International Journal of Security and Its Applications
Vol.8, No.5 (2014), pp.333-348
http://dx.doi.org/10.14257/ijsia.2014.8.5.30

Improvement on Lin et al's Strong Designated Verifier Proxy Signature Scheme for Electronic Commerce

Caixue Zhou

School of Information Science and Technology, University of Jiujiang, JiuJiang, 332005, JiangXi, P.R. China
Charlesjjjx@126.com

Abstract

Strong designated verifier proxy signature (SDVPS) enables a proxy signer to sign messages on behalf of an original signer, and only the designated verifier can be convinced of the validity of the proxy signature, and the verifier cannot prove it to any third party. It is a combination of proxy signature and strong designated verifier signature. In this paper, we define a new formal definition and a new security model of existential unforgery of SDVPS, and then analyze a recently proposed SDVPS scheme, and give four attacks to the scheme, and further we give out an improved SDVPS scheme, and prove it in the new security model. Performance analysis shows the improved scheme is efficient and suitable for electronic commerce.

Keywords: *Designated verifier proxy signature; Existential unforgery; Random oracle model; Discrete logarithm problem; Forking lemma*

1. Introduction

In 1996, Mambo et al. [1] introduced a new concept of proxy signature, in which an original signer can delegate his signing rights to a proxy signer, and then the latter can sign messages on behalf of the former. Since then, many proxy signature schemes have been proposed. Proxy signature is found to have numerous practical applications, such as in grid computing [2], in mobile agents [3] et al.. To categorize delegation types, Mambo, et al. [1] defined three levels of delegation: (1) Full delegation. The original signer gives his secret key to the proxy signer. The proxy signer uses the key to sign documents. (2) Partial delegation. The original signer generates a delegation key from its private key and gives it to the proxy signer. The proxy signer generates a new proxy signature key from the proxy signer's private key and the delegation key. (3) Delegation by warrant. The original signer signs a warrant which describes relative rights and information of the original signer and the proxy signer. The final proxy signature includes two parts: one is the signed warrant, and another is the proxy signature produced by the proxy signer. Proxy signature can be constructed for each of these delegation types, and the most suitable scheme should be selected depending on the user's need for security, message length and signer's and/or verifier's computational ability. The partial delegation and the delegation by warrant are more secure than the full delegation because the created proxy signature is distinguishable from the original signer's signature. The partial delegation is faster than the delegation by warrant, because in the delegation by warrant, the verifier must verify two signatures. Due to its merits of partial delegation, Mambo et al. refer to partial delegation proxy signature as proxy signature.

However, in partial delegation, the proxy signer can abuse his delegated rights because partial delegation does not restrict the proxy signer's signing capability. To overcome this weakness, in 1997, Kim [4] introduced a new kind of proxy signature called partial delegation with warrant. This kind of proxy signature combines the benefits of both the partial delegation and the delegation by warrant, so this delegation has fast processing speed and is appropriate for the restricting documents to be signed. Since then, most work on proxy signature has focused on the partial delegation with warrant proxy signature. In this paper, we simply refer to partial delegation with warrant proxy signature as proxy signature if it does not lead to any confusion.

In Eurocrypt 1996, Jakobsson et al. [5] introduced another new concept of designated verifier signature (DVS). Ordinary signature can convince any person of its validity, while in DVS scheme, only the designated person can be convinced of the validity of the signature, and he can not prove this to any third party, because he can also produce an indistinguishable transcript of the signature. DVS scheme provides authentication of a message without having the non-repudiation property of ordinary signature. DVS scheme is very suitable in the scenario where a signer wishes to keep privacy of his identity to other parties but not to the designated verifier. For example, a merchant and a customer negotiate for a best price of some goods, and the merchant does not want to leave evidence of the negotiation, then he can use DVS scheme. However, DVS scheme has a weakness that any person can verify it, and can make sure there are only two potential signers. Hence, if the signature is eavesdropped before arriving at its destination, then one can identify the signer. Later, in 2003, Saeednia et al. [6] introduced the concept of strong designated verifier signature (SDVS) to overcome this weakness by forcing the designated verifier to use his private key at the time of verification.

