3-1-2014

The Insecurity of Two Proxy Signcryption Schemes: Proxy Credential Forgery Attack and How to Prevent It

Jyh-haw Yeh
Boise State University

The insecurity of two proxy signcryption schemes - proxy credential forgery attack and how to prevent it

Jyh-haw Yeh

Received: date / Accepted: date

Abstract Securing different online e-business activities usually requires applying different cryptographic algorithms. The proxy signcryption algorithms are designed for applications such as online proxy auction or online proxy signatures on business contracts, which require a proxy agent to sign on confidential messages. This paper proposes a proxy credential forgery attack to two recent proxy signcryption schemes in the literature. Using the attack, a malicious proxy signer can create a fake proxy credential from his original credential to extend his signing power. Simple modifications to these two schemes are also provided in this paper to prevent the attack without adding too much computational complexity. In addition to the contribution of introducing a new type of attacks to signcryption schemes, the paper also points out that, while designing a secure proxy signcryption scheme, not only the unforgeability of proxy signatures is important, the unforgeability of proxy credentials should be equally important as well.

Keywords Proxy signcryption · Proxy credential forgery attack · Bilinear pairings

1 Introduction

The notion of proxy signature was introduced by Mambo et al. in 1996 [1]. It allows a proxy signer to sign a message on behalf of an original signer, where the proxy signature can be publicly verified. Following Mambo's paper, quite a few proxy signature schemes were proposed such as those in [2–10].

Jyh-haw Yeh
Computer Science, Boise State University, 1910 University Drive, Boise, Idaho, USA 83725
Tel.: +1-208-4263034
Fax: +1-208-4262470
E-mail: jhyeh@boisestate.edu

However, for e-business applications such as online proxy auction or business contract signing by an authorized proxy, it might be necessary to have proxy signatures on confidential messages. In 1997, Zheng [11] proposed a signcryption (signature and encryption) scheme which only allows a designated recipient to verify the signature to protect message confidentiality. Following Zheng's paper, some other signcryption schemes [12–17] were also proposed in literature. In these proxy signcryption schemes, the original signer creates a cryptographically protected proxy credential and gives it to a proxy signer. Based on the proxy credential, the proxy signer can sign and encrypt a message on behalf of the original signer and then send it to the signature verifier. During or after a signing process, a conflict (either the original signer denies his proxy delegation, or a proxy signer denies his proxy signature) may occur. If the honest party in a conflict is able to provide verifiable evidences, the proxy signcryption scheme is non-repudiated. To resolve a conflict, it needs a trusted judge to verify the provided evidences. Section 2.2 describes these possible conflicts, the verifiable evidences and conflict resolving procedures.

In this paper, we proposed a Proxy Credential Forgery (PCF) attack to two proxy signcryption schemes, which are the LWHY scheme (proposed by Lin, Wu, Huang and Yeh in 2010) [16] and the EA scheme (proposed by Elkamchouchi and Aboueleoud in 2007) [17]. Using the PCF attack, these two schemes can be compromised and fail to preserve the non-repudiation property. That is, a malicious proxy signer is able to create a fake proxy credential to enlarge/extend his signing power in these two schemes and the fake proxy credential can pass the verification checks. The proposed PCF attack is general and can be used to attack either proxy signature or signcryption schemes if the schemes have similar proxy credential generation algorithms as LWHY's and EA's. At the end of the paper, some simple modifications to these two schemes are provided to avoid such attack.

The organization of the paper is as follows. Section 2 discusses the proxy signcryption security model and defines the proxy credential forgery attack. Section 3 gives the mathematical background of the bilinear pairing, which is the underlying cryptosystem used to build the LWHY and the EA proxy signcryption schemes. Section 4 briefly describes the LWHY scheme and proposes how to attack the scheme by forging the proxy credential. Similarly in Section 5, the EA scheme is described and then the proxy credential forgery attack to the scheme is presented. Sections 6 proposes the improvement to both the LWHY and the EA schemes so that the modified schemes are secure against the attack. Section 7 concludes the paper.

2 Proxy signcryption security model

To secure applications such as online proxy auction or business contract signing by an authorized proxy, it requires using an proxy signcryption scheme to ensure some security goals. Before itemizing these security requirements, this

section will first describe the proxy signcryption processing model such as who are the involved parties and what are their roles in the process.

2.1 Proxy signcryption process and involved parties

In an proxy signcryption application, usually it will have four parties involved. The original signer U_o delegates his signing power to a proxy signer U_p to sign and encrypt a message to a designated signature verifier U_v. If there is a conflict occurred, a trusted judge U_j will resolve the conflict by verifying the evidences presented by the conflicting parties.

In general, there are five phases for a proxy signcryption scheme. They are

- `Setup:` A key generation center (KGC) chooses a cryptosystem, picks some system parameters *param* and then make these parameters public. In some schemes, the KGC generates the public-private key pairs for all participants, whereas in other schemes, each participant chooses his own public-private key pair based on the public *param*.
- `Proxy credential generation (PCG):` In this phase, the original signer U_o is responsible to generate the proxy credential. The PCG algorithm takes as input of the private key of U_o and outputs a proxy credential to the proxy signer U_p.
- `Proxy credential verification (PCV):` Upon receiving the proxy credential, the proxy signer U_p should verify the validity of the credential. The PCV algorithm takes as input of the credential and the original signer's public key and then outputs a boolean value indicating either "pass" or "fail".
- `Signcrypted message generation (SMG):` With a valid proxy credential, the proxy signer U_p is capable of signcrypting a message m to the designated signature verifier U_v. The SMG algorithm takes as input of the plaintext m, the proxy credential, the private key of U_p and the public key of U_v. It outputs a signcrypted message δ.
- `Signature recovery and verification (SRV):` After receiving the signcrypted message δ, the signature verifier U_v should recover the message and signature and then verify the validity of the signature. The SRV algorithm takes as input the signcrypted message δ, the private key of U_v and the public keys of U_o and U_p. It outputs a plaintext m and accepts a recovered proxy signature if δ is valid. Otherwise, an error is reported.

2.2 Security requirement

A proxy signcryption scheme must satisfy two non-repudiation security requirement.

- `Proxy credential non-repudiation:` If an original signer U_o did create and give a proxy credential to a proxy signer U_p, U_o cannot deny his delegation later. To achieve this requirement, the scheme must

1. provide a correct PCV algorithm that the validity of the credential can be publicly checked, given the credential and U_o's public key as input.
2. ensure the **proxy credential unforgeability** (PCNF) property. To be more specific, the scheme needs to ensure that nobody, except the original signer U_o or the KGC, is able to generate a valid proxy credential. That is, adversaries cannot generate a fake credential and pass the PCV test.

