A pairing-free key-insulated certificate-based signature scheme with provable security

Hu Xiong[1], Shikun Wu[1], Ji Geng[1], Emmanuel Ahene[1], Songyang Wu[2], Zhiguang Qin[1],
[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China
Chengdu, Sichuan 610054 - CHINA
[e-mail: xionghu.uestc@gmail.com]
[2] Third Research Institute, Ministry of Public Security
Shanghai, 201204 – CHINA
[e-mail : wusongyang@stars.org.cn]
*Corresponding author: Songyang Wu

Received October 2, 2014; revised December 16, 2014; accepted January 21, 2015; published March 31, 2015

Abstract

Certificate-based signature (CBS) combines the advantages of both public key-based signature and identity-based signature, while saving from the disadvantages of drawbacks in both PKS and IBS. The insecure deployment of CBS under the hostile circumstances usually causes the exposure of signing key to be inescapable. To resist the threat of key leakage, we present a pairing-free key insulated CBS scheme by incorporating the idea of key insulated mechanism and CBS. Our scheme eliminates the costly pairing operations and as a matter of fact outperforms the existing key insulated CBS schemes. It is more suitable for low-power devices. Furthermore, the unforgeability of our scheme has been formally proven to rest on the discrete logarithm assumption in the random oracle model.

Keywords: Key-insulated, certificate-based signature, pairing-free, random oracle model

This work is financially supported by National Natural Science Foundation of China under Grant Nos. 61003230 and 61370026, the Fundamental Research Funds for the Central Universities under Grant No. ZYGX2013J073, and the Applied Basic Research Program of Sichuan Province under Grant No. 2014JY0041.

http://dx.doi.org/10.3837/tiis.2015.03.023 **ISSN : 1976-7277**

1. Introduction

The digital signature, as a counterpart to ink signature in the electronic world, can be used to authenticate a digital message or document. A valid signature provides an enough evidence for a receiver to believe that the message or document was indeed issued by the claimed signer [1]. The digital signature was originally introduced in traditional asymmetric cryptography. In this environment, a certificate generated by the trusted certificate authority (CA) is needed to establish the connection between the public key (usually an unreadable random string) and the identity of the signer [2-4]. The extremely expensive overhead of certificate generation, distribution and revocation impedes digital signature from wide adoption in the government affair, financial transaction and software distribution. To lower the maintenance costs of certificates in traditional public key cryptography (PKC), Shamir [5] creatively put forward the notion of Identity-based (ID-based) cryptosystem. In ID-based cryptosystem, the certificates in the traditional PKC is no longer demanded due to the fact that the public key of the signer can be effortlessly derived from signer's known identity information (phone number, email address and so on). For this reason, digital signature is extensively studied in ID-based cryptosystems [6-8]. However, the merits of ID-based cryptosystem are associated with the notorious key escrow problem. Specifically, the private key of the signer will be generated by a fully trusted private key generator (PKG) according to his/her identity and the PKG can impersonate all of the signers in the system without being detected and punished.

By incorporating the basic idea of both ID-based cryptosystem and conventional PKC simultaneously, certificate-based cryptography (CBC) [9] has been suggested to save from their disadvantages simultaneously. In CBC, the public and private key pair of the user is generated by the user himself/herself and a corresponding certificate of his/her public key is requested from the CA. On one hand, the certificate can guarantee the connection between the user and his/her public key as in traditional PKC. On the other hand, this certificate in CBC acts as part of the user's private key such that the cryptographic operation such as signing or decrypting can only be performed by using user's private key and certificate together. Featured with implicit certification, CBC revokes the need of third-party query in traditional PKC, and thus simplifies the complex certificate management. Also, CBC does not inherit key escrow problem from ID-based cryptosystem because the private key is generated and kept by the user himself/herself. Certificate-based signature (CBS) [10-14], the combination of digital signature and CBC, has been naturally investigated.

It could be very disastrous if the private key is exposed in case CBS is insecurely deployed in a hostile environment. In the light of this, the key-insulated CBS [15-16] has been proposed. In the key-insulated mechanism, the life cycle of user's private key is partitioned into different time slices. The private key of the user will evolve from one time period to the next with the aid of a physically secure device (helper) while the public key of the user will be fixed during the whole lifetime. As such, the corruption of private key in some time periods will not affect the security of private key in the other time periods. The first concrete key-insulated CBS scheme [17] has been proposed recently based on the costly bilinear pairing operation [18, 19]. Hence, the construction of pairing-free key insulated CBS along with formal security proof is still an interesting and challenging problem.

In this paper, a pairing-free key-insulated CBS scheme has been presented. Our scheme eliminates the costly pairing operations and as a matter of fact outperforms the existing key insulated CBS schemes. It is more suitable for low-power devices. Furthermore, the unforgeability of our scheme is formally proven under the discrete logarithm assumption in the random oracle model [20].

2. Preliminaries

In this section, we review the formal definition of CBS scheme and the building blocks of our scheme.

2.1 Mathematical Assumption

Definition 1. (Discrete Logarithm (DL) Assumption) Given a group $\mathbb{G}$ with prime order p along with a generator P, the DL problem refers to compute $x \in \mathbb{Z}_p^*$ with the presence of a random element $Q \in \mathbb{G}$ such that $Q = xP$. The DL assumption means that the DL problem in $\mathbb{G}$ cannot be solved by an adversary $\mathcal{A}$ with a non-negligible probability.