To combine the merits of proxy signature and designated verifier signature, Dai et al. [7] in 2003 proposed a designated verifier proxy signature (DVPS) scheme, in which a proxy signer can produce a DVPS on behalf of an original signer, and only the designated verifier can be convinced of the validity of the proxy signature, and he cannot prove it to any third party. Take the above negotiation for example, a company agent can be delegated by the company to sign with the customer using a DVPS scheme. After Dai et al.'s work, Wang [8] in 2004 pointed out Dai et al.'s scheme is not secure and improved it. Li et al. [9] proposed a DVPS scheme from bilinear pairings. Later, in 2005, Wang [10] proposed another two DVPS schemes. Huang et al. [11] proposed a short DVPS scheme from bilinear pairings, and they gave the security model of a DVPS scheme. Lu et al. [12] first proposed a DVPS scheme with message recovery. Cao et al. [13] first proposed three identity-based DVPS schemes. Since then, many other DVPS and SDVPS schemes have been proposed [14-16].

Regarding the security model of unforgeability of DVPS and SDVPS schemes, Huang et al. [11] were the first involved in this field. They defined three types of adversary, the adversary can request SDVPS queries and SDVPS verification queries, at last if he can forge a valid SDVPS on a new message, then he wins the game. Later, Shim [14], Lin et al. [15] and Hsu et al. [16] also gave the security notion of unforgeability of DVPS and SDVPS schemes. But all these security models do not consider the whole abilities of the attacker. For example, after getting a valid SDVPS, the attacker may forge a valid standard signature from the SDVPS.

In 2003, Boldyreva et al. [17] first gave the formal definition and security notion of proxy signature, since then, proxy signature can be proved to be secure rigorously. Later, based on Boldyreva et al.'s work, Herranz et al. [18] gave the security model of fully distributed proxy signature. Malkin et al. [19] gave the security model of hierarchical proxy signature. In 2004, Wang et al. [20] pointed out Boldyreva et al's security model has two security flaws and improved it. In 2008, Schuldt et al. [21] gave a stronger security model, in which when an attacker registers a public key, he does not need to provide the knowledge of knowing the corresponding private key.

In this scenario, we must consider the public key replacement attack, i.e., an attacker constructs a public key and then requests a certificate from CA (Certificate Authority) without knowing the corresponding private key. In practical application, CA may not require a user to provide the knowledge of knowing the corresponding private key when the user registers a public key, due to the efficiency or some other reasons. This kind of attack also exists in multi-signature scheme. Based on Boldyreva et al.'s work, the security models of identity-based proxy signature [22], certificateless proxy signature [23] were also proposed.

In this paper, following Boldyreva et al.'s work, and referencing Wang et al.'s work [20] and Schuldt et al.'s work [21], we give out the security model of SDVPS. In our security model, besides all the above considerations, we also consider the following scenario: note in a proxy signature, where an original signer delegates his signing rights to a proxy signer is realized by means of a signature on the warrant, the proxy signer also uses a signature scheme when he produces the proxy signature, the two signature schemes can be different, we refer to the former as signature scheme 1 and the latter as signature scheme 2. In all previous security models of proxy signature listed above, if an attacker can forge a standard signature of scheme 1, then the scheme is broken. In fact, if an attacker can forge a standard signature of scheme 2, the scheme is also broken. For example, Lee et al. [24] attacked Zhang's [25] scheme, that is, the attacker can forge an original signer's standard signature of scheme 2; Li et al [26] attacked Lu et al's [27] scheme, that is, the original signer can forge a proxy signer's standard signature of scheme 2. So, in our security model, the attacker can also access the standard signing oracle of scheme 2 besides the standard signing oracle of scheme 1. This oracle may help the attacker to attack the scheme.

In 2012, Lin et al [15] proposed a SDVPS scheme. Compared with related schemes, their scheme has not only shorter signature length, but also lower computational costs, and they proved their scheme is unforgeable in the random oracle model. In this paper, we show four attacks on their scheme, then improve it, and give new security proof of the revised scheme in our new security model. Here, we must point out that our attacks can work well in Lin et al.'s original security model.

The rest of the paper is organized as follows. In section 2, we give out some preliminaries, and the new security model of SDVPS. In section 3, we first review Lin et al.'s SDVPS scheme, and then give out four attacks on the scheme. In section 4, we propose an improved SDVPS scheme. In section 5, we give out the security proof and efficiency analysis of the improved SDVPS scheme. We conclude our work in section 6.

