in the figure), but the KIUDS scheme offers higher-level security than the standard scheme does.

FIGURE 11: Comparison on performance of algorithms between the KIUDS and the FSUDS.

The sizes of the input/output of the three algorithms (KI, FS, and *Std*) have also been compared. The result is obtained by analyzing and testing the software implementations of the schemes. As shown in Figure 12, most of the algorithms of the KIUDS scheme have smaller inputs and outputs than those of the FSUDS scheme. Furthermore, although the sizes of the inputs of *UndSigFunGen* and *UndVrfy* and the output of *UndSigFunGen* of the KIUDS scheme are larger than those of the standard scheme, the increased cost is acceptable in consideration of acquiring extra key-insulated security.

FIGURE 12: Comparisons on sizes of inputs/outputs of algorithms (in byte).

6.3. An Example of KIUDS Implementation. To further demonstrate the feasibility of the proposed scheme in practice, we have implemented a KIUDS solution for purchasing/selling books with mobile agents as a concrete example. The implementation is based on the *Java Agent DEvelopment Framework* (JADE) [41], which is an open source software framework that supports implementation of multiagent systems through a middleware complying with the FIPA specifications [42] and a set of tools supporting debugging and deployment phases. A JADE-based system can be distributed across various hosts while the configuration can be controlled via remote connections. The configuration can even be modified at run-time by moving agents from one host to another as necessary.

As an example, we created three instances of e-bookstores that sell books and a buyer agent that represents a customer who purchases books. Each e-bookstore has a number of books for sale with specified prices. The buyer agent aims

to purchase the cheapest copy of the book titled “*Thinking in Java*” by visiting the three e-bookstores in an autonomous manner and comparing the prices retrieved. When the buyer agent purchases the book, a KIUDS is generated to sign the contract. Figure 13 illustrates the process of the book purchasing example.

The implementation has demonstrated the feasibility of the KIUDS scheme and solution, which serves as a proof-of-concept that complements the theoretical analysis and comparisons in previous sections. We also plan to further apply the KIUDS solution in more domains and identify more research issues that are worth in-depth exploration and study in future work.

7. Conclusion

In this paper, we introduce the definition and provide the security model and security notion of KIUDS schemes.

(a) Initialize the agent system

(b) Create an eShop

(c) Add books for selling

(d) Create a buyer agent

(e) Sign and buy

The buyer agent finds the cheapest book and generates a signature as follows (console output):

```
KeyUpdating...
UndSignFunGenerating...
req:less than $100
UndSigning...
(0,685671239497749960534507228070625005436134772098610227180274072133985756490291539587)
Trying to buy Thinking In Java
Found the following seller agents:
eShop01
eShop02
eShop03
Thinking In Java sold to agent customer
Thinking In Java successfully purchased from agent eShop01
Price = 70
Buyer-agent customer terminating.
```

FIGURE 13: An example of buying and selling books.

KIUDS schemes are capable of protecting the digital signature functionality against attacks from remote malicious hosts and controlling the security risk of signing key leakage on the original signer's host at the same time. Practically, a concrete KIUDS scheme with provable security is proposed for secure mobile agents in electronic commerce. The scheme is constructed on bilinear pairings and the security of the scheme is proven to rely on the hardness of the computational Diffie–Hellman problem on gap Diffie–Hellman groups which is infeasible to solve in practice. Our experimental results have demonstrated that the proposed scheme is efficient.

Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

Acknowledgments

This research has been supported by the National Natural Science Foundation of China (no. 61202382), the Fundamental Research Funds for the Central Universities, and the Scientific Research Foundation for the Returned Overseas Chinese Scholars.

