www.ietdl.org

Published in IET Information Security
Received on 17th April 2012
Revised on 28th November 2012
Accepted on 31st December 2012
doi: 10.1049/iet-ifs.2012.0134

ISSN 1751-8709

Strong non-repudiation based on certificateless short signatures

Yu-Chi Chen[1], *Gwoboa Horng*[1], *Chao-Liang Liu*[2]

[1]*Department of Computer Science and Engineering, National Chung Hsing University, Taichung 402, Taiwan*
[2]*Department of Applied Informatics and Multimedia, Asia University, Taichung 413, Taiwan*
E-mail: gbhorng@cs.nchu.edu.tw

Abstract: In this study, the authors consider certificateless signature (CLS) schemes for strong non-repudiation. They show that previous security models which ensure that any user can have a unique key pair cannot guarantee a CLS scheme to achieve strong non-repudiation. The authors then fix the security model and propose a new CLS scheme which provides strong non-repudiation under the new model, assuming the computational Diffie–Hellman problem is intractable.

1 Introduction

Digital signatures, based on public key cryptography, can provide authenticity, integrity and non-repudiation of messages. A digital signature scheme involves two entities, a signer and a verifier. A signer generates a signature of a message with his private key and a verifier verifies the signature using the signer's authenticated public key.

In traditional public key cryptosystems, the certificates of public keys, generated by a trusted certificate authority (CA), serve as the authentication of the public keys. Whereas identity-based public key cryptosystems (ID-PKC) [1] and certificateless public key cryptosystems (CL-PKC) [2] do not need the extra trusted party to manage certificates. However, they still need a trusted key generation centre (KGC) to generate private keys. Differing from ID-PKC, the KGC in CL-PKC is unable to derive the user's actual private key. That is, CL-PKC does not suffer from the key escrow problem. Both CA and KGC are third parties. The security of the corresponding public key schemes depends on the trustiness of these third parties. In 1991, Girault defined three trust levels for a trusted third party (TTP) [3]. The higher the trust level of the TTP is, the higher the security level of the cryptographic scheme is.

- *Level 1*. The TTP knows the private key of any user and is able to impersonate any user without being detected.
- *Level 2*. The TTP does not know the private key of any user. But the TTP is able to generate a false private key to impersonate any user.
- *Level 3*. The TTP does not know the private key of any user. But if the TTP generates a false private key to impersonate a user then it is possible for that user (the victim) to prove that the TTP generated a false private key.

Schemes with trust level 1 or trust level 2 are not acceptable in many applications, such as providing non-repudiations. Reaching trust level 3 is generally the goal. In a traditional public key scheme, if the CA forges certificates, the CA's misbehave can be identified through the existence of two valid certificates for the same user. However, a false public key can be created by the KGC without being detected in the certificateless PKC, since new public keys can be created by both the legitimate user and the KGC. Therefore the traditional public key schemes achieve trust level 3, whereas the certificateless public key schemes reach only trust level 2.

Explicitly, we state clearly the three trust levels of the KGC in the context of certificateless signature (CLS) schemes:

- *Level 1*. The KGC knows the full private key of any user and is able to act as any user to forge signatures and these forged signatures cannot be repudiated by that user (the victim).
- *Level 2*. The KGC does not know the full private key of any user. But the KGC is able to generate a false private key for any user to forge signatures and these forged signatures cannot be repudiated by that user (the victim).
- *Level 3*. The KGC does not know the full private key of any user. But the KGC is able to generate a false private key and public key for any user to forge signatures but that user (the victim) can repudiate these forged signatures.

From legal point of view, using a digital signature scheme with trust level 1 or 2, a signer can always repudiate signatures by blaming the KGC. We say a CLS scheme providing *strong non-repudiation* if the corresponding KGC is of trust level 3. Therefore for a CLS scheme to provide strong non-repudiation, the user must be able to provide proofs to accuse the KGC for wrongdoing (i.e. forging signatures based on his identity). For convenience we say that a CLS scheme achieves level i security if the KGC is of trust level i where $i = 1$, 2 or 3.

© The Institution of Engineering and Technology 2013

In traditional CLS schemes [2, 4], it seems that the KGC does not have the ability to forge signatures since it can only derive user's partial-private-key. Most of schemes, for examples [2, 4–14], address the security issue via two types of adversaries with different attack capabilities. A type 1 adversary is able to replace the public key of any user but cannot access the master key of KGC. Whereas a type 2 adversary is able to access the master key but cannot replace the public key [15–17]. However, based on these two security models, CLS schemes can only achieve level 2 security. In 2007, Hu *et al.* [16] presented a generic construction, based on a new security model, which achieves level 3 security. In 2011, Fan *et al.* [18] proposed an improved CLS scheme, and claimed their scheme can achieve level 3 security based on the same security model, assuming that Boneh and Boyen's short signature scheme [19] is secure.

The contribution of this paper is three-fold. First, we show that the security model proposed by Hu *et al.* fail to guarantee level 3 security by showing the scheme proposed by Fan *et al.* does not achieve level 3 security. Second, a new security model for level 3 security is proposed. Third, based on this new security model, we propose a new CLS scheme which can provide strong non-repudiation, assuming the computational Diffie–Hellman (CDH) problem is intractable. The proposed scheme can achieve the same trust level 3 as public key infrastructure -based systems do without involving any CA. As a result, it also can provide non-repudiation services, including non-repudiation of origin and submission [20].

The rest of this paper is organised as follows. Section 2 consists of some preliminaries, including a generic construction of a CLS scheme, security models and some related work. In Section 3, we discuss the security models for CLS schemes. A new CLS scheme with level 3 security and its formal proof are proposed in Section 4. Finally, we conclude this paper in Section 5.

2 Preliminaries

2.1 Generic construction of a certificateless signature scheme

A CLS scheme consists of three phases, initial setup phase **Init**

Setup, signature generation phase **CL-Sign** and signature verification phase **CL-Verify**:

InitSetup: This phase consists of the following algorithms:

- *Setup*: This algorithm, run by the KGC, takes a security parameter as input, then outputs master-key and system parameter params.
- *Partial-Private-Key-Extract*: This algorithm, run by the KGC, takes **params**, **master-key** and a user's identity ID as inputs, then outputs a partial-private-key D_{ID} to that user.
- *Set-Secret-Value*: This algorithm, run by a user, returns a secret value.
- *Set-Private-Key*: This algorithm, run by a user, takes the user's partial-private-key D_{ID} and his secret value as inputs, and outputs the full private key.
- *Set-Public-Key*: This algorithm, run by a user, takes params and the user's full private key as inputs, and outputs a public key $\mathrm{pk}_{\mathrm{ID}}$ for that user.

CL-Sign: This phase consists of a signature generation algorithm. The algorithm, run by a signer, takes params, a message m, and the user's full private key as inputs, and outputs S as the signature for the message m.

CL-Verify: This phase consists of a signature verification algorithm. The algorithm, run by a verifier, takes params, a public key $\mathrm{pk}_{\mathrm{ID}}$, a message m, a user's identity ID, and a signature S as inputs. The verifier accepts signature S if and only if S is the signature of the message m for the public key $\mathrm{pk}_{\mathrm{ID}}$ of the user with identity ID.

We note that a malicious KGC is able to access the master key as well as to replace users' public keys. We shall call this attack the 'malicious KGC public key replacement' (MKPKR) attack. Let Bob, with identity ID, be a user in a CLS scheme. The malicious KGC, says Alice, can launch the following attack:

- *Step 1*. Alice uses the master-key s' to compute Bob's partial-private-key D_{ID}.
- *Step 2*. Alice sets a new secret value corresponding of identity ID, and computes the public key $\mathrm{pk}'_{\mathrm{ID}}$.
- *Step 3*. Finally, Alice computes a valid signature S' which can be verified using Bob's identity ID and the public key $\mathrm{pk}'_{\mathrm{ID}}$.

However, Bob cannot repudiate S' since he has no way to prove that $\mathrm{pk}'_{\mathrm{ID}}$ is not his public key. Several CLS schemes [4, 8, 12, 13] are also vulnerable to the MKPKR attack; in fact, they can only achieve level 2 security.

2.2 Bilinear map and some hard problems

A bilinear map is a mapping $\hat{e}:\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, where $\mathbb{G}_1$ is an additive cyclic group of prime order q, and $\mathbb{G}_2$ is a multiplicative cyclic group of the same order q. We are interested in bilinear maps with the following properties: (1) computable: given P, $Q \in \mathbb{G}_1$, there exists a polynomial time algorithm to compute $\hat{e}(P, Q) \in \mathbb{G}_2$. (2) Bilinear: for any x, $y \in \mathbb{Z}_q*$, we have $\hat{e}(xP, yP) = \hat{e}(P, P)^{xy}$ for any $P \in \mathbb{G}_1$. (3) Non-degenerate: if P is a generator of $\mathbb{G}_1$, then $\hat{e}(P, P)$ is a generator of $\mathbb{G}_2$.

The following problems are assumed to be intractable when proving the security of some CLS schemes.

CDH Problem: Given a group $\mathbb{G}_1$ of order q, and aP, bP, compute abP, where P is a generator of $\mathbb{G}_1$ and $a, b \in \mathbb{Z}_q^*$ are unknown.

k-CAA Problem [21]: Given a group $\mathbb{G}$ of order q, and an integer k, $P \in \mathbb{G}_1, sP, w_1, w_2 \ldots, w_k \in \mathbb{Z}_q^*, 1/\left(s + w_1\right)P, 1/\left(s + w_2\right)P, \ldots, 1/\left(s + w_k\right)P$, find a pair $\{w, 1/(s+w)P\}$ such that $w \notin \{w_1, w_2 \ldots, w_k\}$, where $s \in \mathbb{Z}_q^*$ is unknown.

Modified k*-CAA Problem [22]*: Given a group $\mathbb{G}_1$ of order q, and an integer k, $P \in \mathbb{G}, rP, aP, bP, raP, w_1, w_2 \ldots, w_k \in \mathbb{Z}_q^*, 1/\left(r + w_1\right)(abP), 1/\left(r + w_2\right)(abP), \ldots, 1/\left(r + w_k\right)(abP)$, find a pair $\{w, 1/(s+w)(abP)\}$ such that $w \notin \{w_1, w_2, \ldots, w_k\}$, where $r, a, b \in \mathbb{Z}_q^*$ are unknown.

The modified k-CAA problem is proven to be polynomial time equivalent to the original k-CAA problem [22].