2. Preliminaries

In this section, we review some security notions along with the computational assumptions.

Discrete Logarithm Problem (DLP). Let p and q be two large primes satisfying $q \mid p-1$, and g be a generator of order q over $GF(p)$. The discrete logarithm problem is, given an instance (y,p,q,g), where $y = g^x \bmod p$ for some $x \in Z_q$, to compute x.

Discrete Logarithm Assumption. A probabilistic polynomial-time algorithm B is said to (t,ε)-break the DLP instance (y,p,q,g), where $y = g^x \bmod p$ for some $x \in Z_q$, B can compute x with probability ε after running at most t steps.

Definition 1. The (t,ε)-DL assumption holds if there is no probabilistic polynomial-time adversary that can (t,ε)-break the DLP.

2.1. Framework of SDVPS

A SDVPS scheme consists of the following ten algorithms.

G: **Setup**. Given a security parameter k, the algorithm generates system's public parameters $params$ and user's public/secret key pair. The original signer's key pair is (pk_0,sk_0), the proxy signer's key pair is (pk_p,sk_p), the designated verifier's key pair is (pk_v,sk_v).

S_1: **Standard signature of scheme 1**. It takes as input the message m, the private key of the signer and produces a signature σ_1.

S_2: **Standard signature of scheme 2**. It takes as input the message m, the private key of the signer and produces a signature σ_2.

V_1: **Standard signature verification of scheme 1**. It takes as input (pk,σ_1), and outputs 0 or 1. If it outputs 1, then the signature σ_1 is valid.

V_2: **Standard signature verification of scheme 2**. It takes as input (pk,σ_2), and outputs 0 or 1. If it outputs 1, then the signature σ_2 is valid.

D: **Delegation**. The original signer first produces a warrant m_w, which includes the types of delegated messages, the identities and public keys of the original signer and the proxy signer, and valid period of delegating and so on, and then produces a signature $cert$ on the warrant m_w using standard signature scheme 1, and transfers $(m_w,cert)$ to the proxy signer.

P: **Proxy key generation**. The proxy signer verifies the validation of the signature $(m_w,cert)$ by using the public key of the original signer pk_o. If it's OK, then the proxy signer accepts the warrant m_w, and produces a proxy public/private key pair (pkp,skp) by using the warrant m_w and his private key sk_p, and keeps skp private; otherwise the signature $(m_w,cert)$ must be reproduced.

PS: **SDVPS generation**. It takes as input system parameters $params$, the warrant m_w, the message m, the proxy private key skp, and the public key of designated verifier pk_v, it generates an SDVPS $p\sigma$ using standard signature scheme 2.

PV: **SDVPS verification**. It takes as inputs $(pk_0,pk_p,sk_v,m_w,m,p\sigma)$, and outputs 0 or 1. If it outputs 1, then the signature $p\sigma$ is valid.

TS: **Transcript simulation**. It takes as input system parameters $params$, the warrant m_w, the message m, the proxy public key pkp, and the private key of the designated verifier sk_v. It generates an SDVPS $p\sigma$ that is indistinguishable from the proxy signer.

To the above algorithms, we require

$$\Pr[V_1(pk,m,S_1(sk,m))=accept]=1,$$

$$\Pr[V_2(pk,m,S_2(sk,m))=accept]=1,$$

$$\Pr[PV(pk_o,pk_p,sk_v,m_w,m,PS(skp,pk_v,m_w,m))=accept]=1,$$

$$\Pr[PV(pk_o,pk_p,sk_v,m_w,m,TS(sk_v,pk_0,pk_p,m_w,m))=accept]=1.$$

2.2. Security model of SDVPS

A SDVPS scheme must satisfy unforgeability, non-transferability, and strong privacy of signer's identity [15].

Boldyreva et al. [17] first gave the formal definition and security notion of proxy signature. In their security model of existential unforgery under adaptive chosen message attack, there are totally n participants and the attacker can corrupt with the original signer or the proxy signer. Consider an extreme case in which the adversary is working against a single honest user, say user 1, the adversary can play the role of user $i \neq 1$ in executions of the proxy-designation protocol with user 1, as original signer or proxy signer, and they allow the adversary request user 1 to run the proxy-designation protocol with itself, and can see the transcript of the execution, and they do not assume the existence of a secure channel between an original signer and a proxy signer. The adversary can access a standard signing oracle and a proxy signing oracle. The adversary's goal is to pretend to be an original signer or a proxy signer to forge a standard signature or a proxy signature.