2.2 Modeling key-insulated CBS

According to [17], a key insulated CBS scheme involves a CA, and a signer equipped with a helper, and consists of following eight algorithms: **Setup**, **UserKeyGen**, **CertGen**, **SetInitialKey**, **UpdH**, **UpdS**, **Sign**, and **Verify**.

- **Setup**: This algorithm is carried out by the CA to generate the system public parameters *params* and master private key msk by taking a security parameter 1^k as input.

- **UserKeyGen**: This algorithm is performed by the user associated with the identity ID itself to generate the public and private key pair (pk_{ID}, sk_{ID}) by taking the system parameters *params* as input.

- **CertGen**: This algorithm is executed by the CA to generate the certificate $Cert_{ID}$ for the user associated with the identity ID and the public key pk_{ID} by taking the master secret key msk, the system public parameters *params* and $\{ID, pk_{ID}\}$ as input.

- **SetInitialKey**: This algorithm is run by the user to generate the initial temporary secret key $TSK_{ID,0}$ and helper key HK_{ID} by taking the system public parameters *params*, the identity ID, the public key pk_{ID} and the certificate $Cert_{ID}$ of the user as input. After that, the helper key HK_{ID} will be stored in a exclusive helper and the initial temporary secret key $TSK_{ID,0}$ will be kept by the user itself.

- **UpdH**: To assist the user to update the temporary secret key from period t to t', an update key $UK_{ID,t,t'}$ will be generated by the helper by taking the system public parameters *params*, the identity ID, the helper key HK_{ID}, and (t, t') as input.

- **UpdS**: This algorithm will be executed by the user to update the temporary secret key $TSK_{ID,t}$ corresponding to the time period t to the temporary secret key $TSK_{ID,t'}$ corresponding to the time period t' by taking the system public parameters *params*, the identity ID of the user, (t, t'), and an update key $UK_{ID,t,t'}$ as input.

- **Sign**: This algorithm is performed by the signer associated with the identity ID and the public key pk_{ID} to generate the signature (t, σ) by taking the temporary secret key $TSK_{ID,t}$, the certificate $Cert_{ID}$, the system public parameters *params*, a message m, and a period index t as input.

- **Verify**: This algorithm can be executed by anyone to verify the validity of the signature pair (t, σ) by taking a message m, the system public parameters *params*, the identity ID and the public key pk_{ID} of the user as input.

2.3 Security definitions

Motivated by the security models for key insulated signature in traditional PKI [15, 16], ID-based cryptosystems [21, 22], and certificate-based signature [12-14], Li *et al.* [17] formalized the security model for key insulated CBS scheme as follows: Two types of adversaries, type I adversary $\mathcal{A}_1$ and type II adversary $\mathcal{A}_2$, along with two security games is considered to capture the attacks launched by the outside attacker and the malicious CA respectively. $\mathcal{A}_1$ is an adversary who acts as an outsider and wants to forge a valid signature with the ability to replace the public key. The restriction for $\mathcal{A}_1$ is that the master secret key cannot be accessed and the certificate of the replaced public key cannot be obtained by this kind of adversary. $\mathcal{A}_2$ models the malicious CA as an adversary who wants to forge a valid signature by using the master secret key. The restriction for $\mathcal{A}_2$ is that this kind of adversary cannot replace the public key of the target user. The security games against $\mathcal{A}_1$ and $\mathcal{A}_2$ are described as follows.

Game I (for the adversary $\mathcal{A}_1$):

Setup: In this phase, the **Setup** algorithm is run by the challenger $\mathcal{C}$. After that, *params* will be sent to $\mathcal{A}_1$ and msk will be kept by $\mathcal{C}$ itself secretly.

Queries: $\mathcal{C}$ will answer the adaptive queries issued by $\mathcal{A}_1$ as follows.

- **UserKeyGen** oracle: $\mathcal{C}$ maintains an initially empty list $\mathcal{L}$ with the item $(ID_i, sk_{ID_i}, pk_{ID_i})$. If the item associated with the identity ID_i has not been created, $\mathcal{C}$ generates $\{sk_{ID_i}, pk_{ID_i}\}$ by executing the **UserKeyGen** algorithm, and inserts $(ID_i, sk_{ID_i}, pk_{ID_i})$ into the list $\mathcal{L}$, $\mathcal{C}$ does nothing otherwise. In both cases, pk_{ID_i} is returned to $\mathcal{A}_1$.

- **ReplacePK** oracle: Given a user's identity ID_i along with a public key $pk^*_{ID_i}$, $\mathcal{C}$ checks the list $\mathcal{L}$ to confirm whether the item associated with ID_i has been created or not. If this item has already been created, $\mathcal{C}$ replaces the user's original public key pk_{ID_i} with $pk^*_{ID_i}$ by updating the item $(ID_i, sk_{ID_i}, pk_{ID_i})$ as $(ID_i, \perp, pk^*_{ID_i})$ in list $\mathcal{L}$. Otherwise, $\mathcal{C}$ adds $(ID_i, \perp, pk^*_{ID_i})$ into the list $\mathcal{L}$.

- **Corruption** oracle: Given a user's identity ID_i, $\mathcal{C}$ checks whether the item associated with ID_i has been created or not. If this item has already been created, $\mathcal{C}$ returns sk_{ID_i} as the answer. Otherwise, $\mathcal{C}$ generates $\{sk_{ID_i}, pk_{ID_i}\}$ by executing the **UserKeyGen** algorithm and inserts $(ID_i, sk_{ID_i}, pk_{ID_i})$ into the list $\mathcal{L}$. After that, sk_{ID_i} **is** returned as the answer.