2.3 Du–Wen's CLS scheme

The three phases of Du–Wen's CLS scheme [4] are as follows:

InitSetup:

- *Setup*: The KGC, takes a security parameter k as input of the setup algorithm and determines a bilinear map $\hat{e}:\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ where $\mathbb{G}_1$ is a cyclic additive group of prime order q with a generator P, $\mathbb{G}_2$ is a cyclic

© The Institution of Engineering and Technology 2013
IET Inf. Secur., 2013, Vol. 7, Iss. 3, pp. 253–263
doi: 10.1049/iet-ifs.2012.0134

multiplicative group of the same order, and two hash functions $H_1:\{0,1\}^* \to \mathbb{Z}_q^*$ and $H_2:\{0,1\}^* \to \mathbb{Z}_q^*$. Then the KGC randomly chooses $s \in \mathbb{Z}_q^*$ as *master-key* and sets master-public-key $P_{pub} = sP$. Finally, the KGC announces the system parameter *params* = $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, q, g, P, P_{pub}, H_1, H_2 \rangle$ and *master-key* = s, where $g = \hat{e}(P, P)$.

- *Set-Secret-Value*: A user generates a random value $r_{ID} \in \mathbb{Z}_q^*$ and sets r_{ID} as his secret value.
- *Set-Public-Key:* A user takes params, his secret value r_{ID} and identity ID as inputs, and computes his public key $pk_{ID} = r_{ID}T$ where $T = P_{pub} + H_1(ID)P$.
- *Partial-Private-Key-Extract:* The KGC takes params, master-key, a user's identity ID, and the corresponding public key pk_{ID} as inputs, and returns a partial-private-key $D_{ID} = 1/(s + H_1(ID))P$ to the user.
- *Set-Private-Key:* A user computes his full private key $sk_{ID} = (D_{ID}, r_{ID})$ from the user's partial-private-key D_{ID} and secret value r_{ID}.

CL-Sign: To generate a signature for a message m, a signer with private key sk_{ID}, computes $S = 1/(r_{ID} + h)D_{ID}$ as the signature for the message where $h = H_2(m, pk_{ID})$.

CL-Verify: To verify a signature S of a message m generated by a user with identity ID and public key pk_{ID}, a verifier takes *params*, the public key pk_{ID}, the message m, the user's identity ID, and the signature S as inputs, then computes $h = H_2(m, pk_{ID})$, $T = P_{pub} + H_1(ID)P$ and accepts this signature if and only if $\hat{e}(S, pk_{ID} + hT) = g$.

2.4 Fan et al.'s scheme

Fan *et al*. analysed Du–Wen's scheme and showed that it is unable to achieve level 3 security [18]. They modified Du–Wen's scheme and proposed another CLS scheme as follows:

Setup: The KGC takes a security parameter k as input of the setup algorithm and determines a bilinear map $\hat{e}:\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic additive groups of prime order q with generators P_1 and P_2 respectively, $\mathbb{G}_T$ is a cyclic multiplicative group of the same order with a generator $g = \hat{e}(P, P)$, and two hash functions $H_1:\{0,1\}^* \to \mathbb{Z}_q^*$, $H_2:\{0,1\}^* \times \mathbb{G}_2 \to \mathbb{Z}_q^*$. Then the KGC randomly chooses $s \in \mathbb{Z}_q^*$ as *master-key* and sets master-public-key $P_{pub} = sP$. Finally, the KGC announces the system parameters *params* = $\langle \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, q, P_1, P_2, g, P_{pub}, H_1, H_2 \rangle$.

Set-Secret-Value: A user generates a random value $r \in \mathbb{Z}_q^*$ and sets r as his secret value.

Set-Public-Key: A user takes *params* and his secret value r as inputs, and computes $pk_{ID} = rP_2$ and $pk'_{ID} = r(P_{pub} + Q_{ID}P_2)$ where $Q_{ID} = H_1(ID)$. (pk_{ID}, pk'_{ID}) is the user's public key.

Partial-Private-Key-Extract: The KGC takes *params*, *master-key* and a user's identity ID as inputs, then outputs a partial-private-key $D_{ID} = 1/(s + Q_{ID} + H_1(ID//pk_{ID}))P_1$ to the user.

Set-Private-Key: A user takes his partial-private-key D_{ID} and secret value r as inputs, and sets his full private key as $sk_{ID} = (D_{ID}, r)$.

CL-Sign: To sign a message m, a signer, with private key sk_{ID}, computes $S = 1/(r + h)D_{ID}$ as the signature for the message m where $h = H_2(m, pk_{ID})$.

CL-Verify: Given *params*, a public key (pk_{ID}, pk'_{ID}), a message m, a singer's identity ID, and a signature S, the verifier computes $h = H_2(m, pk_{ID})$ and accepts the signature if and only if $\hat{e}(S, pk'_{ID} + H_1(ID||pk_{ID})pk_{ID} + h(P_{pub} + Q_{ID}P_2 + H_1(ID||pk_{ID})P_2)) = g$.

3 Security models

Traditionally, a certificate-based digital signature scheme is secure if it is existentially unforgeable against adaptive chosen message attacks. The adversaries do not include the signers themselves and the attack methods are centred on querying signatures for adaptive chosen messages. For a CLS scheme, the situation is more complicated due to potentially many valid public keys of a user. Therefore we need to consider legitimate users acting as adversaries. Furthermore, there are interactions between users and the KGC (when generating keys). Therefore the attackers can do a lot more than merely querying signatures. For example, they can query for the partial private key of any user.

We will distinguish the following types of adversaries. An outsider, referred as a type 1 adversary, can try to forge a valid signature. The KGC, referred as a type 2 adversary, cannot perform the public key replacement attack since the victim can prove that the KGC has misbehaved (assuming that any user can has only a single key pair). The signer himself, referred as a type 3 adversary, can try to perform the public key replacement attack to come up with a valid signature to frame the KGC. Different types of adversaries are with different capabilities. A type 1 adversary, $\mathcal{A}_1$, does not access to the master key, but it is able to replace the public key of any user. A type 2 adversary, $\mathcal{A}_2$, cannot replace the public key of any user, but it is able to access to the master key [2]. They will perform the chosen message and identity attack to existentially forge signatures. These two types of adversarial models are described in two games, namely Game I and Game II. [In [10], Huang *et al*. had defined three different adversaries, normal, strong and super adversaries according to their attack powers. However, the normal type 1 and 2 adversaries are said to be more realistic than others. In this paper, the type 1 and 2 adversaries, mentioned in this section, are normal ones.] A type 3 adversary, a legitimate but malicious user, can try to produce many key pairs with a single identity or try to generate a valid signature which can be verified with a different public key. We will model these attacks in Game III and Game IV. We note that Yang and Tan [23] also presented new security models to analyse level 3 security. However, their new defined type 1 and 2 adversaries act as the outsiders and the KGC, not a legitimate user. Although they claimed that their security models can be used to show level 3 security, we take the real-life scenario, mentioned in Section 2.1, into consideration. Our models are more appropriate than theirs.

3.1 Existential unforgeability for outsiders

The attack methods and goals of a type 1 adversary, $\mathcal{A}_1$ and a type 2 adversary, $\mathcal{A}_2$, are modelled in the following two games.

Game I: $\mathcal{A}_1$ interacts with challenger $\mathcal{C}$.

Setup: $\mathcal{C}$ performs Setup by inputing a security parameter to obtain the system parameter *params*. $\mathcal{C}$ sends *params* to $\mathcal{A}_1$.

Attack: $\mathcal{A}_1$ can adaptively perform the following polynomially bounded queries.

© The Institution of Engineering and Technology 2013

- *Partial-Private-Key query*: $\mathcal{A}_1$ can query for the partial private key of any user with identity ID_i. $\mathcal{C}$ will return the partial private key D_i to $\mathcal{A}_1$.
- *Public-Key query*: $\mathcal{A}_1$ can query for the public key of any user with identity ID_i. $\mathcal{C}$ will return the public key pk_i of that user.
- *Secret-Value query*: $\mathcal{A}_1$ can query for the secret value of any user with identity ID_i. $\mathcal{C}$ will return the secret value r_i of that user to $\mathcal{A}_1$.
- *Public-Key-Replacement*: For any user with identity ID and public key pk, $\mathcal{A}$ can set a new public key pk', and then $\mathcal{C}$ replaces pk with pk'.
- *Sign query*: $\mathcal{A}_1$ can query for the signature S_i corresponding to a message m_i, a user with identity ID_i and public key pk_i. $\mathcal{C}$ will generate S_i, and return it to $\mathcal{A}_1$.

Forgery: $\mathcal{A}_1$ outputs a tuple $(S^*, m^*, \mathrm{ID}^*, \mathrm{pk}^*)$ where S^* is a signature for the message m^* corresponding to the identity ID^* and public key pk^*.

$\mathcal{A}_1$ wins the game if and only if the following conditions hold.

- The forged signature S^* is valid for the message m^* when verified using *params*, identity ID^* and public key pk^*.
- The private key (both secret value and partial private key) of ID^* and the signature S^* have never been queried.

Game II: $\mathcal{A}_2$ interacts with challenger $\mathcal{C}$.

Setup: $\mathcal{C}$ performs Setup by inputing a security parameter to obtain the master-key and the system parameter *params*. $\mathcal{C}$ sends *params* and the master-key to $\mathcal{A}_2$.

Attack: $\mathcal{A}_2$ can adaptively perform the following polynomially bounded queries.

- *Public-Key query*: $\mathcal{A}_2$ can query for the public key of any user with identity ID_i. $\mathcal{C}$ will return the public key pk_i of that user.
- *Secret-Value query*: $\mathcal{A}_2$ can query for the secret value of any user with identity ID_i. $\mathcal{C}$ will return the secret value r_i of that user to $\mathcal{A}_2$.
- *Sign query*: $\mathcal{A}_2$ can query for the signature S_i corresponding to a message m_i, a user with identity ID_i and public key pk_i. $\mathcal{C}$ will generate S_i, and return it to $\mathcal{A}_2$.

Forgery: $\mathcal{A}_2$ outputs a tuple $(S^*, m^*, \mathrm{ID}^*, \mathrm{pk}^*)$ where S^* is a signature for the message m^* corresponding to the identity ID^* and public key pk^*.

$\mathcal{A}_2$ wins the game if and only if the following conditions hold.

- The forged signature S^* is valid for the message m^* when verified using *params*, identity ID^* and public key pk^*.
- The secret value of ID^* and the signature S^* have never been queried.

3.2 Existential unforgeability for insiders

A legitimate user in a CLS scheme can try to forge a key pair by attacking the Partial-Private-Key-Extract algorithm. We use the following game to model this attack.

Game III: $\mathcal{A}_3$ interacts with Challenger $\mathcal{C}$. $\mathcal{A}_3$ is a legitimate but malicious user who wants to obtain more then one key pair out of his identity.

Setup: The challenger $\mathcal{C}$ runs Setup to generate the system parameters and sends them to $\mathcal{A}_3$.