In 2004, Wang et al. [20] pointed out Boldyreva et al's security model has two security flaws. First, in Boldyreva et al's security model, if user 1 never delegates user i as a proxy signer, but eventually an adversary can forge a proxy signature by user i on behalf of user 1, the proxy signature is broken. Besides this situation, Wang et al. pointed out, if user 1 has delegated user i as a proxy signer with some warrants which are put into a list $Warr_1$, but an adversary can forge a proxy signature by user i on behalf of user 1 with a new warrant $m_w^{'} \notin Warr_1$, the scheme is also broken. Second, in Boldyreva et al's security model, user i can delegate user 1 with n different warrants which are put into a list $Warr_i$, then an adversary can query user 1's proxy signature on any message m conforming to the restriction in warrant $m_w \in Warr_i$, the query (m_w, m) are added in a query list Que_i. If the adversary can forge a proxy signature by user 1 on behalf of user i on a message $m^{'}$ which never appears in Que_i, the scheme is broken. Besides this situation, Wang et al. pointed out if an adversary can forge a proxy signature on $(m_w^{'}, m^{'})$ where $m^{'}$ has appeared in Que_i, but $(m_w^{'}, m^{'})$ is not in the Que_i list, the scheme is also broken.

Now, we give our new security model of existential unforgery under adaptive chosen message and adaptive chosen warrant attack of SDVPS as follows.

Definition 2. (Unforgeability). A SDVPS scheme is said to be existential unforgery under adaptive chosen message attack and adaptive chosen warrant attack, if no probabilistic polynomial-time adversary A has a non-negligible advantage in the following game.

(1) Setup. The challenge C runs algorithm G to generate the system public parameters, the public/private key pair (pk_1, sk_1) of user 1, and the public/private key pair (pk_v, sk_v) of the designated verifier U_v. C keeps sk_1, sk_v private.

(2) i registers pk_i. A can request to register public key $pk_i, i = 2, ..., n$, and need not to provide the knowledge of knowing the corresponding private key.

(3) i designates 1 ($i = 2, ..., n$). A chooses a warrant m_{w_i}, produces a signature $cert_i$ on the warrant m_{w_i}, and sends $(m_{w_i}, cert_i)$ to C publicly. C checks the validation of it, if it is OK, then C puts $(m_{w_i}, cert_i)$ into a list $Warrp$.

(4) 1 designates i ($i = 1, 2, ..., n$)(including self-delegation). A chooses a warrant m_{w_i}, C runs algorithm D by using the private key sk_1 to produce a signature $cert_i$ on the warrant m_{w_i}, and sends $(m_{w_i}, cert_i)$ to A publicly. If it is user 1's self-delegation, C puts $(m_{w_i}, cert_i)$ into list $Warrs$; otherwise C puts $(m_{w_i}, cert_i)$ into list $Warro$.

(5) Standard signature queries of scheme 1. A chooses a message m, C runs algorithm S_1 by using the private key sk_1 to produce a signature σ_1 on the message m, returns σ_1 to A, and puts m to the list S_{1-qu}.

(6) Standard signature queries of scheme 2. A chooses a message m, C runs algorithm S_2 by using the private key sk_1 to produce a signature σ_2 on the message m, returns σ_2 to A, and puts m to the list S_{2-qu}.

(7) Proxy signature queries. The original signer is user $i \in \{1,2,...,n\}$ (including self-delegation), the proxy signer is user 1, and the designated verifier is user U_v. A chooses a warrant m_w and a message m, it must satisfy $m_w \in \{Warrp \cup Warrs\}$, and m suits m_w. C runs algorithm P to produce proxy private key skp, and runs algorithm PS to produce a proxy signature $p\sigma$, C then returns $p\sigma$ to A, and puts (m_w, m) to the list PS_{qu}.

(8) Proxy signature verification queries. A can request a proxy signature verification on a tuple $(m, m_w, p\sigma)$, where user 1 can be an original signer or a proxy signer, and user U_v is the designated verifier. C runs algorithm PV and returns 1 if it is correct, or 0 otherwise.