- **Certification** oracle: Given a user's identity ID_i along with a public key pk_{ID_i}, $\mathcal{C}$ executes the algorithm **CertGen** and returns the certificate $Cert_{ID_i}$ as the answer.

- **Temporary secret key** oracle: Given a user's identity ID_i along with a period index t, $\mathcal{C}$ returns the temporary secret key $TSK_{ID,t}$ as the answer by executing the algorithm **UpdS**.

- **Sign** oracle: Given a tuple $\{ID, pk_{ID}, t, m\}$, $\mathcal{C}$ returns the signature (t, σ) as the answer by performing the algorithm **Sign**.

Forgery: $\mathcal{A}_1$ outputs a signature σ^* on message m^* in time period t^* under the identity ID^* and public key $pk^*_{ID_i}$. We say that $\mathcal{A}_1$ wins the game if: (a) **Verify** $(params, (t^*, \sigma^*), m^*, ID^*, pk^*_{ID}) = 1$. (b) $\{ID^*, pk^*_{ID_i}\}$ has never been submitted to the **Certification** query. (c) $\{ID^*, t^*\}$ has never been submitted to the **Temporary secret key** query. (d) $\{ID^*, pk^*_{ID}, m^*, t^*\}$has never been submitted to the **Sign** query.

Game II: (for the adversary $\mathcal{A}_2$)

Setup: In this phase, the **Setup** algorithm is run by the challenger $\mathcal{C}$. After that, *params* and msk will be sent to $\mathcal{A}_2$.

Queries: Similar to **Game I**, the **UserKeyGen**, **Corruption**, **Temporary secret key** and **Sign** oracles can be queried adaptively by $\mathcal{A}_2$ in this phase. Different from **Game I**, the **Certification** oracle is no longer needed to be queried due to the fact that the master secret key can be accessed by $\mathcal{A}_2$ itself. In addition, the **ReplacePK** oracle can not be accessed by $\mathcal{A}_2$.

Forgery: $\mathcal{A}_2$ outputs a signature σ^* on message m^* in time period t^* under the identity ID^*

and public key $pk^*_{ID_i}$. We say that $\mathcal{A}_2$ wins the game if: (a) **Verify**$(params, (t^*, \sigma^*), m^*, ID^*, pk^*_{ID}) = 1$. (b) ID^* has never been submitted to the **Corruption** query. (c) $\{ID^*, t^*\}$ has never been submitted to the **Temporary secret key** query. (d) $\{ID^*, pk^*_{ID}, m^*, t^*\}$ has never been submitted to the **Sign** query.

Definition 2. We say that a key insulated key CBS can achieve existential unforgeability against the adaptively chosen-message attack iff no adversary can win the **Game I** and **Game II** with non-negligible probability.

3. Our Proposed Scheme

In this section, the concrete construction of our key insulated CBS scheme is presented based on the idea in [14, 21].

- **Setup**: CA generates a prime p and determines the tuple $\{\mathbb{F}_p, E/\mathbb{F}_p, \mathbb{G}, P\}$ according to the definition in [23]. Here, E represents a elliptic curve defined over a prime finite field $\mathbb{F}_p$, and $\mathbb{G}$ is a cyclic additive group with generator P. After that, CA selects $x \in_R \mathbb{Z}^*_p$ as the master secret key and computes the master public key $Y = xP$. Four hash functions H_0 , H_1 , H_2 and H_3 are also picked up such that $H_0 : \{0,1\}^* \to \mathbb{Z}^*_p$, $H_1 : \{0,1\}^* \to \mathbb{Z}^*_p$, $H_2 : \{0,1\}^* \to \mathbb{Z}^*_p$ and $H_3 : \{0,1\}^* \to \mathbb{Z}^*_p$. Finally, CA publishes $params = \{\mathbb{F}_p, E/\mathbb{F}_p, \mathbb{G}, P, Y, H_0, H_1, H_2, H_3\}$ and keeps x secret.

- **UserKeyGen**: The user ID_i generates his user public/private key pair $(usk_{ID_i} = x_{ID_i}, upk_{ID_i} = x_{ID_i}P)$ such that x_{ID_i} is a secret value randomly chosen from $\mathbb{Z}^*_p$.

- **CertGen**: Given the public system parameter *params*, master secret key x along with the user public key upk_{ID_i} and identity ID_i, CA randomly chooses $s_{ID_i} \in_R \mathbb{Z}^*_p$ and computes the certificate (W_{ID_i}, d_{ID_i}) such that $W_{ID_i} = s_{ID_i}P$ and $d_{ID_i} = s_{ID_i} + xH_0(ID_i \| upk_{ID_i} \| W_{ID_i})$.

- **SetInitialKey**: The user ID_i chooses the helper key $hk_{ID_i} \in_R \mathbb{Z}^*_p$ and computes the initial secret key $TSK_{ID_i,0} = (S_{ID_i,0}, T_{ID_i})$ such that $T_{ID_i} = hk_{ID_i}P$ and $S_{ID_i,0} = x_{ID_i}H_1(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}) + hk_{ID_i}H_2(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, 0)$ After that, the helper key hk_{ID_i} will be stored in the helper and the initial temporary secret key $TSK_{ID,0}$ will be kept by the user.