In case an original signer U_o later denies a proxy credential held by a proxy signer U_p, the resolving procedure is as follows:

1. The trusted judge U_j asks U_p to present the proxy credential.
2. If the provided credential passes the PCV test, the judge identifies the dishonesty of U_o on denying his own delegation. Otherwise, U_p is considered dishonest on using a false credential.

– `Signcrypted message non-repudiation:` If a proxy signer did signcrypt a message to a signature verifier U_v, U_p cannot deny his signature later. Again, to achieve this non-repudiation property, it requires the scheme to

1. provide a correct SMG algorithm that securely encrypts both the proxy signature and message.
2. provide a correct SRV algorithm that is able to recover and verify the validity of both the proxy signature and message.
3. ensure the **signcrypted messages unforgeability** (SMNF) property. Or in other words, the scheme needs to ensure that nobody but the proxy signer U_p is able to generate the signcrypted message and pass the SRV test.

If a proxy signer U_p later denies a proxy signature held by a designated recipient U_v, the resolving procedure is as follows:

1. The trusted judge U_j asks U_v to present the recovered message and proxy signature from the signcrypted message.
2. With provided evidences, the judge can then use the SRV algorithm to check the validity of the signature. If the checking succeeds, U_p is dishonest on denying his own signature. Otherwise, the signature presented by U_v is invalid.

2.3 Adversary types

Based on different knowledge, there are four types of adversaries. For each type of adversary, we give some possible malicious actions from the adversary if the underlying proxy signcryption scheme is not properly designed to prevent them.

– `Type I adversary:` An original signer U_o knows his own public and private keys plus the public keys of U_p and U_v. A malicious U_o may issue a proxy credential to a U_p and later deny it, or he may proxy signcrypt a message and fool a signature verifier U_v to believe that the signcrypted message is from a proxy signer U_p.

- `Type II adversary:` A proxy signer U_p knows his own public and private keys plus the public keys of U_o and U_v. A malicious U_p may signcrypt a message to a signature verifier U_v and later deny it, or he may create a fake proxy credential so that he can proxy signcrypt a message without a proper delegation from the original signer U_o. The second malicious action from U_p is the proxy credential forgery (PCF) attack we proposed in this paper against the LWHY and the EA proxy signcryption schemes.
- `Type III adversary:` A signature verifier U_v knows his own public and private keys plus the public keys of U_o and U_p. A malicious U_v may try to generate a fake proxy signcrypted message to himself, fooling people to believe the message is from a proxy signer U_p.
- `Type IV adversary:` An outsider only knows the public keys of U_o, U_p and U_v. This type of adversary has the least power since he doesn't know the private key of anyone. Malicious actions may include identity-spoofing himself as an original signer U_o or a proxy signer U_p to generate a fake proxy credential or a fake signcrypted message, respectively. He may be also interested in cryptanalysis of a signcrypted message to recover the message.

2.4 Proxy credential forgery attack

The proposed proxy credential forgery attack in this paper is a type II attack, i.e., an attack from a malicious proxy signer U_p. Before providing examples of the attack in Sections 4 and 5, let's define the attack first.

Definition: *The Proxy Credential Forgery (PCF) attack is an attack from a proxy signer, with the knowledge of an existing proxy credential, public parameters of the the underlying cryptosystem and the public keys of all involved parties, who can maliciously create a fake proxy credential to pass the PCV test.*

3 Cryptographic background - bilinear pairings

Bilinear pairing is a popular cryptosystem used recently in some efficient encryption or signature/signcryption schemes [16–22]. We will briefly describe the bilinear pairings in this section.

Let $(G_1, +)$ and $(G_2, \times)$ be two cyclic groups of the same prime order q, B be the generator of the additive group G_1, and $e : G_1 \times G_1 \rightarrow G_2$ be a bilinear map if it has the following properties:

(1) Bilinearity: $\forall D, E, F \in G_1$, $\forall a, b \in Z_q^*$, it satisfies that

$$e(D, E) = e(E, D) \tag{1}$$

$$e(aD, bE) = e(D, E)^{ab} = e(bD, aE) \tag{2}$$

$$e(D, E + F) = e(D, E)e(D, F) \tag{3}$$

(2) Non-degeneracy: If B is a generator of G_1, then $e(B, B)$ is a generator of G_2.
(3) Computability: $\forall D, E \in G_1$, there exists a polynomial-time algorithm to efficiently compute $e(D, E)$.

In general, most cryptographic schemes based on bilinear maps rely on the bilinear variants of Diffie-Hellman hard problems as follows.

1. Computational Bilinear Diffie-Hellman Problem (CBDHP): Given D, aD, bD for $a, b \in Z_q^*$, to compute abD.
2. Decision Bilinear Diffie-Hellman Problem (DBDHP): Given D, aD, bD, cD for $a, b, c \in Z_q^*$, and an element $h \in G_2$, to decide $h \stackrel{?}{=} e(D, D)^{abc}$.

Currently no known algorithm is available to solve them efficiently.

4 LWHY scheme and the PCF attack

This section gives a brief review of the LWHY scheme and then describes the proxy credential forgery attack to the scheme.

The LWHY scheme has five phases: setup, proxy credential generation, proxy credential verification, signcrypted message generation, and signature recovery and verification. If a signer, either U_o or U_p, later denies a signature, a trusted judge will be asked to identify the dishonest party.

4.1 Setup

Given a security parameter k, a trusted key generation center KGC selects two cyclic groups $(G_1, +)$ and $(G_2, \times)$ of the same prime order q where $|q| = 2^k$. Let B be a generator of order q over G_1 and $e : G_1 \times G_1 \rightarrow G_2$ be a bilinear map. The KGC also chooses three collision-free hash functions

$h_1 : \{0,1\}^k \times G_1 \rightarrow Z_q$
$h_2 : G_1 \rightarrow G_1$
$h_3 : G_2 \times G_1 \rightarrow \{0,1\}^k$

Finally, the KGC publishes $\{G_1, G_2, q, B, e, h_1, h_2, h_3\}$. Each user U_i chooses a private key $x_i \in Z_q$ and a public key

$$Y_i = x_i B \tag{4}$$

4.2 Proxy credential generation

U_o delegates his signing power to U_p, specifying the detail of the delegation in a warrant m_w. For example, it may include the identities of U_o, U_p and U_v, as well as the expiration date of this delegation. U_o chooses a secret random number $d \in Z_q$ and then computes

$$N = dB \tag{5}$$

$$\sigma = x_o + dm_w \bmod q \tag{6}$$

The proxy credential $C = (\sigma, N, m_w)$ will be sent to U_p.