(9) Proxy signature simulation queries. User 1 can be the original signer or the proxy signer, and user U_v is the designated verifier. A chooses a warrant m_w and a message m, it must satisfy $m_w \in Warro$ (user 1 being the original signer) or $m_w \in \{Warrp \cup Warrs\}$ (user 1 being the proxy signer), and m suits m_w. C runs algorithm TS to produce a proxy signature $p\sigma$, returns $p\sigma$ to A, and puts (m_w, m) to the list TS_{qu}.

The adversary A wins the game if any one of the following events occurs:

E_1: A outputs a forgery (m', σ), where $V_1(pk_1, m', \sigma) = 1$, and $m' \notin S_{1-qu}$, i.e., A forges a user 1's standard signature of scheme 1.

E_2: A outputs a forgery (m', σ), where $V_2(pk_1, m', \sigma) = 1$, and $m' \notin S_{2-qu}$, i.e., A forges a user 1's standard signature of scheme 2.

E_3: A outputs a forgery $(m'_w, m', p\sigma)$, where $PV(pk_i, pk_1, sk_v, m'_w, m', p\sigma) = 1$, $(i \in \{1,2,...,n\})$, and $(m'_w, m') \notin (PS_{qu} \cup TS_{qu})$, i.e., A forges a proxy signature by user 1 on behalf of user $i \in \{1,2,...,n\}$, and user U_v is the designated verifier, including self-delegation.

E_4: A outputs a forgery $(m'_w, m', p\sigma)$, where $PV(pk_1, pk_i, sk_v, m'_w, m', p\sigma) = 1$, $(i \in \{2,...,n\})$, and $m'_w \notin Warro$, i.e., A forges a proxy signature by user $i \in \{2,...,n\}$ on behalf of user 1, and user U_v is the designated verifier.

Definition 3. (Non-Transferability [15]). A SDVPS scheme is said to achieve the security requirement of non-transferability if a designated verifier can simulate a computationally indistinguishable transcript intended for himself with his private key.

Definition 4. (Strong Privacy of Signer's Identity [15]). A SDVPS scheme satisfies the security requirement of strong privacy of signer's identity if there is no probabilistic polynomial-time adversary having the ability to determine the identity of signer for an intercepted SDVPS by performing the signature verification process before the SDVPS has been received by the designated verifier.

3. Lin et al.'s strong designated verifier proxy signature scheme and our attacks

3.1. Review of Lin et al.'s scheme

Lin et al's scheme consists of the following four phases.

- **Setup:** Taking as input 1^k, the system authority (SA) selects two large primes p and q where $|q| = k$ and $q \mid (p-1)$. Let g be a generator of order q and $h_1 : \{0,1\}^k \times Z_q \to Z_q$,

$h_2:\{0,1\}^k \times Z_q \to Z_q$, $h_3: Z_q \to Z_q$ be three collision resistant hash functions. The system public parameters are $params=\{p,q,g,h_1,h_2,h_3\}$. Each user U_i chooses his private key $x_i \in Z_q$ and computes his public key as $y_i = g^{x_i}$.

- **Proxy Credential Generation:** Let U_0 be an original user delegating his signing power to a proxy signer U_p. U_0 first chooses $d \in_R Z_q$ to compute

$$T = (g^d \bmod p) \bmod q, \qquad (1)$$
$$\sigma = d - x_0 h_1(m_w, T) \bmod q, \qquad (2)$$

Where m_w is a warrant consisting of the identifiers of original and proxy signer, the delegation duration and so on. (σ, m_w, T) is then sent to U_p. Upon receiving (σ, m_w, T), U_p computes Z as Eq. (3) and performs Eq. (4) to check its validity.

$$Z = y_0^{h_1(m_w,T)} \bmod p, \quad (3)$$
$$T = g^{\sigma} Z \bmod p \pmod q. \qquad (4)$$

If it does not hold, (σ, m_w, T) is requested to be sent again.