Attack: $\mathcal{A}_3$ can query for (1) the public key $\mathrm{pk}_{\mathrm{ID}}$ of any user with identity ID; (2) the secret value r_{ID} of ID; and (3) the partial-private-key of ID. $\mathcal{C}$ will return the partial-private-key D_{ID}, the public key $\mathrm{pk}_{\mathrm{ID}}$, or the secret value r_{ID} to $\mathcal{A}_3$.

Key-Forgery: $\mathcal{A}_3$ outputs a key pair $\left(\mathrm{pk}'_{\mathrm{ID}^*}, D_{\mathrm{ID}^*}\right)$ for identity ID^*.

$\mathcal{A}$ wins this game if and only if the following conditions hold.

1. The key pair $\left(\mathrm{pk}'_{\mathrm{ID}^*}, D_{\mathrm{ID}^*}\right)$ is valid.
2. The partial private key D_{ID^*}, corresponding to identity ID^* and public key $\mathrm{pk}'_{\mathrm{ID}^*}$, has never been queried.
3. $\mathrm{pk}'_{\mathrm{ID}^*} \neq \mathrm{pk}_{\mathrm{ID}^*}$ where $\mathrm{pk}_{\mathrm{ID}^*}$ has been queried before.

Thus, if $\mathcal{A}_3$ wins *Game III*, then $\mathcal{A}_3$ can generate another key pair without formally interacting with the KGC.

Most schemes are shown to achieve level 3 security based on the above three games. However, restricting a user to have a unique key pair does not prevent him from forging a signature which can be verified by a different public key without knowing the corresponding private key. For example, Fan *et al.*'s scheme does not achieve level 3 security despite of the fact that the scheme is shown to be secure under certain security models (Game I, II and III) [24]. Assume Alice is an adversary who is also a user with identity $\mathrm{ID}_{\mathcal{A}}$. The attack goes as follows:

1. Alice sets her secret value $r_{\mathcal{A}}$ and two public key components $\mathrm{pk}_{\mathcal{A}}$ and $\mathrm{pk}'_{\mathcal{A}}$ as in Section 2.4.
2. Alice obtains her partial-private-key $D_{\mathcal{A}}$ from the KGC, then she sets her full private key $\mathrm{sk}_{\mathcal{A}} = \left(D_{\mathcal{A}}, r_{\mathcal{A}}\right)$.
3. Alice randomly chooses $r^* \in \mathbb{Z}_q^*$, and then replaces her second part of public key, $\mathrm{pk}'_{\mathcal{A}}$, with $\mathrm{pk}'^*_{\mathcal{A}}$ where $\mathrm{pk}'^*_{\mathcal{A}} = r_{\mathcal{A}}\left(P_{\mathrm{pub}} + H_1(\mathrm{ID}_{\mathcal{A}})P_2\right) + r^*\left(P_{\mathrm{pub}} + H_1(\mathrm{ID}_{\mathcal{A}})P_2\right) + r^* H_1(\mathrm{ID}_{\mathcal{A}}||\mathrm{pk}_{\mathcal{A}})P_2$.
4. For any message m, Alice computes $h = H2\left(m, \mathrm{pk}_{\mathcal{A}}\right)$, and then she generates a signature S by computing

$$S = \frac{1}{r^* + r_{\mathcal{A}} + h} D_{\mathcal{A}}.$$

The signature S is valid if we use the public key $\left(\mathrm{pk}_{\mathcal{A}}, \mathrm{pk}'^*_{\mathcal{A}}\right)$ to verify it.

Therefore Alice can generate many $\mathrm{pk}'^*_{\mathcal{A}}$ corresponding to $\mathrm{pk}_{\mathcal{A}}$ and generate signatures that are valid when verifying them with the forged public key $\left(\mathrm{pk}_{\mathcal{A}}, \mathrm{pk}'^*_{\mathcal{A}}\right)$. Hence, this scheme does not achieve level 3 security despite of the fact that the scheme is shown to be secure under the security models Game I, II and III. Indeed, a cryptographic scheme can be provably secure under a security model, but it may still suffer from other attacks if the security model does not include these attacks. Observing Game III, the goal of the adversary is to forge another key pair. However, in our attack, the adversary aims to generate a valid signature only. Therefore we believe the above three security models (proposed in [16] and adopted in [18]) are inappropriate for proving level 3 security.

In light of the above attack, we present a new security game, *Game IV*, to model a malicious signer aiming to forge a signature and output the corresponding replaced public key.

Game IV: An adversary $\mathcal{A}_4$ interacts with a challenger $\mathcal{C}$ where $\mathcal{A}_4$ is a legitimate but malicious user.

Setup: The challenger $\mathcal{C}$ runs Setup to generate the system parameters and sends them to $\mathcal{A}_4$.

© The Institution of Engineering and Technology 2013

doi: 10.1049/iet-ifs.2012.0134

Attack: $\mathcal{A}_4$ can query for (1) the public key $\mathrm{pk_{ID}}$ of any user with identity ID and the secret value r_{ID}, (2) the partial-private-key D_{ID} corresponding to the identity ID and public key $\mathrm{pk_{ID}}$, and (3) the signature of a message m corresponding to a user with identity ID and public key $\mathrm{pk_{ID}}$.

Sign-PK-*Forgery*: $\mathcal{A}_4$ outputs a signature S^* of a message m^* and a replaced public key $\mathrm{pk}'_{\mathrm{ID}^*}$ of ID^*. $\mathcal{A}_4$ wins this game if and only if the following conditions hold.

1. The forged signature S^* is valid for the message m^* when verified using *params*, identity ID^* and public key $\mathrm{pk}'_{\mathrm{ID}^*}$.
2. The signature S^* and the partial-private-key D'_{ID^*} corresponding to the identity ID^* and public key $\mathrm{pk}'_{\mathrm{ID}^*}$ have never been queried before.
3. $\mathrm{pk}'_{\mathrm{ID}^*} \neq \mathrm{pk}_{\mathrm{ID}^*}$ where $\mathrm{pk}_{\mathrm{ID}^*}$ is obtained by queries during the *Attack* phase.

We note that this game is different from previous security models in that the goal of the attacker is to forge a signature which is valid when verified using a replaced public key. To summarise, *Game I* and *Game II* are used mainly to prove existential unforgeability, and *Game III* and *Game IV* are used mainly to prove that a CLS scheme achieves level 3 security.

Theorem 1: A certificateless signature scheme provides strong non-repudiation if no adversary, running polynomially probabilistic time algorithms, has non-negligible probability to win any of the above four games.

Proof: We need to show that all signatures that can be verified based on some identity must be generated by the user with that identity and the user can provide proof to accuse the KGC for wrongdoing. If no type 1 adversary has non-negligible probability to win *Game I* then the certificateless signature scheme is existentially unforgeable against the chosen message attacks. That is, all valid signatures are either generated by the users themselves or the KGC. If no Type 2 adversary has non-negligible probability to win *Game II* then the KGC in the certificateless signature scheme cannot forge signatures that are verifiable using the identity and the public key of a user. Furthermore, if no user can have non-negligible probability to win *Game III* or *Game IV* then any user in the certificateless signature scheme can only have one valid key pair and generate signatures of messages that can only be verified with this unique public key. Therefore all forged signatures that are verifiable using some identity ID must associated with a different public key. And these signatures must generated by the malicious KGC since only he has the master key. However, when this happened, the user with identity ID can detect and easily prove that the public key and the signatures are forgeries by presenting his genuine public key. In addition, the user does not have other public keys, so he cannot generate a forged signature to incriminate the KGC. Therefore this CLS scheme provides strong non-repudiation. □

4 Certificateless signature scheme for strong non-repudiation

In this section, we present a new CLS scheme, called CLS-SNR, and show that it provides strong non-repudiation based on the above security models.

4.1 The proposed scheme (CLS-SNR)

The three phases of CLS-SNR are as follows:

InitSetup:

- *Setup*: The KGC determines a bilinear map $\hat{e}:\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ where $\mathbb{G}_1$ is a cyclic additive group of prime order q with a generator P, $\mathbb{G}_2$ is a cyclic multiplicative group of the same order, and three hash functions $H_1:\{0,1\}^* \to \mathbb{Z}_q^*$, $H_2:\{0,1\}^* \to \mathbb{Z}_q^*$, and $H_3:\{0,1\}^* \to \mathbb{G}_1$. Then it randomly chooses $s_1, s_2 \in \mathbb{Z}_q^*(s_1 \neq s_2)$ as *master-key*, and then sets (P_{pub1}, P_{pub2}) as the master-public-key where $P_{\mathrm{pub1}} = s_1P$ and $P_{\mathrm{pub2}} = s_2P$. Finally, it publishes the system parameter $params = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, q, P, P_{\mathrm{pub1}}, P_{\mathrm{pub2}}, H_1, H_2, H_3 \rangle$.
- *Set-Secret-Value*: A user with identity ID, sets a random value $r_{\mathrm{ID}} \in \mathbb{Z}_q^*$ as his secret value.
- *Set-Public-Key*: A user, based on *params*, his secret value r and identity ID, computes his public key $\mathrm{pk_{ID}} = r_{\mathrm{ID}}T$ where $T = P_{\mathrm{pub1}} + H_1(\mathrm{ID})P$.
- *Partial-Private-Key-Extract*: The KGC, based on *params*, *master-key*, user's identity ID, and public key $\mathrm{pk_{ID}}$, computes and returns a partial-private-key $D_{\mathrm{ID}} = s_2/(s_1 + H_1(\mathrm{ID}))H_3(\mathrm{ID}, \mathrm{pk_{ID}})$ to the user with identity ID.
- *Set-Private-Key*: A user computes his full private key $\mathrm{sk_{ID}} = (D_{\mathrm{ID}}, r_{\mathrm{ID}})$ where D_{ID} is the user's partial-private-key obtained from the KGC and r_{ID} is his secret value.

CL-Sign: To generate a signature for a message m, a signer with private key sk_{ID}, computes $S = 1/(r_{\mathrm{ID}} + h)D_{\mathrm{ID}}$ as the signature for the message m where $h = H_2(m, \mathrm{pk_{ID}})$.

CL-Verify: To verify a signature S of a message m generated by a user with identity ID and public key $\mathrm{pk_{ID}}$, a verifier takes *params*, the public key $\mathrm{pk_{ID}}$, the message m, the user's identity ID, and the signature S as inputs, then computes $h = H_2(m, \mathrm{pk_{ID}})$, $T = P_{\mathrm{pub1}} + H_1(\mathrm{ID})P$ and accepts this signature if and only if $\hat{e}(S, \mathrm{pk_{ID}} + hT) = \hat{e}(H_3(\mathrm{ID}, \mathrm{pk_{ID}}), P_{\mathrm{pub2}})$.