- **UpdH**: The helper generates the update key $UK_{ID,t,t'}$ for the user ID from time period t to t' as follows:
$UK_{ID,t,t'} = hk_{ID_i}(H_2(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, t') - H_2(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, t))$

- **UpdS**: Given the update key $UK_{ID_i,t,t'}$, and the temporary secret key $TSK_{ID_i,t} = (S_{ID_i,t}, T_{ID_i})$ for a time period t, the user ID_i computes his temporary secret key for time period t' as $TSK_{ID_i,t} = (S_{ID_i,t'}, T_{ID_i})$, where $S_{ID_i,t'} = S_{ID_i,t} + UK_{ID,t,t'}$.

- **Sign**: Given the temporary secret key $TSK_{ID_i,t} = (S_{ID_i,t}, T_{ID_i})$ in time period t, and a message $m \in \{0,1\}^*$, the signer ID_i associated with the user public key upk_{ID_i} and certificate (W_{ID_i}, d_{ID_i}) performs the following steps:

  1. Randomly choose $r \in_R \mathbb{Z}_p^*$ and compute $U = rP$.
  2. Compute $z = d_{ID_i} + S_{ID_i,t} + rH_3(m, t, T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, U)$.
  3. Output $(z, U, W_{ID_i}, T_{ID_i})$ as the signature on m in time period t.

- **Verify**: To verify a signature $(z, U, W_{ID_i}, T_{ID_i})$ under the identity ID_i and public key upk_{ID_i} on messages m, the verifier performs the following steps:

  1. Computes $h_3 = H_3(m, T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, U)$.
  2. Checks whether the following equation holds or not:

$$zP \overset{?}{=} W_{ID_i} + H_0(ID_i \| upk_{ID_i} \| W_{ID_i})Y + H_1(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i})upk_{ID_i}$$
$$+ H_2(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, t)T_{ID_i} + H_3(m, t, T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, U)U$$

If it holds, accept the signature; else reject it.

4. Analysis of our scheme

4.1 Security analysis

We show that the unforgeability of our key insulated CBS scheme against chosen-message attack rests on the discrete logarithm assumption in this section.

Theorem 1. (Unforgeability against adversary $\mathcal{A}_1$) If a Type I adversary $\mathcal{A}_1$ has forged a valid key insulated CBS scheme successfully in Game I defined in Section 2.3, then the DL problem can be solved. If a Type I adversary $\mathcal{A}_1$ has an advantage ε in forging a certificate-based signature in Game I defined in Section 2.3 and making q_{H_i} (i=0, 1, 2, 3) queries to H_i queries, q_u queries to the UserKeyGen request oracle, q_{cert} queries to the Certification extraction oracle, q_r queries to the ReplacePK extraction oracle, q_{cor} queries to the Corruption extraction oracle, q_t queries to the Temporary secret key extraction oracle, and q_s queries to the `Sign` oracle, then the discrete logarithm problem can be solved with probability $\varepsilon' \geq \frac{1}{q_{H_0}}(1-\frac{1}{q_{H_0}})^{q_e+q_s}\varepsilon$.

Proof. The basic idea of our security proof, borrowed from [14,17], is that the simulator $\mathcal{C}$ can use the type I adversary $\mathcal{A}_1$ as a function to solve the DL problem. $\mathcal{C}$ is given a tuple $\{\mathbb{F}_p, E/\mathbb{F}_p, \mathbb{G}, P\}$ according to the definition in [23]. The task of $\mathcal{C}$ is to find $\alpha \in_R \mathbb{Z}_p^*$ in the presence of $Y=\alpha P$.

Setup: $\mathcal{C}$ chooses four hash functions $H_0:\{0,1\}^* \rightarrow \mathbb{Z}_p^*$, $H_1:\{0,1\}^* \rightarrow \mathbb{Z}_p^*$, $H_2:\{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and $H_3:\{0,1\}^* \rightarrow \mathbb{Z}_p^*$, and sends the system parameter $params = \{\mathbb{F}_p, E/\mathbb{F}_p, \mathbb{G}, P, Y, H_0, H_1, H_2, H_3\}$ to $\mathcal{A}_1$. Also, $\mathcal{C}$ will simulate these four hash functions as random oracles and keep α secret.

Queries: Seven initially empty lists $\mathcal{H}_0$, $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$, $\mathcal{K}$, $\mathcal{M}$ and $\mathcal{L}$ will be maintained by $\mathcal{C}$ to avoid the conflict of the simulation. A random index j will also be chosen by $\mathcal{C}$ such that $1 \leq j \leq q_{H_0}$ and q_{H_0} is the maximum times $\mathcal{A}_1$ can query the oracle H_0. After that, $\mathcal{C}$ sets $ID_j = ID^*$ where ID_j is the j-th query to the oracle H_0. Finally, $\mathcal{C}$ answers the following adaptive queries issued by $\mathcal{A}_1$.

H_0 oracle: Given a tuple $(ID_i, upk_{ID_i}, W_{ID_i})$ as the query, $\mathcal{C}$ first searches the list $\mathcal{H}_0$ to confirm whether the hash function of H_0 on this tuple has already been created or not. If not, $\mathcal{C}$ randomly chooses $h_{i0} \in \mathbb{Z}_p^*$ as a hash function of $(ID_i, upk_{ID_i}, W_{ID_i})$ and inserts the item $(ID_i, upk_{ID_i}, W_{ID_i}, h_{i0})$ into the list $\mathcal{H}_0$, and $\mathcal{C}$ does nothing otherwise. In both cases, h_{i0} will be returned as the answer.