4.3 Proxy credential verification

Upon receiving the proxy credential C from U_o, the proxy signer U_p verifies its validity by performing the following PCV test.

$$\sigma B \stackrel{?}{=} Y_o + m_w N \tag{7}$$

According to LWHY's paper, they claimed the credential is valid if Eq. (7) holds since

$$\begin{aligned} \sigma B &= (x_o + dm_w)B && \text{- by Eq. (6)} \\ &= x_o B + dBm_w \\ &= Y_o + m_w N && \text{- by Eq. (4), (5)} \end{aligned}$$

4.4 Signcrypted message generation

After the proxy credential is verified, U_p signcrypts a message m on behalf of U_o. U_p chooses a secret random number $r \in Z_q$ and computes

$$R = rB \tag{8}$$

$$S = r(h_1(m, R) + x_p + \sigma)^{-1} B \tag{9}$$

$$\Phi = e(h_2(\sigma Y_v), x_p Y_v) \tag{10}$$

$$\Psi = E_v(S) \tag{11}$$

$$\Omega = h_3(\Phi, R) \oplus m \tag{12}$$

where $E_v()$ denotes a symmetric encryption function with a share secret key v between U_p and U_v. Then U_p delivers the signcrypted message $\delta = (R, \Psi, \Omega, N, m_w)$ to the recipient U_v.

4.5 Signature recovery and verification

U_v receives the signcrypted message δ from U_p. U_v first computes

$$\Phi = e(h_2(x_v(Y_o + m_w N)), x_v Y_p) \tag{13}$$

since

$$
\begin{aligned}
& e(h_2(x_v(Y_o + m_w N)), x_v Y_p) \\
=& e(h_2(x_v(\sigma B)), x_v x_p B) && \text{- by Eq. (7), (4)} \\
=& e(h_2(\sigma Y_v), x_p Y_v) && \text{- by Eq. (4)} \\
=& \Phi && \text{- by Eq. (10)}
\end{aligned}
$$

Thus, the computation of Eq. (13) can recover the original Φ generated from Eq. (10) by U_p. With Φ, U_v can recover the message m since

$$m = h_3(\Phi, R) \oplus \Omega \tag{14}$$

U_v then recovers S from the received Ψ by

$$S = D_v(\Psi) \tag{15}$$

where $D_v()$ is the symmetric decryption function with the key v. Finally U_v verifies the proxy signature by performing the following SRV test.

$$e(h_1(m, R)B + Y_p + Y_o + m_w N, S) \stackrel{?}{=} e(B, R) \tag{16}$$

Since

$$
\begin{aligned}
& e(h_1(m, R)B + Y_p + Y_o + m_w N, S) \\
=& e(\ h_1(m, R)B + Y_p + Y_o + m_w N, \\
& \quad r(h_1(m, R) + x_p + \sigma)^{-1} B) && \text{- by Eq. (9)} \\
=& e(\ h_1(m, R)B + x_p B + x_o B + dBm_w, && \text{- by Eq. (4), (5)} \\
& \quad r(h_1(m, R) + x_p + x_o + dm_w)^{-1} B) && \text{- by Eq. (6)} \\
=& e(\ (h_1(m, R) + x_p + x_o + dm_w)B, \\
& \quad r(h_1(m, R) + x_p + x_o + dm_w)^{-1} B) \\
=& e(B, rB) && \text{- by Eq. (2)} \\
=& e(B, R) && \text{- by Eq. (8)}
\end{aligned}
$$

4.6 The PCF attack to the LWHY scheme

This section describes a proxy credential forgery attack to the LWHY scheme. Without proper delegation from the original signer U_o, a malicious proxy signer U_p can unlawfully sign a message on behalf of U_o without being detected by the designed recipient U_v. Furthermore, if U_o discovers the unlawful proxy signature later, he hands over the case to a trusted judge. Using the non-repudiation tests in the LWHY scheme, the judge cannot identify the dishonesty of U_p.

4.6.1 U_p generates a fake proxy credential

A malicious U_p can increase his signing power without a proper delegation from U_o, such as extending the expiration date of the warrant or adding additional signature recipients into the warrant, by forging a fake proxy credential (σ', N', m'_w) from the original (σ, N, m_w) that he received from U_o before. U_p

chooses a fake proxy warrant m'_w (with the extended signing power) and then computes the corresponding σ' and N' as follows:

$$\sigma' = \frac{m'_w}{m_w}\sigma = \frac{m'_w}{m_w}x_o + dm'_w \mod q \tag{17}$$

$$N' = \frac{\frac{m'_w}{m_w}Y_o + m'_w N - Y_o}{m'_w} \tag{18}$$

Now, U_p picks a random number $r' \in Z_q$ and uses the fake σ' to compute $R', S', \Phi', \Psi', \Omega'$ as Eq. (8) to Eq. (12). That is,

$$R' = r'B \tag{19}$$

$$S' = r'(h_1(m, R') + x_p + \sigma')^{-1}B \tag{20}$$

$$\Phi' = e(h_2(\sigma' Y_v), x_p Y_v) \tag{21}$$

$$\Psi' = E_v(S') \tag{22}$$

$$\Omega' = h_3(\Phi', R') \oplus m \tag{23}$$

and sends the signcrypted message $\delta' = (R', \Psi', \Omega', N', m'_w)$ to U_v.

4.6.2 Pass signature recovery and verification

Upon receiving the fake signcrypted message, U_v performs the procedure as described from Eq. (13) to Eq. (16) and will not detect the forgery of the signcrypted message. U_v computes

$$\begin{aligned}
\Phi &= e(h_2(x_v(Y_o + m'_w N')), x_v Y_p) && \text{- by Eq. (13)}\\
&= e(h_2(x_v x_o B + x_v m'_w \cdot \tfrac{\frac{m'_w}{m_w}Y_o + m'_w N - Y_o}{m'_w}), x_v x_p B) && \text{- by Eq. (4), (18)}\\
&= e(h_2(x_v x_o B + x_v B(\tfrac{m'_w}{m_w}x_o + dm'_w - x_o)), x_v x_p B) && \text{- by Eq. (4), (5)}\\
&= e(h_2((x_o + \tfrac{m'_w}{m_w}x_o + dm'_w - x_o)Y_v), x_p Y_v) && \text{- by Eq. (4)}\\
&= e(h_2(\sigma' Y_v), x_p Y_v) && \text{- by Eq. (17)}\\
&= \Phi' && \text{- by Eq. (21)}
\end{aligned}$$