References

- [1] R. Singh and M. Dave, "Antecedence graph approach to check-pointing for fault tolerance in mobile agent systems," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 247–258, 2013.
- [2] F. Raji and B. T. Ladani, "Anonymity and security for autonomous mobile agents," *IET Information Security*, vol. 4, no. 4, pp. 397–410, 2010.
- [3] P. Kotzanikolaou, M. Burmester, and V. Chrissikopoulos, "Secure transactions with mobile agents in hostile environments," in *Information Security and Privacy: 5th Australasian Conference, ACISP 2000, Brisbane, Australia, July 10–12, 2000. Proceedings*, vol. 1841 of *Lecture Notes in Computer Science*, pp. 289–297, Springer, Berlin, Germany, 2000.
- [4] B. Lee, H. Kim, and K. Kim, "Secure mobile agent using strong non-designated proxy signature," in *Information Security and Privacy, Proceedings*, pp. 474–486, Springer, Berlin, Germany, 2001.
- [5] S. Han, E. Chang, and T. Dillon, "Secure e-transactions using mobile agents with agent broker," in *Proceedings of the International Conference on Services Systems and Services Management (ICSSSM '05)*, vol. 2, pp. 849–855, IEEE, June 2005.
- [6] N. Borselius, C. J. Mitchell, and A. Wilson, "Undetachable threshold signatures," in *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pp. 239–244, Cirencester, UK, December 2001.
- [7] Y. Shi, L. Cao, and X. Wang, "A security scheme of electronic commerce for mobile agents uses undetachable digital signatures," in *Proceedings of the 3rd International Conference on Information Security (InfoSecu '04)*, pp. 242–243, November 2004.
- [8] Y. Shi, X. P. Wang, L. M. Cao, and J. X. Ren, "Secure mobile agents in electronic commerce by using undetachable signatures from pairings," in *Shaping Business Strategy in a Networked World, Vols 1 and 2, Proceedings*, pp. 1038–1043, 2004.
- [9] Y. Shi and G. Xiong, "An undetachable threshold digital signature scheme based on conic curves," *Applied Mathematics and Information Sciences*, vol. 7, no. 2, pp. 823–828, 2013.
- [10] Y. Shi, Q. Zhao, and Q. Liu, "Secure mobile agents in eCommerce with forward-secure undetachable digital signatures," *ETRI Journal*, vol. 37, no. 3, pp. 573–583, 2015.
- [11] D. Milojicic, M. Breugst, I. Busse et al., "MASIF: the OMG mobile agent system interoperability facility," *Personal Technologies*, vol. 2, no. 2, pp. 117–129, 1998.
- [12] A. Corradi, M. Cremonini, R. Montanari, and C. Stefanelli, "Mobile agents integrity for electronic commerce applications," *Information Systems*, vol. 24, no. 6, pp. 519–533, 1999.
- [13] A. Corradi, R. Montanari, and C. Stefanelli, "Security of mobile agents on the Internet," *Internet Research*, vol. 11, no. 1, pp. 84–95, 2001.
- [14] F. Huseynov, S. Y. Huseynov, and S. O zkan, "The influence of knowledge-based e-commerce product recommender agents on online consumer decision-making," *Information Development*, vol. 32, no. 1, pp. 81–90, 2015.
- [15] M. Cao, X. Luo, X. Luo, and X. Dai, "Automated negotiation for e-commerce decision making: a goal deliberated agent architecture for multi-strategy selection," *Decision Support Systems*, vol. 73, pp. 1–14, 2015.
- [16] B. M. Balachandran and M. Mohammadian, "Development of a fuzzy-based multi-agent system for E-commerce Settings," in *Knowledge-Based and Intelligent Information & Engineering Systems 19th Annual Conference, KES-2015*, L. Ding, C. Pang, L. M. Kew, L. C. Jain, and R. J. Howlett, Eds., *Procedia Computer Science*, pp. 593–602, Elsevier Science, Amsterdam, The Netherlands, 2015.
- [17] A. J. C. Trappey, C. V. Trappey, and F. T. L. Lin, "Automated silicon intellectual property trade using mobile agent technology," *Robotics and Computer-Integrated Manufacturing*, vol. 22, no. 3, pp. 189–202, 2006.
- [18] Y.-F. Chung, Y.-T. Chen, T.-L. Chen, and T.-S. Chen, "An agent-based English auction protocol using Elliptic Curve Cryptosystem for mobile commerce," *Expert Systems with Applications*, vol. 38, no. 8, pp. 9900–9907, 2011.
- [19] G. Wang, T. N. Wong, and X. H. Wang, "A hybrid multi-agent negotiation protocol supporting agent mobility in virtual enterprises," *Information Sciences*, vol. 282, pp. 1–14, 2014.
- [20] A. Aloui, O. Zerdoumi, and O. Kazar, "Architecture for mobile business based on mobile agent," in *Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS '12)*, pp. 954–958, May 2012.
- [21] B. Wyseur, *White-Box Cryptography*, Katholieke Universiteit, Heverlee, Belgium, 2009.
- [22] S. Chow, P. Eisen, H. Johnson, and P. C. Van Oorschot, "A white-box DES implementation for DRM applications," in *Digital Rights Management*, pp. 1–15, Springer, Berlin, Germany, 2003.
- [23] O. Esparza, J. L. Muñoz, J. Tomàs-Buliart, and M. Soriano, "An infrastructure for detecting and punishing malicious hosts using mobile agent watermarking," *Wireless Communications & Mobile Computing*, vol. 11, no. 11, pp. 1446–1462, 2011.
- [24] F. Hohl, "Time limited blackbox security: protecting mobile agents from malicious hosts," in *Mobile Agents and Security*, vol. 1419 of *Lecture Notes in Computer Science*, pp. 92–113, Springer, Berlin, Germany, 1998.
- [25] T. Sander and C. Tschudin, "Protecting mobile agents against malicious hosts," in *Mobile Agents and Security*, pp. 44–60, Springer, Berlin, Germany, 1998.
- [26] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *Journal of Cryptology*, vol. 17, no. 4, pp. 297–319, 2004.
- [27] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Key-insulated public key cryptosystems," in *Advances in Cryptology—EUROCRYPT 2002*, pp. 65–82, Springer, Berlin, Germany, 2002.
- [28] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Strong key-insulated signature schemes," in *Public Key Cryptography*, vol. 2567 of *Lecture Notes in Computer Science*, pp. 130–144, Springer, Berlin, Germany, 2002.
- [29] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *Advances in Cryptology—EUROCRYPT '96: International Conference on the Theory and Application of Cryptographic Techniques Saragossa, Spain, May 12–16, 1996 Proceedings*, vol. 1070 of *Lecture Notes in Computer Science*, pp. 387–398, Springer, Berlin, Germany, 1996.
- [30] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, vol. 13, no. 3, pp. 361–396, 2000.
- [31] J. C. Choon and J. H. Cheon, "An identity-based signature from gap Diffie-Hellman groups," in *Public Key Cryptography—PKC 2003*, pp. 18–30, Springer, Berlin, Germany, 2002.
- [32] D. M. Freeman and T. Satoh, "Constructing pairing-friendly hyperelliptic curves using Weil restriction," *Journal of Number Theory*, vol. 131, no. 5, pp. 959–983, 2011.