H_1 oracle: Given a tuple $(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i})$ as the query, $\mathcal{C}$ first searches the list H_1 to confirm whether the hash function of H_1 on this tuple has already been created or not. If not, $\mathcal{C}$ randomly chooses $h_{i1} \in \mathbb{Z}_p^*$ as a hash function of $(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i})$ and inserts the item $(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, h_{i1})$ into the list H_1, and $\mathcal{C}$ does nothing otherwise. In both cases, h_{i1} will be returned as the answer.

H_2 oracle: Given a tuple $(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, t)$ as the query, $\mathcal{C}$ first searches the list H_2 to confirm whether the hash function of H_2 on this tuple has already been created or not. If not, $\mathcal{C}$ randomly chooses $h_{i2} \in \mathbb{Z}_p^*$ as a hash function of $(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, t)$ and inserts the item $(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, t, h_{i2})$ to H_2, and $\mathcal{C}$ does nothing otherwise. In both cases, h_{i2} will be returned as the answer.

H_3 oracle: Given a tuple $(m, t, T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, U)$ as the query, $\mathcal{C}$ first searches the list H_3 to confirm whether the hash function of H_3 on this tuple has already been created or not. If not, $\mathcal{C}$ randomly chooses $h_{i3} \in \mathbb{Z}_p^*$ as a hash function of $(m, t, T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, U)$ and inserts the item $(m, t, T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, U, h_{i3})$ to H_3, and $\mathcal{C}$ does nothing otherwise. In both cases, h_{i3} will be returned as the answer.

- `Certification` oracle: Given an identity ID_i along with the user public key upk_{ID_i}, $\mathcal{C}$ can generate the certificate for this user by performing the following steps.

1. If $i \neq j$, $\mathcal{C}$ scans the list $\mathcal{M}$ to confirm whether the certificate associated with (ID_i, upk_{ID_i}) has already been created or not. If not, $\mathcal{C}$ chooses $h_{i0}, d_{ID_i} \in \mathbb{Z}_p^*$ at random and computes $W_{ID_i} = d_{ID_i}P - h_{i0}Y$. Then $\mathcal{C}$ searches the list H_0 to confirm whether the item $(ID_i, upk_{ID_i}, W_{ID_i})$ has already been created or not. If so, $\mathcal{C}$ will rechoose $h_{i0}, d_{ID_i} \in \mathbb{Z}_p^*$ to avoid collision, otherwise $\mathcal{C}$ further inserts $(ID_i, upk_{ID_i}, W_{ID_i}, h_{i0})$ into

the list H_0 and $(ID_i, upk_{ID_i}, W_{ID_i}, d_{ID_i})$ into the list $\mathcal{M}$ respectively. In both cases, $\mathcal{C}$ then returns (W_{ID_i}, d_{ID_i}) as the answer.

2. If $i = j$, the simulation will be aborted.

- `UserKeyGen` oracle: Given a query ID_i, $\mathcal{C}$ first searches the list $\mathcal{K}$ to confirm whether the item associated with ID_i has already been created or not. If not, $\mathcal{C}$ selects $x_{ID_i} \in \mathbb{Z}_p^*$ at random as user secret key usk_{ID_i}, and computes the corresponding public key $upk_{ID_i} = x_{ID_i}P$. After that, $\mathcal{C}$ inserts the item $\{ID_i, usk_{ID_i}, upk_{ID_i}\}$ into the list $\mathcal{K}$. Otherwise, $\mathcal{C}$ does nothing. In both cases, upk_{ID_i} will be returned as the answer.
- `ReplacePK` oracle: Given an identity ID_i along with a user public key $upk^*_{ID_i}$, $\mathcal{C}$ searches the list $\mathcal{K}$ to confirm whether ID_i has already been created or not. If so, $\mathcal{C}$ updates the item $(ID_i, upk_{ID_i}, usk_{ID_i})$ as $(ID_i, upk^*_{ID_i}, \perp)$. Otherwise, $\mathcal{C}$ inserts the item $(ID_i, upk^*_{ID_i}, \perp)$ into list $\mathcal{K}$.
- `Corruption` oracle: Given the identity ID_i, $\mathcal{A}_1$ performs the following steps to generate the user secret key for this user.
  1. If there is an item $(ID_i, upk_{ID_i}, usk_{ID_i})$ in the list $\mathcal{K}$, $\mathcal{C}$ returns usk_{ID_i} as the answer.
  2. Otherwise $\mathcal{C}$ chooses $x_{ID_i} \in \mathbb{Z}_p^*$ as his secret key usk_{ID_i} , and computes $upk_{ID_{i1}} = x_{ID_i}P$. Then $\mathcal{C}$ returns usk_{ID_i} as the answer and inserts the item $(ID_i, upk_{ID_i}, usk_{ID_i})$ into the list $\mathcal{K}$.
- `Temporary secret key` oracle: Given an identity ID_i and time index t, $\mathcal{C}$ first searches the list $\mathcal{L}$ to confirm whether the item associated with ID_i has already been created or not. If not, $\mathcal{C}$ selects $hk_{ID_i} \in \mathbb{Z}_p^*$ at random as the helper key, computes $T_{ID_i} = hk_{ID_i}P$, and inserts the item $(ID_i, hk_{ID_i}, T_{ID_i})$ into the list $\mathcal{L}$. Otherwise, $\mathcal{C}$ extracted the hk_{ID_i} from the item $(ID_i, hk_{ID_i}, T_{ID_i})$ in the list $\mathcal{L}$. After that, $\mathcal{C}$ extracted the x_{ID_i} by querying the `UserKeyGen` oracle. Finally, $\mathcal{C}$ chooses $h_{i1}, h_{i2} \in \mathbb{Z}_p^*$ at random and computes $S_{ID_i,t} = x_{ID_i}h_{i1} + hk_{ID_i}h_{i2}$. Then, $\mathcal{C}$ sets $(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}) = h_{i1}$ and $(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, t) = h_{i2}$. If the hash functions H_1 and H_2 have already been defined, $\mathcal{C}$ need to rechoose these random values until the collision is avoided. Otherwise, $\mathcal{C}$ inserts the item $\{ID_i, upk_{ID_i}, usk_{ID_i}\}$ into the list $\mathcal{K}$, the item $(ID_i, hk_{ID_i}, T_{ID_i})$ into the list $\mathcal{L}$, the item $(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i})$ into the list $\mathcal{H}_1$, the item $(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, t)$ into the list H_2 respectively, and returns $(T_{ID_i}, S_{ID_i,t})$ as the answer.
- `Sign` oracle: Given a tuple (m, t, ID_i, upk_{ID_i}) as the query, $\mathcal{C}$ queries the `Temporary secret key` oracle to get $(T_{ID_i}, S_{ID_i,t})$. After that, $\mathcal{C}$ creates a valid signature as follow.
  1. If $i = j$, $\mathcal{C}$ chooses $h_{j0}, h_{j1}, h_{j2}, h_{j3}, z_j \in \mathbb{Z}_p^*$ at random and computes $U_j = h_{j3}^{-1}(z_jP - h_{j1}upk_{ID_i} - h_{j2}T_{ID_i})$ and $W_{ID_j} = h_{j0}Y$. After that, $\mathcal{C}$ sets $H_0(ID_j \| upk_{ID_j} \| W_{ID_j}) = h_{j0}, H_1(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}) = h_{j1}$, $H_2(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, t) = h_{j2}$ and $H_3(m, t, T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, U) = h_{j3}$. If the hash functions H_0, H_1, H_2 and H_3 have already been defined, $\mathcal{C}$ needs to rechoose these random values until the collision is avoided. Otherwise, $\mathcal{C}$ inserts the item $(ID_j, upk_{ID_j}, W_{ID_j}, h_{j0})$ into the list $\mathcal{H}_0$, the item $(T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, h_{j1})$ into the list H_1, the item