The correctness of the signature verification follows from the equation below

$$\begin{aligned}
\hat{e}(S, \mathrm{pk_{ID}} + hT) &= \hat{e}(S, r_{\mathrm{ID}}T + hT) \\
&= \hat{e}\left(\frac{1}{r_{\mathrm{ID}} + h}D_{\mathrm{ID}}, (r_{\mathrm{ID}} + h)T\right) \\
&= \hat{e}(D_{\mathrm{ID}}, T) \\
&= \hat{e}\left(\frac{s_2}{(s_1 + H_1(ID))}H_3(\mathrm{ID}, \mathrm{pk_{ID}}), P_{\mathrm{pub1}} + H_1(\mathrm{ID})P\right) \\
&= \hat{e}\left(\frac{s_2}{(s_1 + H_1(\mathrm{ID}))}H_3(\mathrm{ID}, \mathrm{pk_{ID}}), s_1P + H_1(\mathrm{ID})P\right) \\
&= \hat{e}(s_2H_3(\mathrm{ID}, \mathrm{pk_{ID}}), P) \\
&= \hat{e}(H_3(\mathrm{ID}, \mathrm{pk_{ID}}), s_2P) \\
&= \hat{e}(H_3(\mathrm{ID}, \mathrm{pk_{ID}}), P_{\mathrm{pub2}})
\end{aligned}$$

4.2 Security analysis

We give the formal proofs of CLS-SNR being existentially unforgeable against the chosen message attacks of Type 1

© The Institution of Engineering and Technology 2013

and 2 adversaries in Appendix A. We summarise the results in the following lemma.

Lemma 1: Assume that the k-CAA problem and the modified k-CAA problem are intractable. Then no adversary, running polynomially probabilistic time algorithms, has non-negligible probability to win Game I or Game II in CLS-SNR.

The remainder of this section will be devoted to the security analysis of a type 3 adversary. The following lemma shows that a user with identity ID can only have one valid key pair ($\mathrm{pk}_{\mathrm{ID}}$, $\mathrm{sk}_{\mathrm{ID}}$) associated with this identity in CLS-SNR.

Lemma 2: Assume that the Computational Diffie–Hellman problem is intractable. Then no adversary, running polynomially probabilistic time algorithms, has non-negligible probability to win *Game III*.

Proof: Suppose an adversary $\mathcal{A}$ in *Game III* is able to existentially forge a key pair in CLS-SNR. Then we can construct a probabilistic polynomial time algorithm $\mathcal{C}$ which solves the CDH problem by running $\mathcal{A}$ and supplying correct responses to $\mathcal{A}$'s queries as defined in *Game III*.

Assume the challenger $\mathcal{C}$ is given a random instance of the CDH problem: Given P, aP, $bP \in \mathbb{G}_1$, compute abP where $a, b \in \mathbb{Z}_q^*$ and are unknown. Now $\mathcal{C}$ and $\mathcal{A}$ play the roles of the challenger and adversary, respectively, in *Game III*. Eventually, $\mathcal{C}$ may base on $\mathcal{A}$'s forgery to solve this instance of the CDH problem. Assume $\mathcal{A}$ will make at most q_{H_3} H_3 queries and q_{pk} public key queries.

Setup: $\mathcal{C}$ randomly chooses $s_1 \in \mathbb{Z}_q^*$ and sets $P_{\mathrm{pub1}} = s_1P$ and $P_{\mathrm{pub2}} = aP$ from the CDH instance. It then gives *params* to $\mathcal{A}$ where params $= (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_{\mathrm{pub1}}, P_{\mathrm{pub2}}, P, H_1, H_2, H_3)$. $\mathcal{C}$ will maintain three initially empty tables: the H_1-table, the H_3-table, and the K-table containing tuples of $\langle \mathrm{ID}_j, w_j\rangle$, $\langle \mathrm{ID}_j, \mathrm{pk}_{\mathrm{ID}_j}, Q_j, \alpha_j, c_j\rangle$, and $\langle \mathrm{ID}_j, \mathrm{pk}_{\mathrm{ID}_j}, D_{\mathrm{ID}_j}, r_{\mathrm{ID}_j}\rangle$, respectively.

Attack: $\mathcal{A}$ can adaptively perform the following polynomially bounded queries.

- H_1 *queries*: $\mathcal{A}$ can query for the value of $H_1(\mathrm{ID}_i)$ for any user's identity ID_i. $\mathcal{C}$ works as follows:

  - If $H_1(\mathrm{ID}_i)$ has been queried before then $\mathcal{C}$ returns w_i obtained from the H_1-table as $H_1(\mathrm{ID}_i)$.
  - Otherwise, $\mathcal{C}$ randomly chooses $w_i \in \mathbb{Z}_q^*$. $\mathcal{C}$ returns w_i to $\mathcal{A}$ and inserts the pair $\langle \mathrm{ID}_j, w_j\rangle$ into the H_1-table.

- H_3 *queries*: $\mathcal{A}$ can query for the value of $H_3(\mathrm{ID}_i, \mathrm{pk}_{\mathrm{ID}_i})$ for any pair $(\mathrm{ID}_i, \mathrm{pk}_{\mathrm{ID}_i})$. $\mathcal{C}$ works as follows:

  - If $H_3(\mathrm{ID}_i, \mathrm{pk}_{\mathrm{ID}_i})$ has been queried before then $\mathcal{C}$ returns Q_i obtained from the H_3-table to $\mathcal{A}$.
  - Otherwise, $\mathcal{C}$ randomly sets c_i to 0 or 1 with $\Pr[c_i = 0] = 1/q_{H_3}$. If $c_i = 0$ then $\mathcal{C}$ randomly chooses $\alpha_i \in \mathbb{Z}_q^*$ and computes $Q_i = \alpha_i(bP)$. Otherwise, $\mathcal{C}$ chooses $\alpha_i \in \mathbb{Z}_q^*$ and computes $Q_i = \alpha_iP$. $\mathcal{C}$ returns Q_i to $\mathcal{A}$ and inserts the tuple $\langle \mathrm{ID}_i, \mathrm{pk}_{\mathrm{ID}_i}, Q_i, \alpha_i, c_i\rangle$ into the H_3-table.

- *Public-Key queries*: $\mathcal{A}$ can query for the public key associated with identity ID_i. If ID_i has been queried before then $\mathcal{C}$ returns the public key $\mathrm{pk}_{\mathrm{ID}}$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ randomly chooses $r_{\mathrm{ID}_i} \in \mathbb{Z}_q^*$ and then generate a public key $\mathrm{pk}_{\mathrm{ID}_i} = r_{\mathrm{ID}_i}(P_{\mathrm{pub1}} + w_iP)$ where w_i is from the H_1-table. $\mathcal{C}$ sends $\mathrm{pk}_{\mathrm{ID}_i}$ to $\mathcal{A}$ then inserts the tuple $\langle \mathrm{ID}_i, \mathrm{pk}_{\mathrm{ID}_i}, \bigstar, r_{\mathrm{ID}_i}\rangle$ into the K-table. (Hereafter, $\bigstar$ means an empty space.)
- *Partial-Private-Key queries*: $\mathcal{A}$ can query for the partial-private-key associated with any identity ID_i and public key $\mathrm{pk}_{\mathrm{ID}_i}$.

  - If $(\mathrm{ID}_i, \mathrm{pk}_{\mathrm{ID}_i})$ corresponds to $c_i = 0$ in the H_3-table then $\mathcal{C}$ aborts and terminates.
  - Otherwise, $\mathcal{C}$ goes back to perform H_1, H_3 queries and bases on the H_1-table, the H_3-table, and the K-table to determine and return the partial-private-key D_{ID_i} to $\mathcal{A}$ where $D_{\mathrm{ID}_i} = \alpha_i/(s_1 + w_i)(aP)$. Note that s_1 is known, w_i is taken from the H_1-table, and α_i is taken from the H_3-table.

Finally, $\mathcal{C}$ sends D_{ID_i} to $\mathcal{A}$ and then inserts the tuple $\langle \mathrm{ID}_i, \bigstar, D_{\mathrm{ID}_i}, \bigstar\rangle$ into the K-table.

Key-Forgery: $\mathcal{A}$ generates a partial-private-key D_{ID^*} and a public key $\mathrm{pk}'_{\mathrm{ID}^*}$. If $(\mathrm{ID}^*, \mathrm{pk}'_{\mathrm{ID}^*})$ corresponds to $c^* \neq 0$ in the H_3-table and $\mathrm{pk}'_{\mathrm{ID}^*} = \mathrm{pk}_{\mathrm{ID}^*}$ in the K-table then $\mathcal{C}$ aborts and terminates. When $\mathcal{A}$ successfully forges a valid partial-private-key D_{ID^*} where $D_{\mathrm{ID}^*} = \alpha^*/(s_1 + w^*)(abP)$, $\mathcal{C}$ is able to use $\mathcal{A}$'s forgery to compute

$$abP = (s_1 + w^*)\alpha^{*-1}D_{\mathrm{ID}^*}$$

Therefore $\mathcal{C}$ can successfully solve the CDH problem if the following two events occurred simultaneously:

- ε_1: $\mathcal{C}$ does not abort in any $\mathcal{A}$'s queries during the *Attack* phase. That is, $\mathcal{C}$ does not abort in any $\mathcal{A}$'s partial-private-key queries.
- ε_2: $\mathcal{C}$ does not abort in *Key-Forgery*.

Claim 1. The probability that $\mathcal{C}$ does not abort in any $\mathcal{A}$'s partial-private-key queries is at least $1/e$ where e is the base of the natural logarithm.

Proof: The probability that $c_i = 0$ in each of the partial-private-key query is $1/q_{H_3}$. Thus, $\Pr[\mathcal{E}_1] = (1 - 1/(q_{H_3}))^{q_{H_3}} \geq 1/e$. □

Claim 2. The probability that $\mathcal{C}$ does not abort in *Forgery* is $1/(q_{H_3})((q-1)/q)$.