U_v then recovers m and S' by computing $m = h_3(\Phi', R') \oplus \Omega'$ and $S' = D_v(\Psi')$. Finally U_v performs the proxy signature verification process in Eq. (16), but the attack can pass the verification without being detected since

$$\begin{aligned}
& e(h_1(m, R')B + Y_p + Y_o + m'_w N', S')\\
=& e(\ h_1(m, R')B + Y_p + Y_o + \tfrac{m'_w}{m_w}Y_o + m'_w N - Y_o, && \text{- by Eq. (18)}\\
& \quad r'(h_1(m, R') + x_p + \sigma')^{-1}B) && \text{- by Eq. (20)}\\
=& e(\ (h_1(m, R') + x_p + x_o + \tfrac{m'_w}{m_w}x_o + dm'_w - x_o)B, && \text{- by Eq. (4), (5)}\\
& \quad r'(h_1(m, R') + x_p + \sigma')^{-1}B)\\
=& e(\ (h_1(m, R') + x_p + \sigma')B, && \text{- by Eq. (17)}\\
& \quad r'(h_1(m, R') + x_p + \sigma')^{-1}B)\\
=& e(B, r'B) && \text{- by Eq. (2)}\\
=& e(B, R') && \text{- by Eq. (19)}
\end{aligned}$$

4.6.3 Failure of resolving repudiation

The original signer U_o later finds out the signature from the proxy signer U_p is not delegated by him and asks a trusted judge to resolve the conflict. The judge first asks U_v to present the converted proxy signature (S', R', N') and (m, m'_w), and then performs the SRV test in Eq. (16). As described in the previous section, the verification will succeed and the judge realizes that the proxy signature was indeed signed by U_p.

Next, the judge will ask U_p to present the proxy credential he received from U_o. The malicious U_p can simply present the forged credential (σ', N', m'_w) to the judge. The forged credential can falsely fool the judge to believe U_p is innocent since the checking of the PCV test in Eq. (7) will succeed as follows:

$$\begin{aligned}
\sigma' B &= (\tfrac{m'_w}{m_w} x_o + d m'_w) B && \text{- by Eq. (17)} \\
&= \tfrac{m'_w}{m_w} Y_o + m'_w N && \text{- by Eq. (4), (5)} \\
&= Y_o + \tfrac{m'_w}{m_w} Y_o + m'_w N - Y_o \\
&= Y_o + m'_w (\tfrac{\frac{m'_w}{m_w} Y_o + m'_w N - Y_o}{m'_w}) \\
&= Y_o + m'_w N' && \text{- by Eq. (18)}
\end{aligned}$$

5 EA scheme and the PCF attack

The EA scheme is a proxy identity-based signcryption scheme. It also has the same five phases as the LWHY scheme.

5.1 Setup

Given security parameters k and n, the key generation center KGC chooses two cyclic groups $(G_1, +)$ and $(G_2, \times)$ of prime order q, a bilinear map $e : G_1 \times G_1 \to G_2$, a generator B of G_1, a master secret $s \in Z_q^*$, a secure symmetric encryption algorithm (E, D) such as AES, a system public key

$$Y_{pub} = sB \in G_1 \tag{24}$$

and collision-free hash functions

$h_1 : \{0,1\}^* \to G_1$
$h_2 : G_2 \to \{0,1\}^n$
$h_3 : \{0,1\}^* \times G_2 \to Z_q$

The KGC publishes $\{G_1, G_2, q, B, Y_{pub}, e, h_1, h_2, h_3\}$. The KGC also assigns each user U_i, with identity ID_i, a public-private key pair (Y_i, X_i) by computing

$$Y_i = h_1(ID_i) \tag{25}$$

$$X_i = sY_i \tag{26}$$

where both Y_i and $X_i \in G_1$. The key pair will be sent to the user in a secure channel.

5.2 Proxy credential generation

U_o generates a proxy credential to delegate his signing power. U_o first chooses a secret random number $d \in Z_q$ and compute

$$N = dB \tag{27}$$

$$\sigma = X_o + dY_{pub} \tag{28}$$

The proxy credential $C = (\sigma, N)$ is then sent to U_p.

5.3 Proxy credential verification

U_p verifies the validity of the received credential by performing the following PCV test.

$$e(B, \sigma) \stackrel{?}{=} e(Y_{pub}, Y_o + N) \tag{29}$$

since

$$\begin{aligned} e(B,\sigma) &= e(B, X_o + dY_{pub}) && \text{- by Eq. (28)} \\ &= e(B, sY_o + dsB) && \text{- by Eq. (26), (24)} \\ &= e(sB, Y_o + dB) && \text{- by Eq. (2)} \\ &= e(Y_{pub}, Y_o + N) && \text{- by Eq. (24), (27)} \end{aligned}$$

5.4 Signcrypted message generation

With a valid proxy credential, U_p is able to signcrypt a message m to U_v on behalf of U_o. U_p chooses a random number $r \in Z_q^*$ and computes

$$k_1 = e(B, Y_{pub})^r \tag{30}$$

$$k_2 = h_2(e(Y_{pub}, Y_v)^r) \tag{31}$$

$$c = E_{k_2}(m) \tag{32}$$

$$a = h_3(c, k_1) \tag{33}$$

$$S = rY_{pub} - a\sigma \tag{34}$$

U_p sends the signcrypted message $\delta = (c, a, S, N)$ to U_v.

5.5 Signature recovery and verification

Upon receiving the signcrypted message $\delta = (c, a, S, N)$ from U_p, U_v first recovers k_1 by computing

$$k_1 = e(B, S)e(Y_{pub}, Y_o + N)^a \tag{35}$$

since

$$\begin{aligned}
&e(B,S)e(Y_{pub}, Y_o+N)^a \\
=&e(B, rY_{pub} - a\sigma)e(sB, a(Y_o+N)) && \text{- by Eq. (34), (24), (2)}\\
=&e(B, rY_{pub} - a(X_o + dY_{pub}))e(sB, a(Y_o+N)) && \text{- by Eq. (28)}\\
=&e(B, rsB - a(sY_o + dsB))e(sB, a(Y_o+N)) && \text{- by Eq. (24), (26)}\\
=&e(sB, rB - a(Y_o+N))e(sB, a(Y_o+N)) && \text{- by Eq. (2)}\\
=&e(sB, rB) && \text{- by Eq. (3)}\\
=&e(Y_{pub}, B)^r && \text{- by Eq. (24), (2)}\\
=&e(B, Y_{pub})^r && \text{- by Eq. (1)}\\
=&k_1 && \text{- by Eq. (30)}
\end{aligned}$$

U_v then recovers k_2 by computing

$$k_2 = h_2(e(S, Y_v)e(Y_o + N, X_v)^a) \tag{36}$$

since

$$\begin{aligned}
&h_2(e(S,Y_v)e(Y_o+N, X_v)^a) \\
=&h_2(e(rY_{pub} - a\sigma, Y_v)e(Y_o+N, sY_v)^a) && \text{- by Eq. (34), (26)}\\
=&h_2(e(rY_{pub} - a\sigma, Y_v)e(as(Y_o+N), Y_v)) && \text{- by Eq. (2)}\\
=&h_2(e(rY_{pub} - a\sigma, Y_v)e(a(X_o + dsB), Y_v)) && \text{- by Eq. (26), (27)}\\
=&h_2(e(rY_{pub} - a\sigma, Y_v)e(a(X_o + dY_{pub}), Y_v)) && \text{- by Eq. (24)}\\
=&h_2(e(rY_{pub} - a\sigma, Y_v)e(a\sigma, Y_v)) && \text{- by Eq. (28)}\\
=&h_2(e(rY_{pub}, Y_v)) && \text{- by Eq. (3)}\\
=&h_2(e(Y_{pub}, Y_v)^r) && \text{- by Eq. (2)}\\
=&k_2 && \text{- by Eq. (31)}
\end{aligned}$$