$T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, t, h_{j2})$ into the list $\mathcal{H}_2$, the item $(m, t, T_{ID_i}, ID_i, upk_{ID_i}, W_{ID_i}, U_j, h_{j3})$ into the list $\mathcal{H}_3$. Finally, the signature $(z_j, U_j, W_{ID_j}, T_{ID_j})$ is returned as the answer.

2. If $i \neq j$, $\mathcal{C}$ queries the `Certification` oracle to get (W_{ID_i}, d_{ID_i}), and generates the signature by performing the **Sign** algorithm using the certificate (W_{ID_i}, d_{ID_i}) and the temporary secret key $(T_{ID_i}, S_{ID_i,t})$.

Forgery: Finally, $\mathcal{A}_1$ outputs a valid signature $(z^*, U^*, W_{ID^*}, T_{ID^*})$ on message m^* on behalf of the identity ID^* and the corresponding public key upk^*_{ID} in time period t^*. If $ID^* \neq ID_j$, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ can replay $\mathcal{A}_1$ with the same random tape in the simulation of H_1, H_2 and H_3 but different choice of the hash function H_0 based on the forking lemma [24]. After that, $\mathcal{C}$ can get another valid signature $(z', U^*, W_{ID^*}, T_{ID^*})$. From these two valid ring signatures, $\mathcal{C}$ obtain

$$z^*P = W_{ID^*} + h^*_{i0}Y + h^*_{i1}upk^*_{ID} + h^*_{i2}T_{ID^*} + h^*_{i3}U^*$$

and

$$z'P = W_{ID^*} + h'_{i0}Y + h^*_{i1}upk^*_{ID} + h^*_{i2}T_{ID^*} + h^*_{i3}U^*$$

According to the above two equations we observe that $\alpha = (h^*_{i0} - h'_{i0})^{-1}(z^* - z')$. It is obvious that the DL problem can be solved by $\mathcal{C}$ successfully.

This completes the descripton of the simulation and it remains to analyze the probability of $\mathcal{C}$'s advantage. Define the event E_1 as that $\mathcal{C}$ does not abort when answering oracle queres, the event E_2as that $\mathcal{A}_1$outputs a valid signature successfully and the eventE_3as that the identity of the forged signature output by the $\mathcal{A}_1$ in the forgery phase is ID_j. Observations from the simulation demonstrate that $\Pr[E_1] \geq (1 - \frac{1}{q_{H_0}})^{q_e+q_s}$, $\Pr[E_2] = \varepsilon$ and $\Pr[E_3|E_3 \wedge E_3] = \frac{1}{q_{H_0}}$. Thus, the success probability of $\mathcal{C}$ solving the DL problem is $\varepsilon' \geq \frac{1}{q_{H_0}}(1 - \frac{1}{q_{H_0}})^{q_e+q_s}\varepsilon$.