Proof: Assume $\mathcal{A}$ outputs a forged key pair $(D_{\mathrm{ID}^*}, \mathrm{pk}'_{\mathrm{ID}^*})$. Then $\mathcal{C}$ does not abort if $(\mathrm{ID}^*, \mathrm{pk}'_{\mathrm{ID}^*})$ corresponds $c^* = 0$ and $\mathrm{pk}'_{\mathrm{ID}^*} \neq \mathrm{pk}_{\mathrm{ID}^*}$. The probability of that $(\mathrm{ID}^*, \mathrm{pk}'_{\mathrm{ID}^*})$ corresponds to $c^* = 0$ is $1/q_{H_3}$. The probability of $\mathrm{pk}'_{\mathrm{ID}^*} \neq \mathrm{pk}_{\mathrm{ID}^*}$ is $(q-1)/q$. Hence, $\Pr[\mathcal{E}_2] = 1/(q_{H_3})(q-1)/q$. □

Let $(q-1)/q = 1/\gamma$. By Claim 1 and Claim 2, $\mathcal{C}$ does not abort with probability $\Pr[\mathcal{E}_1]\Pr[\mathcal{E}_2] \geq 1/(eq_{H_3}\gamma)$. Let the running time of $\mathcal{A}$ be t. Then the running time of $\mathcal{C}$ is $t' < t + (q_{H_3} + q_{\mathrm{pk}})t_{\mathrm{sm}}$, where t_{sm} is the running time of computing a scalar multiplication in $\mathbb{G}_1$. Hence if $\mathcal{A}$ runs in polynomial time so does $\mathcal{C}$. Therefore $\mathcal{C}$ solves the CDH problem with probability $\varepsilon' \geq 1/(eq_{H_3}\gamma)\varepsilon$ where ε is the probability of $\mathcal{A}$ to win *Game III*.

Since CDH problem is assumed to be intractable, ε' is negligible. Hence, ε is negligible. That is, there is no adversary, running polynomially probabilistic time algorithms, has non-negligible probability to win *Game III* in CLS-SNR. □

© The Institution of Engineering and Technology 2013

doi: 10.1049/iet-ifs.2012.0134

In the following, we show that a user with identity ID and key pair (pk_{ID}, sk_{ID}) can only generate signatures which are valid when verified using *params*, identity ID and public key pk_{ID} in CLS-SNR.

Lemma 3: Assume that the Computational Diffie–Hellman problem is intractable. Then no adversary, running polynomially probabilistic time algorithms, has non-negligible probability to win *Game IV*.

Proof: Suppose an adversary $\mathcal{A}$ in *Game IV* is able to existentially forge a signature which is valid when verified using a replaced public key in CLS-SNR. Then we can construct a probabilistic polynomial time algorithm $\mathcal{C}$ which solves the CDH problem by running $\mathcal{A}$ and supplying correct responses to $\mathcal{A}$'s queries as defined in *Game IV*.

Assume the challenger $\mathcal{C}$ is given a random instance of the CDH problem: Given P, aP, $bP \in \mathbb{G}_1$, compute abP, where $a, b \in \mathbb{Z}_q^*$ and are unknown. Now $\mathcal{C}$ and $\mathcal{A}$ play the roles of the challenger and adversary, respectively, in *Game IV*. Eventually, $\mathcal{C}$ may base on $\mathcal{A}$'s forgery to solve this instance of the CDH problem. Assume $\mathcal{A}$ will make at most q_{H_3} H_3 queries, q_{pk} public key queries, and q_S sign queries.

Setup: $\mathcal{C}$ randomly chooses $s_1 \in \mathbb{Z}_q^*$ and sets $P_{pub1} = s_1P$ and $P_{pub2} = aP$ (taking from the CDH instance). It then gives params to $\mathcal{A}$ where params $= (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_{pub1}, P_{pub2}, P, H_1, H_2, H_3)$. $\mathcal{C}$ will maintain four initially empty tables: the H_1-table, the H_2-table, the H_3-table and the K-table containing tuples of $\langle ID_j, w_j\rangle$, $\langle m_j, pk_{ID_j}, h_j\rangle$, $\langle ID_j, pk_{ID_j}, Q_j, \alpha_j, c_j\rangle$ and $\langle ID_j, pk_{ID_j}, D_{ID_j}, r_{ID_j}\rangle$, respectively.

Attack: $\mathcal{A}$ can adaptively perform the following polynomially bounded queries.

• H_1 *queries*: $\mathcal{A}$ can query for the value of $H_1(ID_i)$ for any user's identity ID_i. $\mathcal{C}$ works as follows:

• If $H_1(ID_i)$ has been queried before then $\mathcal{C}$ returns w_i obtained from the H_1-table as $H_1(ID_i)$.
• Otherwise, $\mathcal{C}$ randomly chooses $w_i \in \mathbb{Z}_q^*$. $\mathcal{C}$ returns w_i to $\mathcal{A}$ and inserts the pair $\langle ID_j, w_j\rangle$ into the H_1-table.

• H_2 *queries*: $\mathcal{A}$ can query for the value of $H_2(m_i, pk_{ID_i})$ for any public key pk_{ID_i} and message m_i. $\mathcal{C}$ works as follows:

• If $H_2(m_i, pk_{ID_i})$ has been queried before then $\mathcal{C}$ returns h_i obtained from the H_2-table as $H_2(m_i, pk_{ID_i})$.
• Otherwise, $\mathcal{C}$ randomly chooses $h_i \in \mathbb{Z}_q^*$. $\mathcal{C}$ returns h_i to $\mathcal{A}$ and inserts the tuple $\langle m_i, pk_{ID_i}, h_i\rangle$ into the H_2-table.

• H_3 *queries*: $\mathcal{A}$ can query for the value of $H_3(ID_i, pk_{ID_i})$ for any pair (ID_i, pk_{ID_i}). $\mathcal{C}$ works as follows:

• If $H_3(ID_i, pk_{ID_i})$ has been queried before then $\mathcal{C}$ returns Q_i obtained from the H_3-table to $\mathcal{A}$.
• Otherwise, $\mathcal{C}$ randomly sets c_i to 0 or 1 with $\Pr[c_i = 0] = 1/q_{H_3}$. If $c_i = 0$ then $\mathcal{C}$ randomly chooses $\alpha_i \in \mathbb{Z}_q^*$ and computes $Q_i = \alpha_i(bP)$. Otherwise, chooses $\alpha_i \in \mathbb{Z}_q^*$ and computes $Q_i = \alpha_iP$. $\mathcal{C}$ returns Q_i to $\mathcal{A}$ and inserts the tuple $\langle ID_i, pk_{ID_i}, Q_i, \alpha_i, c_i\rangle$ into the H_3-table.

• *Public-Key queries*: $\mathcal{A}$ can query for the public key associated with identity ID_i. If ID_i has been queried before then $\mathcal{C}$ returns the public key pk_{ID} to $\mathcal{A}$. Otherwise, $\mathcal{C}$ randomly chooses $r_{ID_i} \in \mathbb{Z}_q^*$ and generate a public key $pk_{ID_i} = r_{ID_i}(P_{pub1} + w_iP)$ where w_i is obtained from the H_1-table. $\mathcal{C}$ sends pk_{ID_i} to $\mathcal{A}$ and inserts the tuple $\langle ID_i, pk_{ID_i}, \bigstar, r_{ID_i}\rangle$ into the K-table.
• *Partial-Private-Key queries*: $\mathcal{A}$ can query for the partial-private-key associated with any identity ID_i and public key pk_{ID_i}.

• If (ID_i, pk_{ID_i}) corresponds to $c_i = 0$ in the H_3-table then $\mathcal{C}$ aborts and terminates.
• Otherwise, $\mathcal{C}$ goes back to perform H_1, H_3 queries and bases on H_1-table, the- H_3 table, and the K-table to determine and return the partial-private-key D_{ID_i} to $\mathcal{A}$ where $D_{ID_i} = \alpha_i/(s_1 + w_i)(aP)$. Note that s_1 is known, w_i is taken from the H_1-table, and α_i is taken from the H_3-table.

Finally, $\mathcal{C}$ sends D_{ID_i} to $\mathcal{A}$ and then inserts the tuple $\langle ID_i, \bigstar, D_{ID_i}, \bigstar\rangle$ into the K-table.

• *Sign queries*: $\mathcal{A}$ can query for the signature of a message m corresponding to a user with identity ID_i and public key pk_{ID_i}.

• If (ID_i, pk_{ID_i}) corresponds to $c_i = 0$ in the H_3-table then $\mathcal{C}$ aborts and terminates.
• Otherwise, $\mathcal{C}$ will base on H_1-table, the H_3-table and the K-table to return the signature S to $\mathcal{A}$ where S is computed as $S = \alpha_i/((r_{ID_i} + h_i)(s_1 + w_i))(aP)$.

Sign-PK-Forgery: $\mathcal{A}$ generates a forged signature S^* of (m^*, ID^*, pk'_{ID^*}) and outputs one replaced public key pk'_{ID^*}. If the identity (ID^*, pk'_{ID^*}) corresponds to $c^* \neq 0$ in the H_1-table or $pk'_{ID^*} = pk_{ID^*}$, $\mathcal{C}$ aborts and terminates. If pk_{ID^*} has not been queried, $\mathcal{A}$ will receive a valid pk_{ID^*}.

When $\mathcal{A}$ successfully forges a valid signature S^* of (m^*, ID^*, pk'_{ID^*}) and outputs one corresponding public key pk'_{ID^*} where $S^* = \alpha^*/((r'_{ID^*} + h^*)(s_1 + w^*))(abP)$. Note that h^* corresponding (m^*, pk'_{ID^*}) is taken from the H_2-table, r'_{ID^*} is taken from the K-table. Finally, $\mathcal{C}$ is able to use $\mathcal{A}$'s forgery to compute

$$abP = (r'_{ID^*} + h^*)(s_1 + w^*)\alpha^{*-1}S^*$$

Therefore $\mathcal{C}$ can successfully solve the CDH problem if the following two events occurred simultaneously:

• ε_1: $\mathcal{C}$ does not abort in any $\mathcal{A}$'s query.
• ε_2: $\mathcal{C}$ does not abort in *Sign-PK-Forgery*.

Claim 1. The probability that $\mathcal{C}$ does not abort in any $\mathcal{A}$'s queries in *Attack* is at least $1/e^2$.

Proof: The probability that $c_i = 0$ in each of the H_3 query or the partial-private-key query is $1/q_{H_3}$. Thus, $\Pr[\mathcal{E}_1] = ((1 - 1/(q_{H_3}))^{q_{H_3}})^2 \geq 1/e^2$.
Claim 2. The probability that $\mathcal{C}$ does not abort in *Forgery* is $1/(q_{H_3})(q-1)/q$.