Finally, U_v is able to recover the message $m = D_{k_2}(c)$ and accepts the signcrypted message δ if and only if the following SRV test holds

$$a \stackrel{?}{=} h_3(c, k_1) \tag{37}$$

5.6 The PCF attack to the EA scheme

Unlike the LWHY scheme, the EA scheme does not have a proxy warrant. Thus, the motivation of U_p to forge the proxy credential will not be extending his proxy signing power. However, a malicious U_p can forge a credential and secretly give it to another person $U_{p'}$ without the agreement of the original signer U_o. With the forged credential, $U_{p'}$ can proxy signcrypted a message to U_v on behalf of U_o, though he does not have a proper delegation from U_o.

5.6.1 U_p generates a fake proxy credential

The Malicious U_p can use his proxy credential $C = (\sigma, N)$ to generate a fake proxy credential $C' = (\sigma', N')$. U_p first picks a random number $d' \in Z_q$ and then computes

$$N' = N + d'B = dB + d'B = (d + d')B = d''B \tag{38}$$

where $d'' = d + d'$. U_p then computes the corresponding σ' by

$$\sigma' = \sigma + d'Y_{pub} = X_o + dY_{pub} + d'Y_{pub} = X_o + d''Y_{pub} \tag{39}$$

U_p can unlawfully give the fake proxy credential C' to a third person $U_{p'}$. Based on the fake C', $U_{p'}$ can randomly pick a number $r' \in Z_q^*$ and signcrypt a message $\delta' = (c', a', S', N')$ to U_v by computing Eq. (30) to Eq. (34), that is,

$$k_1' = e(B, Y_{pub})^{r'} \tag{40}$$

$$k_2' = h_2(e(Y_{pub}, Y_v)^{r'}) \tag{41}$$

$$c' = E_{k_2'}(m) \tag{42}$$

$$a' = h_3(c'.k_1') \tag{43}$$

$$S' = r'Y_{pub} - a'\sigma' \tag{44}$$

5.6.2 Pass signature recovery and verification

Upon receiving the fake signcrypted message (c', a', S', N'), U_v performs the signature recovery and verification process as described in Section 5.5. He recovers both k_1 and k_2 using Eq. (35) and Eq. (36) but with inputs from the fake δ'. The recovery will be successful since

$$\begin{aligned}
k_1 &= e(B, S')e(Y_{pub}, Y_o + N')^{a'} && \text{- by Eq. (35)}\\
&= e(B, r'Y_{pub} - a'\sigma')e(sB, Y_o + N')^{a'} && \text{- by Eq. (44), (24)}\\
&= e(B, r'Y_{pub} - a'\sigma')e(sB, a'(Y_o + N')) && \text{- by Eq. (2)}\\
&= e(B, r'Y_{pub} - a'(X_o + d''Y_{pub}))e(sB, a'(Y_o + N')) && \text{- by Eq. (39)}\\
&= e(B, r'sB - a'(sY_o + d''sB))e(sB, a'(Y_o + N')) && \text{- by Eq. (24), (26)}\\
&= e(sB, r'B - a'(Y_o + N'))e(sB, a'(Y_o + N')) && \text{- by Eq. (2)}\\
&= e(sB, r'B) && \text{- by Eq. (3)}\\
&= e(Y_{pub}, r'B) && \text{- by Eq. (24)}\\
&= e(Y_{pub}, B)^{r'} && \text{- by Eq. (2)}\\
&= e(B, Y_{pub})^{r'} && \text{- by Eq. (1)}\\
&= k_1' && \text{- by Eq. (40)}
\end{aligned}$$

and

$$
\begin{aligned}
k_2 &= h_2(e(S', Y_v)e(Y_o + N', X_v)^{a'}) && \text{- by Eq. (36)}\\
&= h_2(e(r'Y_{pub} - a'\sigma', Y_v)e(Y_o + N', sY_v)^{a'}) && \text{- by Eq. (44), (26)}\\
&= h_2(e(r'Y_{pub} - a'\sigma', Y_v)e(Y_o + N', Y_v)^{sa'}) && \text{- by Eq. (2)}\\
&= h_2(e(r'Y_{pub} - a'\sigma', Y_v)e(a's(Y_o + N'), Y_v)) && \text{- by Eq. (2)}\\
&= h_2(e(r'Y_{pub} - a'\sigma', Y_v)e(a'(X_o + sd''B), Y_v)) && \text{- by Eq. (26), (38)}\\
&= h_2(e(r'Y_{pub} - a'\sigma', Y_v)e(a'(X_o + d''Y_{pub}), Y_v)) && \text{- by Eq. (24)}\\
&= h_2(e(r'Y_{pub} - a'\sigma', Y_v)e(a'\sigma', Y_v)) && \text{- by Eq. (39)}\\
&= h_2(e(r'Y_{pub}, Y_v)) && \text{- by Eq. (3)}\\
&= h_2(e(Y_{pub}, Y_v)^{r'}) && \text{- by Eq. (2)}\\
&= k_2' && \text{- by Eq. (41)}
\end{aligned}
$$

After computing both k_1' and k_2', U_v recovers the message $m = D_{k_2'}(c')$ and performs the SRV test using Eq. (37). The fake signcrypted message $\delta' = (c', a', S', N')$ and the recovered k_1' can pass the checking $a' \stackrel{?}{=} h_3(c', k_1')$ since, by Eq. (43), that is how a' was first generated by $U_{p'}$.

5.6.3 Failure of resolving repudiation

U_o later discovers that the proxy signcrypted message generated by $U_{p'}$ does not have his proxy delegation. He will ask a trusted judge to decide who is dishonest. The judge first asks U_v to present the signcrypted message $\delta' = (c', a', S', N')$ and check the validity of the message. It will pass the SRV test as described in the previous section and the judge concludes that U_v is honestly presenting a valid proxy signcrypted message.