Theorem 2. (Unforgeability against adversary $\mathcal{A}_2$) If a Type II adversary $\mathcal{A}_2$ has an advantage ε in forging a certificate-based signature in Game II defined in Section 2.3 and making q_{H_i} (i=0, 1, 2, 3) queries to H_i queries, q_u queries to the `UserKeyGen` request oracle, q_{cert} queries to the `Certification` extraction oracle, q_r queries to the `ReplacePK` extraction oracle, q_{cor} queries to the `Corruption` extraction oracle, q_t queries to the `Temporary secret key` extraction oracle, and q_s queries to the `Sign` oracle, then the discrete logarithm problem can be solved with probability $\varepsilon' \geq \frac{1}{q_{H_0}}(1 - \frac{1}{q_{H_0}})^{q_{cor}+q_s}\varepsilon$.

The security proof is similar to Theorem 1 and is omitted here.

Theorem 3. Our key insulated CBS scheme can offer secure key-updates.

This theorem follows from the fact that for any target identity ID_i, public key upk_{ID_i} and the time indices i and j, the update key $UK_{ID,t,t'}$ can be derived from the temporary secret key $S_{ID_i,t}$ and $S_{ID_i,t'}$.

4.2 Performance evaluation

We compare our approach with Li *et al.*'s key insulated CBS scheme [17] in terms of communication overhead and the computation cost. To offer the security level equal to 1024-bit RSA in Li *et al.*'s pairing-based scheme, a Tate pairing defining on the supersingular elliptic curve $E/\mathbb{F}_p : y^2 = x^3 + x$ is adopted. Here, the embedding degree of curve $E/\mathbb{F}_p$ is

2, $q = 2^{159} + 2^{17} + 1$ refers to a 160-bit Solinas prime, and $p = 12qr - 1$ is a 512-bit prime. To achieve the similar level of security for our pairing-free approach, the Koblitz elliptic curve $y^2 = x^3 + ax^2 + b$ defined on $\mathbb{F}_{2^{163}}$ can be used. Here, $a = 11$ and b is a randomly chosen prime with the length of 163-bit. The running time of the cryptographic operation listed in **Table 1** can be derived using the standard cryptographic library MIRACAL [25], and the hardware and OS for the experiment is PIV 3 GHZ processor with 512 M bytes storage capacity, and the Windows XP operating system respectively [26].

Table 1. Cryptographic operation time in milliseconds

Operations	Time
ECC-based scalar multiplication	0.83
Exponential in $\mathbb{F}_{p^2}$	11.20
Pairing-based scalar multiplication	6.38
Pairing	20.01

The computation and communication efficiency is evaluated based on the method proposed in [27]. For example, in the **Verify** algorithm of Li *et al*.'s scheme [17], five pairing operations are needed, and thus the computation time is $5 \times 20.01 = 100.05$ ms. The signature Li *et al*.'s scheme [17] consists of three points in the pairing-based group, and thus the bandwidth for Li *et al*.'s scheme [17] is $512 \times 3/8 = 192$ byte. Observing the comparison results listed in **Table 2**, our scheme outperforms the existing key insulated CBS scheme and is more suitable for the low power device.

Table 2. Performance comparisons with existing work

	Bandwidth (in bytes)	Sign (in ms)	Verify (in ms)
[17]	192	12.76	100.05
Our scheme	82	1.66	4.15

5. Conclusion

In this paper, a pairing-free key insulated CBS scheme has been proposed. The proposal can achieve unforgeability in the random oracle model provided that the discrete logarithm problem is hard. Our approach can trigger the deployment of CBS in the hostile and resource-limited scenarios.

References

[1] W. Diffie, M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644-654, 1976. Article (CrossRef Link)

[2] R. L. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, 1978. Article (CrossRef Link)

[3] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *Advances in Cryptology-CRYPTO 1984*, Springer-Verlag, LNCS 196, pp. 10-18, 1984. Article (CrossRef Link)

[4] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *Advances in Cryptology- ASIACRYPT 2001*, Springer-Verlag, LNCS 2248, pp. 514-532, 2001. Article (CrossRef Link)

[5] A. Shamir, "Identity-based cryptosystems and signature schemes," *Advances in Cryptology-CRYPTO 1984*, Springer-Verlag, LNCS 196, pp. 47-53, 1984. Article (CrossRef Link)

[6] F. Hess, "Efficient identity based signature schemes based on pairings," *Selected Areas in Cryptography-SAC 2002*, Springer-Verlag, LNCS 2595, pp. 310-324, 2003. Article (CrossRef Link)

[7] K. G. Paterson, "ID-based signatures from pairings on elliptic curves," *Electronics Letters*, vol. 38, no. 18, pp. 1025-1026, 2002. Article (CrossRef Link)

[8] M. Bellare, C. Namprempre, G. Neven, "Security Proofs for Identity-Based Identification and Signature Schemes," *Journal of Cryptology*, vol. 22, no. 1, pp. 1-61, 2009. Article (CrossRef Link)

[9] C. Gentry, "Certificate-based encryption and the certificate revocation problem," *Advances in Cryptology- EUROCRYPT 2003*, Springer-Verlag, LNCS 2656, pp. 272-293, 2003. Article (CrossRef Link)

[10] B. G. Kang, J. H. Park, S. G. Hahn, "A certificate-based signature scheme," in *Proc. of Topics in Cryptology-CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004*, Springer-Verlag, LNCS 2964, pp. 99-111, 2004. Article (CrossRef Link)

[11] K. Joseph, J. B. Liu, S. Willy, J. Zhou, "Certificate-Based Signature Schemes without Pairings or Random Oracles," in *Proc. of 11th International Conference on Information Security (ISC 2008)*, Springer-Verlag, LNCS 5222, pp. 285-297, 2008. Article (CrossRef Link)