Proof: Assume $\mathcal{A}$ outputs a valid signature S^* and a public key pk_{ID^*}. Then $\mathcal{C}$ does not abort if (ID^*, pk'_{ID^*}) corresponds to $c^* = 0$ and $pk'_{ID^*} \neq pk_{ID^*}$. The probability of that (ID^*, pk'_{ID^*}) corresponds to $c^* = 0$ is $1/(q_{H_3})$. The

© The Institution of Engineering and Technology 2013

Table 1 Comparisons among various schemes

	IBS	CBS	AP-CLS [2]	DW-CLS [4]	FHH-CLS [18]	Proposed
trust level	1	3	2	2	2	3
game I	N/A	N/A	◯	◯	◯	◯
game II	N/A	N/A	◯	◯	◯	◯
game III	N/A	N/A	×	×	◯	◯
game IV	N/A	N/A	×	×	×	◯

probability of $pk'_{ID^*} \neq pk_{ID^*}$ is $(q-1)/q$. Hence, $\Pr[\mathcal{E}_2] = 1/\left(q_{H_3}\right)(q-1)/q$. □

Let $(q-1)/q = 1/\gamma$. By claim 1 and claim 2, $\mathcal{C}$ does not abort with probability $\varepsilon' = \Pr[\mathcal{E}_1]\Pr[\mathcal{E}_2]\varepsilon \geq 1/(e^2 q_{H_3}\gamma)\varepsilon$ where ε is the probability of $\mathcal{A}$ to win *Game IV*. Let the running time of $\mathcal{A}$ be t. Then the running time of $\mathcal{C}$ is $t' < t + (q_{H_3} + q_{pk} + q_S)t_{sm} + q_S t_{inv}$, where t_{sm} is the running time of computing a scalar multiplication in $\mathbb{G}_1$, and t_{inv} is the running time of performing an inversion computation in $\mathbb{G}_1$. Hence if $\mathcal{A}$ runs in polynomial time so does $\mathcal{C}$. Therefore $\mathcal{C}$ solves the CDH problem with probability $\varepsilon' \geq 1/(e^2 q_{H_3}\gamma)\varepsilon$.

Since the CDH problem is assumed to be intractable, ε' is negligible. Hence, ε is negligible. That is, there is no adversary, running polynomially probabilistic time algorithms, has non-negligible probability to win *Game IV* in CLS-SNR. □

By Lemmas 1–3, and Theorem 1, we have the following:

Theorem 2: CLS-SNR provides strong non-repudiation.

4.3 Discussions and comparisons

Public key cryptography requires an entity to help to authenticate public keys. The certificates and public key infrastructure, created by a TTP, are commonly adopted in traditional PKC. The public keys of ID-based Public Key Cryptography are strings that identify the users [25]. The ID-PKC requires a trusted authority to generate secret keys and send them to users securely. Owing to key escrow, identity-based signature (IBS) schemes can only offer weak non-repudiation services. Like ID-PKC, CL-PKC does not require the use of digital certificates to guarantee the authenticity of public keys. Most CLS schemes achieve level 2 security based on Game I and Game II. However, for security level 3, CLS schemes must be proved based on Game I, II, III and IV. Table 1 summarises the results of comparisons among traditional IBS, certificate-based signature [26], and several CLS schemes [2, 4, 18].

For non-repudiation, ISO/IEC 13888-1 [20] defines several types of specific non-repudiation services which are listed in Appendix B. Our scheme can truly provide non-repudiation of origin since it is a signature scheme where a user only has one key pair. Moreover, it also can provide non-repudiation of submission. As the scenario of the malicious KGC mentioned in Section 2.1, the user can give significant evidence (a signature generated by him) to show that signature is generated by the malicious KGC.

5 Conclusions

Public key cryptography has found many applications in our modern society. To guarantee the authenticity of public keys, we need a TTP. In 1991, Girault defined three trust levels for a TTP. The higher the trusted level of the TTP is, the higher the security level of the cryptographic scheme is. In this paper, we have considered in depth the security requirements of CLS schemes and, based on cryptanalyzing Fan *et al.*'s scheme, shown that previous models are inappropriate for achieving the desired level of security. More precisely, to achieve level 3 security (i.e. provide strong non-repudiation), a digital signature scheme should be able to provide proof to accuse the KGC for wrongdoing. We have presented a complete set of security models for this purpose. Furthermore, we have also proposed a new CLS scheme, and shown that it can provide strong non-repudiation.

6 Acknowledgments

The authors thank the reviewers for their comments that helped improve the paper. This work was partially supported by the National Science Council of the Republic of China under contract Nos. NSC100-2221-E-468-014 and NSC101-2221-E-005-083.

7 References

1 Shamir, A.: 'Identity based cryptosystems and signature schemes'. Proc. CRYPTO'84, (*LNCS*, **196**), 1984, pp. 47–53
2 Al-Riyami, S., Paterson, K.: 'Certificateless public key cryptography'. Proc. ASIACRYPT'03, 2003 (*LNCS*, **2894**), pp. 452–473
3 Girault, M.: 'Self-certified Public Keys'. Proc. EUROCRYPTO'91, 1991, (*LNCS*, **547**) pp. 490–497
4 Du, H., Wen, Q.: 'Efficient and provably-secure certificateless short signature scheme from bilinear pairings', *Comput Stand Interfaces*, 2009, **31**, pp. 390–394
5 Au, M.H., Chen, J., Liu, J.K., Mu, Y., Wong, D.S., Yang, G.: 'Malicious KGC attack in certificateless cryptography'. Proc. 2nd ACM Symp. on Information, Computer and Communications Security, 2007, pp. 302–311
6 Chen, Y.C., Liu, C.L., Horng, G., Chen, K.C.: 'A provably secure certificateless proxy signature scheme', *Int. J. Innov. Comput Inf. Control*, 2011, **7**, (9), pp. 5557–5569
7 Choi, K., Park, J., Hwang, J., Lee, D.: 'Efficient certificateless signature schemes'. Proc. ACNS 2007, 2007 (*LNCS*, **4521**), pp. 443–458
8 Gong, Z., Long, Y., Hong, X., Chen, K.: 'Two certificateless aggregate signatures from bilinear maps'. Proc. SNPD, 2007, pp. 188–193
9 Gorantla, M.C., Saxena, A.: 'An efficient certificateless signature scheme'. Proc. CIS 2005, 2005, (*LNCS*, **3802**) pp. 110–116
10 Huang, X., Mu, Y., Susilo, W., Wong, D.S., Wu, W.: 'Certificateless signature revisited'. Proc. ACISP 2007, 2007, (*LNCS*, **4586**), pp.308–322
11 Huang, X., Susilo, W., Mu, Y., Zhang, F.: 'On the security of a certificateless signature scheme'. Proc. CANS 2005, 2005 (*LNCS*, **3810**), pp. 13–25
12 Yum, D.H., Lee, P.J.: 'Generic construction of certificateless signature'. Proc. ACISP 2004, 2004 (*LNCS*, **3108**), pp. 200–211
13 Zhang, L., Zhang, F.: 'A new certificateless aggregate signature scheme', *Comput. Commun.*, 2009, **32**, pp. 1079–1085
14 Zhang, Z., Wong, D.S., Xu, J., Feng, D.: 'Certificateless public-key signature: security model and efficient construction'. Proc. ACNS 2006, 2006, (*LNCS*, **3989**), pp. 293–308
15 Cao, X., Paterson, K.G., Kou, W.: 'An attack on a certificateless signature scheme', Cryptology ePrint Archive, Report 2006/367, 2006

© The Institution of Engineering and Technology 2013
doi: 10.1049/iet-ifs.2012.0134

16 Hu, B.C., Wong, D.S., Zhang, Z., Deng, X.: 'Certificateless signature: a new security model and an improved generic construction', *Des. Codes Cryptogr.*, 2007, **42**, (2), pp. 109–126
17 Zhang, Z., Feng, D.: 'Key replacement attack on a certificateless signature scheme', Cryptology ePrint Archive, Report 2006/453, 2006
18 Fan, C.I., Hsu, R.H., Ho, P.H.: 'Truly non-repudiation certificateless short signature scheme from bilinear pairings', *J. Inf. Sci. Eng.*, 2011, **24**, pp. 969–982
19 Boneh, D., Boyen, X.: 'Short signatures without random oracles and the SDH assumption in bilinear groups', *J. Cryptol.*, 2008, **21**, pp. 149–177
20 ISO/IEC 13888-1:2009: 'Information technology - security techniques - non-repudiation – Part 1: General,' 2009, available at http://www.iso.org/iso/catalogue_detail.htm?csnumber=50432
21 Mitsunari, S., Sakai, R., Kasahara, M.: 'A new traitor tracing', *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 2002, **E85-A**, (2), pp. 481–484
22 Tso, R., Yi, X., Huang, X.: 'Efficient and short certificateless signatures secure against realistic adversaries', *J. Supercomput.*, 2011, **55**, (2), pp. 173–191
23 Yang, G., Tan, C.H.: 'Certificateless cryptography with KGC trust level 3', *Theor. Comput. Sci.*, 2011, **412**, (39), pp. 5446–5457
24 Chen, Y.C., Horng, G.: 'On the security models for certificateless signature schemes achieving level 3 security', Cryptology ePrint Archive 2011/554, 2011
25 Boneh, D., Franklin, M.: 'Identity-based encryption from the weil pairing', *SIAM J. Comput.*, 2003, **32**, (3), pp. 586–615
26 Gentry, C.: 'Certificate-based encryption and the certificate revocation problem'. Proc. EUROCRYPTO'03, 2003 (*LNCS*, **2656**), pp. 272–293

8 Appendix A: Proofs of existential unforgeability

Lemma 4: Assume that the the modified k-CAA problem is intractable. Then no adversary, running polynomially probabilistic time algorithms, has non-negligible probability to win *Game I* in CLS-SNR.

Proof: Suppose an adversary $\mathcal{A}$ in *Game I* can existentially forge one signature of a message in CLS-SNR. Then we can construct a probabilistic polynomial time algorithm $\mathcal{C}$ which solves the modified k-CAA problem by running $\mathcal{A}$ and supplying correct responses to $\mathcal{A}$'s queries as defined in *Game I*. Assume the challenger $\mathcal{C}$ is given a random instance of the modified k-CAA problem as follows: given

$$\Big\{P \in \mathbb{G}_1, rP, aP, bP, raP, w_1, w_2 \ldots, w_k \in \mathbb{Z}_q^*, W_1 = \frac{1}{r+w_1}(abP), W_2 = \frac{1}{r+w_2}(abP), \ldots, W_k = \frac{1}{r+w_k}(abP)\Big\}$$

output a pair $(w', W' = 1/(r+w')abP)$ such that $w' \notin \{w_1, w_2, \ldots, w_k\}$, where $r, a, b \in \mathbb{Z}_q^*$ are unknown. Now $\mathcal{C}$ and $\mathcal{A}$ play the roles of the challenger and adversary, respectively, in *Game I*. Eventually, $\mathcal{C}$ may base on $\mathcal{A}$'s forgery to solve this modified k-CAA problem. Assume $\mathcal{A}$ will make at most q_{H_2} H_2 queries. $\mathcal{C}$ will maintain three initially empty tables: the H_1-table, the H_2-table, the H_3-table and the K-table containing tuples of $\langle \mathrm{ID}_j, \delta_j\rangle$, $\langle m_j, \mathrm{pk}_{\mathrm{ID}_j}, h_j, c_j\rangle$, $\langle \mathrm{ID}_j, \mathrm{pk}_{\mathrm{ID}_j}, Q_j, \alpha_j\rangle$ and $\langle \mathrm{ID}_j, \mathrm{pk}_{\mathrm{ID}_j}, r_{\mathrm{ID}_j}, D_{\mathrm{ID}_j}\rangle$, respectively.
Setup: $\mathcal{C}$ randomly chooses $s_1 \in \mathbb{Z}_q^*$ and sets $P_{\mathrm{pub}1} = s_1P$ and $P_{\mathrm{pub}2} = aP$. It then gives *params* to $\mathcal{A}$ where params $= \left(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_{\mathrm{pub}1}, P_{\mathrm{pub}2}, P, H_1, H_2, H_3\right)$.
Attack: $\mathcal{A}$ can adaptively perform the following queries.