Next, the judge needs to decide either U_o or $U_{p'}$ is dishonest. He asks $U_{p'}$ to present the proxy credential he received from U_o. The forged credential $C' = (\sigma', N')$ presented by $U_{p'}$ will pass the PCV test in Eq. (29) and the judge will falsely believe that $U_{p'}$ is innocent since

$$
\begin{aligned}
e(B, \sigma') &= e(B, X_o + d''Y_{pub}) && \text{- by Eq. (39)}\\
&= e(B, sY_o + d''sB) && \text{- by Eq. (26), (24)}\\
&= e(sB, Y_o + d''B) && \text{- by Eq. (2)}\\
&= e(Y_{pub}, Y_o + N') && \text{- by Eq. (24), (38)}
\end{aligned}
$$

6 Prevention of PCF attacks

This paper gave two examples of the PCF attack to the LWHY and the EA schemes. The same attack can be applied to other proxy signature/signcryption schemes if they do not satisfy the proxy credential unforgeability (PCNF) property described in Section 2.2. Actually, we have found another proxy signature scheme [23] which is also insecure against the proposed PCF attack, where a malicious signature verifier can generate a fake signature on a different message m' from a signature of m. To save the page size, we do not include the discussion of this scheme in the paper.

This section discusses how to modify both the LWHY and the EA schemes to prevent the attack. In most of the proxy signature/signcryption schemes, the original signer U_o creates a proxy credential with some embedded secrets such as a random number and the U_o's private key. The proxy signer U_p can then use U_o's public key to verify that the credential is indeed constructed from U_o's private key without revealing the embedded secrets.

In both the LWHY and the EA schemes, there are two major components N and σ in the proxy credential. These two components need to match each other, i.e., only a matched pair can pass the PCV test. However, the matching relationship between N and σ is easy to re-produce in the LWHY and the EA schemes. The PCF attack presented in this paper re-produces the matching relationship to a fake pair N' and σ' from the original pair N and σ. This is why both schemes are vulnerable to the PCF attacks.

Thus, the prevention of the PCF attacks must focus on making it difficult to re-produce the matching relationship of N and σ. From Eq. (5) and (6) in the LWHY scheme and Eq. (27) and (28) in the EA scheme, we can see that both N and σ are constructed using the same random secret d and that is the only matching requirement. If we are able to add an extra matching requirement between N and σ, the proposed PCF attack can be prevented. Our approach is to add the component N into the construction of the other component σ. The detail of the modification to the LWHY and the EA schemes are described in the following two sections.

6.1 Modification to the LWHY scheme

A simple and practical modification to the LWHY scheme is to change the credential (σ, N, m_w) generation to

$$N = dB \tag{45}$$

$$\sigma = [N]_x x_o + dm_w \bmod q \tag{46}$$

where $d \in Z_q$ is a randomly picked one-time secret number and $[N]_x$ is the x-coordinate of the point $N \in G_1$. In addition, the PCV test in Eq. (7) should be changed to

$$\sigma B \stackrel{?}{=} [N]_x Y_o + m_w N \tag{47}$$

For message signcryption and signature recovery and verification, no change is required since the proposed attack only targets at the proxy credential generation.

Now, let's show how the proxy credential verification works in the modified LWHY scheme. Upon receiving a proxy credential (σ, N, m_w) from U_o, the proxy signer U_p verifies the credential using the PCV test in Eq. (47). If the test returns true, the credential is valid since

$$\begin{aligned} \sigma B &= ([N]_x x_o + dm_w)B && \text{- by Eq. (46)} \\ &= [N]_x Y_o + m_w N && \text{- by Eq. (4), (5)} \end{aligned}$$

6.1.1 Secure against the PCF attack

Proposition 1 below shows that the modified LWHY scheme is secure against the PCF attack proposed in Section 4.6.

Proposition 1. *The PCF attack proposed in Section 4.6 will not be successful in the modified LWHY scheme.*

Proof: Applying the same PCF attack to the modified scheme, the malicious proxy signer U_p generates a fake proxy warrant m'_w to extend his signing power. Based on his/her original credential (σ, N, m_w), he computes

$$\sigma' = \frac{m'_w}{m_w}\sigma = \frac{m'_w}{m_w}[N]_x x_o + dm'_w \bmod q \tag{48}$$

$$N' = \frac{\frac{m'_w}{m_w}Y_o + m'_w N - Y_o}{m'_w} \tag{49}$$

The fake credential (σ', N', m'_w), which passes the PCV test in the original LWHY scheme, will not pass the PCV test, defined in Eq. (47), in the modified LWHY scheme since

$$\begin{aligned}
\sigma' B &= (\tfrac{m'_w}{m_w}[N]_x x_o + dm'_w)B && \text{- by Eq. (48)} \\
&= \tfrac{m'_w}{m_w}[N]_x Y_o + m'_w N && \text{- by Eq. (4), (5)} \\
&= [N']_x Y_o + \tfrac{m'_w}{m_w}[N]_x Y_o + m'_w N - [N']_x Y_o \\
&= [N']_x Y_o + m'_w(\tfrac{\frac{m'_w}{m_w}[N]_x Y_o + m'_w N - [N']_x Y_o}{m'_w}) \\
&\neq [N']_x Y_o + m'_w N' && \text{- by Eq. (49)}
\end{aligned}$$ ◇

The modified scheme only changes the proxy credential generation and its verification in the LWHY scheme. The message signcryption, recovery and verification procedures are not modified. Thus, the security proofs of their scheme in [16] against IND-CCA2 (confidentiality against indistinguishability under adaptive chosen-ciphertext attacks) and EF-CMA (unforgeability against existential forgery under adaptive chosen-message attacks) can be directly applied to the modified scheme.

6.1.2 Added computational complexity

For the computational complexity, the modified LWHY scheme only adds a few operations to the credential generation and to the credential verification, where these two processes are required only at the time a new proxy delegation from U_o to U_p occurs. Comparing to the bilinear pairing operations in the more frequent signcryption generation, recovery and verification processes, the added operations are not significant. Comparing Eq. (46) to Eq. (6), the original signer U_o in the modified LWHY scheme requires to compute an extra modular multiplication $[N]_x x_o \bmod q$. Similarly, comparing Eq. (47) to Eq. (7), one

Table 1 Number of required operations for the LWHY scheme and the modified LWHY scheme, where PCG, PCV, SMG and SRV represents the proxy credential generation, proxy credential verification, signcrypted message generation and signature recovery & verification.