[12] J. Li, X. Huang, Y. Mu, W. Susilo, and Q. Wu, "Certificate-based signature: Security model and efficient construction," in *Proc. of 4th European PKIWorkshop: Theory and Practice (EuroPKI' 07)*, Springer-Verlag, LNCS 4582, pp. 110-125, 2007. Article (CrossRef Link)

[13] J. Li, X. Huang, X. Zhang, L. Xu, "An efficient short certificate-based signature scheme," *Journal of Systems and Software*, vol. 85, no. 2, pp. 314-322, 2012. Article (CrossRef Link)

[14] J. Li, Z. Wang, Y. Zhang, "Provably secure certificate-based signature scheme without pairings," *Information Sciences*, vol. 233, no. 1, pp. 313-320, 2013. Article (CrossRef Link)

[15] Y. Dodis, J. Katz, S. Xu, and M.Yung, "Key-insulated public key cryptosystems," in *Advances in Cryptology- Eurocrypt'02*, Springer-Verlag, LNCS 2332, pp. 65-82, 2002. Article (CrossRef Link)

[16] Y. Dodis, J. Katz, S. Xu, and M.Yung, "Strong key-insulated signature scheme," in *Proc. of 6th International Workshop on Practice and Theory in Public Key Cryptography(PKC 2003)*, Springer-Verlag, LNCS 2567, pp. 130-144, 2003. Article (CrossRef Link)

[17] J. Li, H. Du, Y. Zhang, T. Li, Y. Zhang, "Provably Secure Certificate-based Key-Insulated Signature Scheme," *Concurrency and Computation Practice and Experience*, vol. 26, no. 8, pp. 1546-1560, 2014. Article (CrossRef Link)

[18] D. Hofheinz, T. Jager, E. Kiltz, "Short signatures from weaker assumptions," *Advances in Cryptology-ASIACRYPT 2011*, LNCS 7073, Berlin: Springer-Verlag, pp. 647-666, 2011. Article (CrossRef Link)

[19] L. Chen, Z. Cheng, N. P. Smart, "Identity-based key agreement protocols from pairings", *International Journal of Information Security*, vol. 6, no. 4, pp. 213-241, 2007. Article (CrossRef Link)

[20] M. Bellare, P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," in *Proc. of 1st ACM Conf. on Computer and Communications Security (CCS 1993)*, pp. 62-72, 1993. Article (CrossRef Link)

[21] J. Weng, S. Liu, K. Chen, X. Li., "Identity-Based Key-Insulated Signature with Secure Key-Updates," in *Proc. of 2nd SKLOIS Conference on Information Security and Cryptology-Inscrypt 2006*, Springer-Verlag, LNCS 4318, pp. 13-26, 2006. Article (CrossRef Link)

[22] Y. Zhou, Z. Cao, and Z. Chai, "Identity based key insulated signature," in *Proc. of 2nd International Conference on Information Security Practice and Experience (ISPEC 2006)*, Springer-Verlag, LNCS 3903, pp. 226-234, 2006. Article (CrossRef Link)

[23] A. Cilardo, L. Coppolino, N. Mazzocca, L. Romano, “Elliptic curve cryptography engineering,” *Proceedings of the IEEE*, vol. 94, no. 2, pp. 395-406, 2006. Article (CrossRef Link)

[24] D. Pointcheval, J. Stern, “Security arguments for digital signatures and blind signatures,” *Journal of Cryptology*, vol. 13, no. 3, pp. 361-369, 2000. Article (CrossRef Link)

[25] Shamus Software Ltd., “Multiprecision Integer and Rational Arithmetic Cryptographic Library (Miracl)”, http://www.certivox.com/miracl/

[26] X. Cao, W. Kou, X. Du, “A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges,” *Information Sciences*, vol. 180, no. 15, pp. 2895-2903, 2010. Article (CrossRef Link)

[27] K. Ren, W. Lou, K. Zeng, P. J.Moran, “On broadcast authentication in wireless sensor networks,” *IEEE Trans. Wireless Commun.*, vol. 6, no. 11, pp. 4136-4144, 2007. Article (CrossRef Link)

Hu Xiong received his Ph.D. degrees from University of Electronic Science and Technology of China (UESTC) in 2009. He is now an associate professor in the UESTC. His research interests include cryptography and ad hoc networks security.

Shikun Wu received his B.S. degree in the School of Computer Science and Engineering, Anhui University of Science and Technology (AUST) in Jun 2009. He is currently pursuing his M.S. degree in the School of Computer Science and Engineering, UESTC. His research interests include cryptographic protocols and network security.

Ji Geng is a professor in the School of Computer Science and Engineering, UESTC. He received his Ph.D. degrees from UESTC in 2014. His research interests include: information security and system software.

Emmanuel Ahene received his B.S degree in computer science from the University for Development Studies, Ghana in 2012. He is currently pursuing his M.S degree in Computer Science and Technology at the School of computer science and Engineering, University of Electronic Science and Technology of China. His research interest include Cloud computing and Information security.

Songyang Wu is an associate professor at The Third Research Institute of Ministry of Public Security, China .Vice director. He received his Ph.D. Degree in computer Science from TongJi University, China in 2011. His current research interests are in information security, cloud computing and digital forensics.

Zhiguang Qin is a full professor in the School of Computer Science and Engineering and now with the president of School of Computer Science and Engineering, UESTC. He received his Ph.D. degree from UESTC in 1996. His research interests include: information security and wireless networks.