- H_1 *queries*: $\mathcal{A}$ can query for the value of $H_1(\mathrm{ID}_i)$ for any user's identity ID_i. $\mathcal{C}$ works as follows:

  - If ID_i has been queried then $\mathcal{C}$ returns δ_i from the H_1-table.
  - Otherwise, $\mathcal{C}$ randomly chooses δ_i and returns δ_i to $\mathcal{A}$, and then inserts $\langle \mathrm{ID}_i, \delta_i\rangle$ into the H_1-table.

- H_2 *queries*: $\mathcal{A}$ can query for the value of $H_2(m_i, \mathrm{pk}_{\mathrm{ID}_i})$ for any pair $(m_i, \mathrm{pk}_{\mathrm{ID}_i})$. $\mathcal{C}$ works as follows:

  - If $(m_i, \mathrm{pk}_{\mathrm{ID}_i})$ has been queried before then $\mathcal{C}$ returns h_i to $\mathcal{A}$ from the H_3-table.
  - Otherwise, $\mathcal{C}$ randomly sets c_i to 0 or 1 with $\Pr[c_i = 0] = 1/q_{H_2}$. If $c_i = 0$, $\mathcal{C}$ randomly chooses h_i such that $h_i \notin \{w_1, w_2, \ldots, w_k\}$; otherwise, chooses $h_i \in \{w_1, w_2, \ldots, w_k\}$. $\mathcal{C}$ returns h_i to $\mathcal{A}$ and inserts $\langle m_i, \mathrm{pk}_{\mathrm{ID}_i}, h_i, c_i\rangle$ into the H_2-table.

- H_3 *queries*: $\mathcal{A}$ can query for the value of $H_3(\mathrm{ID}_i, \mathrm{pk}_{\mathrm{ID}_i})$ for any pair $(\mathrm{ID}_i, \mathrm{pk}_{\mathrm{ID}_i})$. If $(\mathrm{ID}_i, \mathrm{pk}_{\mathrm{ID}_i})$ has been queried before then $\mathcal{C}$ returns Q_i to $\mathcal{A}$ from the H_3-table. Otherwise, $\mathcal{C}$ performs as follows:

  - If $\mathrm{pk}_{\mathrm{ID}_i}$ corresponds to $c_i = 0$ in H_2-table, $\mathcal{C}$ randomly generates α_i and computes $Q_i = \alpha_i(bP)$. $\mathcal{C}$ thus returns Q_i to $\mathcal{A}$ and inserts $\langle \mathrm{ID}_i, \mathrm{pk}_{\mathrm{ID}_i}, Q_i, \alpha_i\rangle$ into the H_3-table.
  - Otherwise, $\mathcal{C}$ randomly generates α_i and computes $Q_i = \alpha_i P$. $\mathcal{C}$ thus returns Q_i to $\mathcal{A}$ and inserts $\langle \mathrm{ID}_i, \mathrm{pk}_{\mathrm{ID}_i}, Q_i, \alpha_i\rangle$ into the H_3-table.

- *Public-Key queries*: $\mathcal{A}$ can query for the public key of any user for identity ID_i. If ID_i has been queried before, $\mathcal{C}$ returns the public key $\mathrm{pk}_{\mathrm{ID}}$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ performs as follows:

  - If ID_i corresponds to $c_i = 0$ in H_2-table, $\mathcal{C}$ sets the public key $\mathrm{pk}_{\mathrm{ID}_i} = raP + \delta_i(rP)$. In fact, $\mathrm{pk}_{\mathrm{ID}_i} = raP + \delta_i(rP)$ would be set before H_2 queries.
  - Otherwise, $\mathcal{C}$ randomly chooses a secret value r_{ID_i} and a public key $\mathrm{pk}_{\mathrm{ID}_i}$, then sends $\mathrm{pk}_{\mathrm{ID}_i}$ to $\mathcal{A}$ and inserts $\langle \mathrm{ID}_i, r_{\mathrm{ID}_i}, \mathrm{pk}_{\mathrm{ID}_i}, \bigstar\rangle$ into the K-table.

- *Secret-Value queries*: $\mathcal{A}$ can query for the public key of any identity ID_i. If r_{ID_i} is in the K-table then $\mathcal{C}$ returns r_{ID_i} to $\mathcal{A}$. Otherwise, $\mathcal{C}$ does the following:

  - If ID_i's $\mathrm{pk}_{\mathrm{ID}_i}$ corresponds to $c_i = 0$ in the H_2-table then $\mathcal{C}$ aborts and terminates.
  - Otherwise, $\mathcal{C}$ returns $\perp$ to indicate that r_{ID_i} has not been set.

- *Partial-Private-Key queries*: $\mathcal{A}$ can query for the partial-private-key of any identity ID_i. If D_{ID_i} is in the K-table then $\mathcal{C}$ returns D_{ID_i} to $\mathcal{A}$. Otherwise, $\mathcal{C}$ does the following:

  - If ID_i's $\mathrm{pk}_{\mathrm{ID}_i}$ corresponds to $c_i = 0$ in the H_2-table, $\mathcal{C}$ aborts and terminates.
  - Otherwise, $\mathcal{C}$ goes back to perform H_1, H_3 queries and bases on the H_1-table and the-H_3 table to determine and return the partial-private-key D_{ID_i} to $\mathcal{A}$ where $D_{\mathrm{ID}_i} = 1/(s_1 + \delta_i)\alpha_i(aP)$ and δ_i and α_i are taken from the H_1 and H_3-tables, respectively. Finally, $\mathcal{C}$ inserts $\langle \mathrm{ID}_i, \bigstar, \bigstar, D_{\mathrm{ID}_i}\rangle$ into the K-table.

doi: 10.1049/iet-ifs.2012.0134
© The Institution of Engineering and Technology 2013

- *Public-Key replacement*: For any user with public key $\mathrm{pk}_{\mathrm{ID}_i}$, $\mathcal{A}$ has the ability to set a new public key $\mathrm{pk}'_{\mathrm{ID}_i}$, then replace $\mathrm{pk}_{\mathrm{ID}_i}$ with $\mathrm{pk}'_{\mathrm{ID}}$.
- *Sign query*: $\mathcal{A}$ can query for the signature, S_i, generated by a user with identity ID_i for a message m_i. $\mathcal{C}$ will generate a signature S_i corresponding to ID_i, m_i and public key pk_i. If S_i has been queried before then $\mathcal{C}$ will return the previous result to $\mathcal{A}$. Otherwise, $\mathcal{C}$ does the following:

  - If ID_i corresponds to $c_i = 0$ in the H_2-table then $\mathcal{C}$ aborts and terminates.
  - Otherwise, $\mathcal{C}$ will generate the signature S_i by computing $S_i = 1/(s_1 + \delta_i)\alpha_i W_i$ based on the H_1, H_2, H_3 and K-tables, where W_i corresponds to h_i in the H_2-table.

Forgery: $\mathcal{A}$ forges a signature, S^* of a message, m^*, and the current public key $\mathrm{pk}_{\mathrm{ID}^*}$. If the identity ID^* corresponds to $c^* \neq 0$ in the H_2-table then $\mathcal{C}$ aborts and terminates. If $\mathcal{A}$ successfully forges a valid signature, S^* of a message, m^* where $S^* = 1/((s_1 + \delta^*)(r + h^*))\alpha^*(abP)$, h^* corresponds to m^* and $\mathrm{pk}_{\mathrm{ID}^*}$ from H_2-table. Thus $\mathcal{C}$ can use $\mathcal{A}$'s forgery to compute $1/(r + w^*)(abP) = (\alpha^*)^{-1}(s_1 + \delta^*)S^*$ where $w^* = h^* \notin \{w_1, \ldots, w_k\}$. Therefore $\mathcal{C}$ can successfully solve the modified k-CAA problem if the following two events occurred simultaneously:

- ε_1: $\mathcal{C}$ does not abort in any $\mathcal{A}$'s query.
- ε_2: $\mathcal{C}$ does not abort in *Forgery*.

Claim 1. The probability that $\mathcal{C}$ does not abort in any $\mathcal{A}$'s partial-private-key query and secret-value query is at least $1/e^2$.

Proof: The probability of asking D_{ID} and r_{ID} corresponding to $c = 0$ is $1/q_{H_2}$. Thus, $\Pr[\mathcal{E}_1] = ((1 - 1/(q_{H_2}))^{q_{H_2}})^2 \geq 1/e^2$, where e is the base of the natural logarithm. □

Claim 2. The probability that $\mathcal{C}$ does not abort in *Forgery* is $1/q_{H_2}$.

Proof: Assume $\mathcal{A}$ outputs a forged key pair $(D_{\mathrm{ID}^*}, \mathrm{pk}_{\mathrm{ID}^*})$. Then $\mathcal{C}$ does not abort if the corresponding $c^* = 0$. Hence, $\Pr[\mathcal{E}_2] = 1/q_{H_2}$. □
By Claim 1 and Claim 2, $\mathcal{C}$ does not abort with probability $\varepsilon' = \Pr[\mathcal{E}_1]\Pr[\mathcal{E}_2]\varepsilon \geq 1/(e^2 q_{H_2})\varepsilon$ where ε is the probability of $\mathcal{A}$ to win *Game I*. Therefore $\mathcal{C}$ solves the modified k-CAA problem with probability $\varepsilon' \geq 1/(q_{H_2}e^2)\varepsilon$ where ε is the probability of $\mathcal{A}$ to win *Game I*. Since the modified k-CAA problem is assumed to be intractable, ε' is negligible. Hence, ε is negligible. That is, there is no type 1 adversary can existentially forge a valid signature in CLS-SNR. □

Lemma 5: Assume that the k-CAA problem is intractable. Then no adversary, running polynomially probabilistic time algorithms, has non-negligible probability to win *Game II* in CLS-SNR.