	Schemes	PCG	PCV	SMG	SRV
# of M_m	LWHY	**1**		1	
	Modified	**2**		1	
# of P_m	LWHY	1	**2**	4	5
	Modified	1	**3**	4	5
# of P_a	LWHY		1		4
	Modified		1		4
# of I	LWHY			1	
	Modified			1	
# of H	LWHY			3	3
	Modified			3	3
# of E/D	LWHY			1	1
	Modified			1	1
# of B_p	LWHY			1	3
	Modified			1	3

additional point multiplication $[N]_x Y_o$ in G_1 is required for the proxy signer U_p in the modified LWHY scheme. Table 1 gives the required operations in both LWHY and the modified LWHY scheme, where $M_m, H, P_m, P_a, I, E/D$ and B_p represent the modular multiplication in Z_q, hashing, point multiplication in G_1, point addition in G_1, inversion in Z_q, symmetric encryption/decryption and bilinear pairing operations, respectively. We ignore operations such as the modular addition in Z_q and bit-wise exclusive-or since their computational time is insignificant comparing to other operations. There are three kinds of hash functions used h_1, h_2 and h_3 in the LWHY scheme. They are comparable to each other in terms of efficiency. Thus, for simplicity, we just count the number of hash operations without further differentiating them in Table 1.

A blank entry in Table 1 means that no corresponding operation required in both schemes. The entries with bold numbers indicate that the modified LWHY scheme needs more operations than the LWHY scheme does. If the practical elliptic curve E/F_3163 is used to implement the group G_1, according to the best result in [24], one bilinear pairing (B_p) operation is about $11,110$ modular multiplications (M_m) in F_3163, while a point multiplication (P_m) of E/F_3163 is just a few hundred modular multiplications in F_3163. Thus, comparing to the required B_p operations in both schemes, the added one M_m and one P_m operations in the modified LWHY scheme is insignificant.

6.2 Modification to the EA scheme

To prevent the PCF attack, we also propose a modified EA scheme using the same approach used in the modified LWHY scheme. The proxy credential (σ, N) generation of the EA scheme in Eq. (27) and (28) will be modified to

$$N = dB \tag{50}$$

$$\sigma = [N]_x X_o + dY_{pub} \tag{51}$$

The corresponding PCV test in Eq. (29) also needs to change to

$$e(B, \sigma) \stackrel{?}{=} e(Y_{pub}, [N]_x Y_o + N) \tag{52}$$

The modified EA scheme does not require to change the signcrypted message generation procedure from the EA scheme. That is, the modified EA scheme still uses Eq. (30) to Eq. (34) to generate a signcrypted message.

However, the signature recovery and verification procedures of the original EA scheme do need to be modified. In the modified EA scheme, after receiving a signcrypted message (c, a, S, N) from U_p, the signature verifier U_v recovers k_1 and k_2 using Eq. (53) and Eq. (54), respectively.

$$k_1 = e(B, S)e(Y_{pub}, [N]_x Y_o + N)^a \tag{53}$$

$$k_2 = h_2(e(S, Y_v)e([N]_x Y_o + N, X_v)^a) \tag{54}$$

since

$$\begin{aligned}
& e(B, S)e(Y_{pub}, [N]_x Y_o + N)^a \\
=& e(B, S)e(sB, a([N]_x Y_o + N)) && \text{- by Eq. (24), (2)} \\
=& e(B, rY_{pub} - a\sigma)e(sB, a([N]_x Y_o + N)) && \text{- by Eq. (34)} \\
=& e(B, rsB - a([N]_x sY_o + dsB))e(sB, a([N]_x Y_o + N)) && \text{- by Eq. (24), (51)} \\
=& e(sB, rB - a([N]_x Y_o + dB))e(sB, a([N]_x Y_o + N)) && \text{- by Eq. (2)} \\
=& e(sB, rB - a([N]_x Y_o + N))e(sB, a([N]_x Y_o + N)) && \text{- by Eq. (50)} \\
=& e(sB, rB) && \text{- by Eq. (3)} \\
=& e(Y_{pub}, B)^r && \text{- by Eq. (24), (2)} \\
=& e(B, Y_{pub})^r && \text{- by Eq. (1)} \\
=& k_1 && \text{- by Eq. (30)}
\end{aligned}$$

and

$$\begin{aligned}
& h_2(e(S, Y_v)e([N]_x Y_o + N, X_v)^a) \\
=& h_2(e(rY_{pub} - a\sigma, Y_v)e([N]_x Y_o + N, sY_v)^a) && \text{- by Eq. (34), (26)} \\
=& h_2(e(rY_{pub} - a\sigma, Y_v)e([N]_x Y_o + N, Y_v)^{as}) && \text{- by Eq. (2)} \\
=& h_2(e(rY_{pub} - a\sigma, Y_v)e(as([N]_x Y_o + N), Y_v)) && \text{- by Eq. (2)} \\
=& h_2(e(rY_{pub} - a\sigma, Y_v)e(a([N]_x X_o + dsB), Y_v)) && \text{- by Eq. (26), (50)} \\
=& h_2(e(rY_{pub} - a\sigma, Y_v)e(a([N]_x X_o + dY_{pub}), Y_v)) && \text{- by Eq. (24)} \\
=& h_2(e(rY_{pub} - a\sigma, Y_v)e(a\sigma, Y_v)) && \text{- by Eq. (51)} \\
=& h_2(e(rY_{pub}, Y_v)) && \text{- by Eq. (3)} \\
=& h_2(e(Y_{pub}, Y_v)^r) && \text{- by Eq. (2)} \\
=& k_2 && \text{- by Eq. (31)}
\end{aligned}$$

Eq. (53) and Eq. (54) in the modified EA scheme are the replacements of Eq. (35) and Eq. (36) in the EA scheme. To recover the message m and verify the signcrypted message, the modified EA scheme performs the exact same procedures as the EA scheme. That is, $m = D_{k_2}(c)$ and accepts the signcrypted message if the SRV test $a \stackrel{?}{=} h_3(c, k_1)$ returns true.

6.2.1 Secure against the PCF attack

The modified EA scheme is secure against the PCF attack proposed in Section 5.6. In the PCF attack to the original EA scheme, the malicious U_p creates a fake credential (σ', N') from the original credential (σ, N) and then give it to another person $U_{p'}$ who has no proper delegation from U_o. Proposition 2 shows this attack will not be successful in the modified EA scheme.