- [33] T. Kim, S. Kim, and J. H. Cheon, "On the final exponentiation in tate pairing computations," *IEEE Transactions on Information Theory*, vol. 59, no. 6, pp. 4033–4041, 2013.
- [34] Y. Kawahara, T. Kobayashi, G. Takahashi, and T. Takagi, "Faster maptopoint on supersingular elliptic curves in characteristic 3," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 94, no. 1, pp. 150–155, 2011.
- [35] G. Barthe, B. Grégoire, S. Heraud, F. Olmedo, and S. Zanella-Béguelin, "Verified indifferentiable hashing into elliptic curves," *Journal of Computer Security*, vol. 21, no. 6, pp. 881–917, 2013.
- [36] R. R. Farashahi, P.-A. Fouque, I. E. Shparlinski, M. Tibouchi, and J. F. Voloch, "Indifferentiable deterministic hashing to elliptic and hyperelliptic curves," *Mathematics of Computation*, vol. 82, no. 281, pp. 491–512, 2013.
- [37] S. Hada, "Secure obfuscation for encrypted signatures," in *Advances in Cryptology—EUROCRYPT 2010*, vol. 6110 of *Lecture Notes in Computer Science*, pp. 92–112, Springer, 2010.
- [38] R. Nishimaki and K. Xagawa, "Verifiably encrypted signatures with short keys based on the decisional linear problem and obfuscation for encrypted VES," *Designs, Codes and Cryptography*, vol. 77, no. 1, pp. 61–98, 2015.
- [39] Y. Shi, Q. Zhao, H. Fan, and Q. Liu, "Secure obfuscation for encrypted group signatures," *PLoS ONE*, vol. 10, no. 7, Article ID e0131550, 2015.
- [40] A. De Caro and V. Iovino, "jPBC: java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications (ISCC '11)*, pp. 850–855, Kerkira, Greece, July 2011.
- [41] JADE, "JAVA Agent DEvelopment Framework," <http://jade.tilab.com/>.
- [42] A. C. L. Fipa, *Message Structure Specification*, Foundation for Intelligent Physical Agents, 2002.