Proof: Suppose an adversary $\mathcal{A}$ in *Game II* can existentially forge one signature of a message in CLS-SNR. Then we can construct a probabilistic polynomial time algorithm $\mathcal{C}$ which solves the k-CAA problem by running $\mathcal{A}$ and supplying correct responses to $\mathcal{A}$'s queries as defined in *Game II*. Assume the challenger $\mathcal{C}$ is given a random instance of the k-CAA problem as follows: given

$$\left\{P \in \mathbb{G}_1, rP, w_1, w_2 \ldots, w_k \in \mathbb{Z}_q^*, W_1 = \frac{1}{r + w_1}P, W_2 = \frac{1}{r + w_2}P, \ldots, W_k = \frac{1}{r + w_k}P\right\}$$

output a pair (w', $W' = 1/(r + w')P$) such that $w' \notin \{w_1, w_2 \ldots, w_k\}$, where $r, a, b \in \mathbb{Z}_q^*$ are unknown. Now $\mathcal{C}$ and $\mathcal{A}$ play the roles of the challenger and adversary, respectively, in *Game II*. Eventually, $\mathcal{C}$ may base on $\mathcal{A}$'s forgery to solve this k-CAA problem. Assume $\mathcal{A}$ will make at most q_{H_2} H_2 queries. $\mathcal{C}$ will maintain three initially empty tables: the H_1-table, the H_2-table, the H_3-table and the K-table containing tuples of $\langle \mathrm{ID}_j, \delta_j \rangle$, $\langle m_j, \mathrm{pk}_{\mathrm{ID}_j}, h_j, c_j \rangle$, $\langle \mathrm{ID}_j, \mathrm{pk}_{\mathrm{ID}_j}, Q_j, \alpha_j \rangle$ and $\langle \mathrm{ID}_j, \mathrm{pk}_{\mathrm{ID}_j}, r_{\mathrm{ID}_j}, D_{\mathrm{ID}_j} \rangle$, respectively.
Setup: $\mathcal{C}$ randomly chooses $s_1 \in \mathbb{Z}_q^*$ and sets $P_{\mathrm{pub1}} = s_1 P$ and $P_{\mathrm{pub2}} = s_2 P$. It then gives params, s_1, s_2 to $\mathcal{A}$ where params $= (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_{\mathrm{pub1}}, P_{\mathrm{pub2}}, P, H_1, H_2, H_3)$.

Attack: $\mathcal{A}$ can adaptively perform the following queries.

- H_1 *queries*: As the proof of Lemma 4.
- H_2 *queries*: $\mathcal{A}$ can query for the value of $H_2(m_i, \mathrm{pk}_{\mathrm{ID}_i})$ for any pair $(m_i, \mathrm{pk}_{\mathrm{ID}_i})$. $\mathcal{C}$ works as follows:

  - If $(m_i, \mathrm{pk}_{\mathrm{ID}_i})$ has been queried before then $\mathcal{C}$ returns h_i to $\mathcal{A}$ from the H_3-table.
  - Otherwise, $\mathcal{C}$ randomly sets c_i to 0 or 1 with $\Pr[c_i = 0] = 1/q_{H_2}$. If $c_i = 0$, $\mathcal{C}$ randomly chooses h_i such that $h_i \notin \{w_1, w_2 \ldots, w_k\}$; otherwise, chooses $h_i \in \{w_1, w_2 \ldots, w_k\}$. $\mathcal{C}$ returns h_i to $\mathcal{A}$ and inserts $\langle m_i, \mathrm{pk}_{\mathrm{ID}_i}, h_i, c_i \rangle$ into the H_2-table.

- H_3 *queries*: $\mathcal{A}$ can query for the value of $H_3(\mathrm{ID}_i, \mathrm{pk}_{\mathrm{ID}_i})$ for any pair $(\mathrm{ID}_i, \mathrm{pk}_{\mathrm{ID}_i})$. If $(\mathrm{ID}_i, \mathrm{pk}_{\mathrm{ID}_i})$ has been queried before then $\mathcal{C}$ returns Q_i to $\mathcal{A}$ from the H_3-table. Otherwise, $\mathcal{C}$ randomly generates α_i and computes $Q_i = \alpha_i P$. $\mathcal{C}$ thus returns Q_i to $\mathcal{A}$ and inserts $\langle \mathrm{ID}_i, \mathrm{pk}_{\mathrm{ID}_i}, Q_i, \alpha_i \rangle$ into the H_3-table.
- *Public-Key queries*: $\mathcal{A}$ can query for the public key of any user for identity ID_i. If ID_i has been queried before, $\mathcal{C}$ returns the public key $\mathrm{pk}_{\mathrm{ID}}$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ performs as follows:

  - If ID_i corresponds to $c_i = 0$ in H_2-table, $\mathcal{C}$ sets the public key $\mathrm{pk}_{\mathrm{ID}_i} = (s_1 + \delta_i)rP$. In fact, $\mathrm{pk}_{\mathrm{ID}_i} = (s_1 + \delta_i)rP$ would be set before H_2 queries.
  - Otherwise, $\mathcal{C}$ randomly chooses a secret value r_{ID_i} and a public key $\mathrm{pk}_{\mathrm{ID}_i}$, then sends $\mathrm{pk}_{\mathrm{ID}_i}$ to $\mathcal{A}$ and inserts $\langle \mathrm{ID}_i, r_{\mathrm{ID}_i}, \mathrm{pk}_{\mathrm{ID}_i}, \bigstar \rangle$ into the K-table.

- *Secret-Value queries*: $\mathcal{A}$ can query for the public key of any identity ID_i. If r_{ID_i} is in the K-table then $\mathcal{C}$ returns r_{ID_i} to $\mathcal{A}$. Otherwise, $\mathcal{C}$ does the following:

  - If ID_i's $\mathrm{pk}_{\mathrm{ID}_i}$ corresponds to $c_i = 0$ in the H_2-table then $\mathcal{C}$ aborts and terminates.
  - Otherwise, $\mathcal{C}$ returns $\perp$ to indicate that r_{ID_i} has not been set.

- *Sign query*: $\mathcal{A}$ can query for the signature, S_i, generated by a user with identity ID_i for a message m_i. $\mathcal{C}$ will generate a signature S_i corresponding to ID_i, m_i and public key pk_i. If

© The Institution of Engineering and Technology 2013

doi: 10.1049/iet-ifs.2012.0134

S_i has been queried before then $\mathcal{C}$ will return the previous result to $\mathcal{A}$. Otherwise, $\mathcal{C}$ does the following:

- If ID_i corresponds to $c_i = 0$ in the H_2-table then $\mathcal{C}$ aborts and terminates.
- Otherwise, $\mathcal{C}$ will generate the signature S_i by computing $S_i = s_2\alpha_i/(s_1 + \delta_i)W_i$ based on the H_1, H_2, H_3 and K-tables, where W_i corresponds to h_i in the H_2-table.

Forgery: $\mathcal{A}$ forges a signature, S^* of a message, m^*, and the current public key $\mathrm{pk}_{\mathrm{ID}^*}$. If the identity ID^* corresponds to $c^* \neq 0$ in the H_2-table then $\mathcal{C}$ aborts and terminates. If $\mathcal{A}$ successfully forges a valid signature, S^* of a message, m^* where $S^* = 1/((s_1 + \delta^*)(r + h^*))\alpha^*(s_2P)$, h^* corresponds to m^* and $\mathrm{pk}_{\mathrm{ID}^*}$ from H_2-table. Thus $\mathcal{C}$ can use $\mathcal{A}$'s forgery to compute $1/(r + w^*)P = (s_2\alpha^*)^{-1}(s_1 + \delta^*)S^*$ where $w^* = h^* \notin \{w_1, \ldots, w_k\}$. Therefore $\mathcal{C}$ can successfully solve the k-CAA problem if the following two events occurred simultaneously:

- ε_1: $\mathcal{C}$ does not abort in any $\mathcal{A}$'s query.
- ε_2: $\mathcal{C}$ does not abort in *Forgery*.

Claim 1. The probability that $\mathcal{C}$ does not abort in any $\mathcal{A}$'s secret-value query is at least $1/e$.

Proof: The probability of asking D_{ID} and r_{ID} corresponding to $c = 0$ is $1/q_{H_2}$. Thus, $\Pr[\mathcal{E}_1] = ((1 - 1/(q_{H_2}))^{q_{H_2}}) \geq 1/e$, where e is the base of the natural logarithm. □

Claim 2. The probability that $\mathcal{C}$ does not abort in *Forgery* is $1/q_{H_2}$.

Proof: Assume $\mathcal{A}$ outputs a forged key pair $(D_{\mathrm{ID}^*}, \mathrm{pk}_{\mathrm{ID}^*})$. Then $\mathcal{C}$ does not abort if the corresponding $c^* = 0$. Hence, $\Pr[\mathcal{E}_2] = 1/q_{H_2}$. □

By Claim 1 and Claim 2, $\mathcal{C}$ does not abort with probability $\varepsilon' = \Pr[\mathcal{E}_1]\Pr[\mathcal{E}_2]\varepsilon \geq 1/(eq_{H_2})\varepsilon$, where ε is the probability of $\mathcal{A}$ to win *Game II*. Therefore $\mathcal{C}$ solves the k-CAA problem with probability $\varepsilon' \geq 1/(q_{H_2}e)\varepsilon$, where ε is the probability of $\mathcal{A}$ to win *Game II*. Since the k-CAA problem is assumed to be intractable, ε' is negligible. Hence, ε is negligible. That is, there is no type 2 adversary can existentially forge a valid signature in CLS-SNR. □

9 Appendix B: Specific non-repudiation services (ISO/IEC 13888-1:2009)

Non-repudiation of creation: This service is intended to protect against an entity's false denial of having created the content of a message (i.e. being responsible for the content of a message).

Non-repudiation of delivery: This service is intended to protect against a recipient's false denial of having received the message and recognised the content of a message.

Non-repudiation of knowledge: This service is intended to protect against a recipient's false denial of having taken notice of the content of a received message.

Non-repudiation of origin: This service is intended to protect against the originator's false denial of having approved the content of a message and of having sent a message.

Non-repudiation of receipt: This service is intended to protect against a recipient's false denial of having received a message.

Non-repudiation of sending: This service is intended to protect against the sender's false denial of having sent a message.

Non-repudiation of submission: This service is intended to provide evidence that a delivery authority has accepted the message for transmission.

Non-repudiation of transport: This service is intended to provide evidence for the message originator that a delivery authority has delivered the message to the intended recipient.

© The Institution of Engineering and Technology 2013