Proposition 2. *The PCF attack proposed in Section 5.6 will not be successful in the modified EA scheme.*

Proof: Applying the same attack to the modified EA scheme, the malicious U_p picks a random number $d' \in Z_q$ and then computes

$$N' = N + d'B = dB + d'B = (d + d')B = d''B \tag{55}$$

$$\sigma' = \sigma + d'Y_{pub} = [N]_x X_o + dY_{pub} + d'Y_{pub} = [N]_x X_o + d''Y_{pub} \tag{56}$$

where $d'' = d + d'$. However, this pair of (σ', N') in the fake credential will not be matched to pass the PCV test in the modified EA scheme as defined in Eq. (52) since

$$\begin{aligned} e(B, \sigma') &= e(B, [N]_x X_o + d''Y_{pub}) && \text{- by Eq. (56)} \\ &= e(B, [N]_x sY_o + d''sB) && \text{- by Eq. (26), (24)} \\ &= e(sB, [N]_x Y_o + d''B) && \text{- by Eq. (2)} \\ &= e(Y_{pub}, [N]_x Y_o + N') && \text{- by Eq. (24), (55)} \\ &\neq e(Y_{pub}, [N']_x Y_o + N') && \text{- by Eq. (52)} \end{aligned}$$ ♢

6.2.2 Added computational complexity

The notations used in Section 6.1.2 will be used here again to describe the added computational complexity. The modified EA scheme only changes the computation of σ (Eq. (51) vs. Eq. (28)), the PCV test (Eq. (52) vs. Eq. (29)), and the recovery of k_1 and k_2 (Eq. (53) vs. Eq. (30) and Eq. (54) vs. Eq. (31), respectively). The modified EA scheme requires that each of U_o, U_p and U_v performs one extra P_m operation in the σ computation, in the PCV test, and in the recovery of k_1 and k_2, respectively. Note that U_v can compute $[N]_x Y_o$ and remember it while recovering k_1 and use it later for recovering k_2. Assume the elliptic curve E/F_3163 is used to implement the group G_1. The added three P_m operations are insignificant, comparing to the required B_p operations in both schemes since the computation complexity of one B_p operations is about

Table 2 Number of required operations for the EA scheme and the modified EA scheme, where PCG, PCV, SMG and SRV represents the proxy credential generation, proxy credential verification, signcrypted message generation and signature recovery & verification.

	Schemes	PCG	PCV	SMG	SRV
# of P_m	EA	**2**	**0**	2	**1**
	Modified	**3**	**1**	2	**2**
# of P_a	EA	1	1	1	1
	Modified	1	1	1	1
# of H	EA			2	2
	Modified			2	2
# of E/D	EA			1	1
	Modified			1	1
# of B_p	EA		2	2	4
	Modified		2	2	4

11,110 M_m operations in F_3163, whereas one P_m operation in E/F_3163 is only about a few hundred M_m operations in F_3163. To show the insignificance of the added P_m operations, Table 2 gives all required operations in both schemes.

7 Conclusion

The non-repudiation of a proxy signature/signcryption of a message, from a proxy signer to a signature verifier, is usually the main focus in many schemes. In this paper, we point out that the non-repudiation of a proxy credential, from an original signer to a proxy signer, is important as well by presenting a proxy credential forgery attack to two recent proxy signcryption schemes. The paper also proposes possible modifications to the two schemes so that the improved schemes are secure against the attack.

References

1. M. Mambo, K. Usuda, E. Okamoto, Proxy signature for delegating signature operation, Proc. of the 3rd ACM Conf. on Computer and Commun. Security, pp. 48-57 (1996).
2. M. Mambo, K. Usuda, E. Okamoto, Proxy signatures: delegation of the power to sign messages, IEICE Transactions on Fundamentals of Electronic Communications and Computer Science E79-A(9), pp. 1338-1354 (1996).
3. S. Kim, S. Park, D. Won, Proxy signatures, revisited, Proc. of ICICS'97, Springer-Verlag, pp. 223-232 (1997).
4. K. Zhang, Threshold proxy signature schemes, 1997 Information Security Workshop, Springer-Verlag, pp. 191-197 (1997).
5. N.Y. Lee, T. Hwang, C.H. Wang, On Zhang's nonrepudiable proxy signature schemes, ACISP'98, Springer-Verlag, pp. 415-422 (1998)
6. H.M. Sun, An efficient nonrepudiable threshold proxy signature scheme with known signers, Computer Commun., 22(8), pp. 717-722 (1999).
7. L. Yi, G. Bai, G.Xiao, Proxy multi-signature scheme: a new type of proxy signature scheme, Electronics Letters, 36(6), pp. 527-528 (2000).

8. C.L. Hsu, T.S. Wu, T.C. Wu, New nonrepudiable threshold proxy signature scheme with known signers, J. of Systems and Software, 58, pp. 119-124 (2001).
9. M.S. Hwang, S.F. Tzeng, C.S. Tsai, Generalization of proxy signature based on elliptic curves, Computer Standards & Interfaces, 26, pp. 73-84 (2004).
10. T.S. Chen, Y.F. Chung, K.H.Huang, A traceable proxy multisignature scheme based on the elliptic curve cryptosystem, Applied Mathematics and Computation, 159, pp. 137-145 (2004).
11. Y. Zheng, Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption), CRYPTO'97, pp. 165-179 (1997).
12. Y. Zheng, Signcryption and its applications in efficient public key solutions, Info. Security Workshop, Springer-Verlag, pp. 291-312 (1997).
13. F. Bao, R.H. Deng, A signcryption scheme with signature directly verifiable by public key, Workshop on Public Key Cryptography, Springer-Verlag, pp. 55-59 (1998).
14. X. Boyen, Multipurpose identity-based signcryption: A Swiss army knife for identity-based cryptoraphy, CRYPTO'03, pp. 383-399 (2003).
15. R.J. Hwang, C.H. Lai, F.F. Su, An efficient signcryption scheme with forward secrecy based on elliptic curve, Applied Mathematics and Computation, 167(2), pp. 870-881 (2005).
16. H.Y. Lin, T.S. Wu, S.K. Huang, Y.S. Yeh, Efficient proxy signcryption scheme with provable CCA and CMA security, Computers and Mathematics with Applications, 60(7), pp. 1850-1858 (2010).
17. H. Elkamchouchi, Y. Abouelseoud, A new proxy identity-based signcryption scheme for partial delegation of signing rights, Cryptology ePrint Archive, Report 2008/041, (2008).
18. P.S.L.M. Barreto, H.Y. Kim, B. Lynn, M. Scott, Efficient algorithms for pairing-based cryptosystems, CRYPTO'02, pp. 354-368 (2002).
19. D. Boneh, M. Franklin, Identity-based encryption from the Weil pairing, CRYPTO'01, 213-229 (2001).
20. D. Boneh, B. Lynn, H. Shacham, Short signature from the Weil pairing, Advances in Cryptography - ASIACRYPT'01, Springer-Verlag, pp. 514-532 (2001).
21. C. Gentry, A. Silverberg, Hierarchical ID-based cryptography, Advances in Cryptography - ASIACRYPT'02, Springer-Verlag, pp. 548-566 (2002).
22. F. Zhang, K. Kim, ID-based blind signature and ring signature from pairings, Advances in Cryptography - ASIACRYPT'02, Springer-Verlag, pp. 533-547 (2002).
23. Y.S. Kim, J.H. Chang, Self Proxy Signature Scheme, IJCSNS International Journal of Computer Science and Network Security, 7(2), pp. 335-338 (2007).
24. P.S.L.M. Barreto, B. Lynn, M. Scott, On the selection of pairing-friendly groups, Selected Areas in Cryptography, Springer-verlag, (2003).