- **Proxy Signature Generation:** For signing a message $m \in_R \{0,1\}^*$ on behalf of the original signer U_0, U_p chooses $w \in_R Z_q$ to compute

$$s_1 = h_3((y_v^w \bmod p) \bmod q), \qquad (5)$$
$$s_2 = w - (x_p + \sigma) h_2(m, T) \bmod q, \qquad (6)$$

and then delivers (m, m_w) along with the SDVPS $\delta = (s_1, s_2, T)$ to a designated recipient U_v.

- **Proxy Signature Verification:** Upon receiving (m, m_w) and $\delta = (s_1, s_2, T)$, U_v first computes (R_1, R_2) as follows:

$$R_1 = y_v^{s_2} \bmod p, \qquad (7)$$
$$R_2 = (T y_p y_0^{-h_1(m_w,T)})^{x_v h_2(m,T)} \bmod p. \qquad (8)$$

U_v then verifies the proxy signature by checking if

$$s_1 = h_3((R_1 R_2 \bmod p) \bmod q). \qquad (9)$$

If it holds, the SDVPS $\delta = (s_1, s_2, T)$ for m is valid.

3.2. Cryptanalysis of Lin et al's scheme

Lin et al. proved that their scheme is existentially unforgeable under adaptive chosen message attack. However, we found this is not the fact. In the following, we will give four attacks, and these attacks can work well in Lin et al. [15]'s original security model.

Attack1: The original signer U_0 can forge a valid proxy signature. First, U_0 intercepts a valid proxy signature (m, m_w, s_1, s_2, T) generated by U_p on behalf of U_0, then U_0 generates a new warrant m_w', computes $\sigma' = d - x_0 h_1(m_w', T) \bmod q$, $s_2' = s_2 + \sigma h_2(m,T) - \sigma' h_2(m,T) \bmod q$, then (m, m_w', s_1, s_2', T) is a valid proxy signature being likely to be generated by the proxy signer U_p on behalf of U_0. The following shows (m, m_w', s_1, s_2', T) can pass the verification equation. The designated receiver U_v computes $R_1 = y_v^{s_2'} \bmod p$, $R_2 = (T y_p y_o^{-h_1(m_w', T)})^{x_v h_2(m,T)} \bmod p$, then

$R_1R_2 = y_v^{s_2'}(Ty_p y_o^{-h_1(m_w',T)})^{x_v h_2(m,T)} \bmod p = y_v^{s_2'}(g^{\sigma'} g^{x_p})^{x_v h_2(m,T)} \bmod p = y_v^{s_2'} y_v^{(\sigma' + x_p)h_2(m,T)} \bmod p = y_v^{s_2'}$ $\cdot y_v^{(w-s_2')} \bmod p = y_v^w \bmod p$, so $s_1 = h_3((y_v^w \bmod p) \bmod q) = h_3((R_1R_2 \bmod p) \bmod q)$.

Attack2: An adversary U_a, first forges a warrant m_w', which records himself U_a being the original signer and U_p being the proxy signer, then he chooses two random number $t, w \in_R Z_q$, and computes $T = (y_p)^{-1} g^t \bmod p \, (\bmod q)$, $s_1 = h_3((y_v^w \bmod p) \bmod q)$, $s_2 = w - (t - x_a h_1(m_w',T))h_2(m,T) \bmod q$. Finally, the forged proxy signature is (m, m_w', s_1, s_2, T), and the U_a is the original signer, and the U_p is the proxy signer. The following shows (m, m_w', s_1, s_2, T) can pass the verification equation. The designated receiver U_v, computes $R_1 = y_v^{s_2} \bmod p$, $R_2 = (Ty_p y_a^{-h_1(m_w',T)})^{x_v h_2(m,T)} \bmod p$, then

$R_1R_2 = y_v^{s_2}(Ty_p y_a^{-h_1(m_w',T)})^{x_v h_2(m,T)} \bmod p = y_v^{s_2}(g^t g^{-x_a h_1(m_w',T)})^{x_v h_2(m,T)} \bmod p = y_v^{s_2 + (t - x_a h_1(m_w',T))h_2(m,T)}$ $\bmod p = y_v^w \bmod p$, so $s_1 = h_3((y_v^w \bmod p) \bmod q) = h_3((R_1R_2 \bmod p) \bmod q)$.

Attack3: An adversary U_a, first forges a warrant m_w', which records himself U_a being the original signer and U_p being the proxy signer, then he chooses three random number $w, t, d' \in_R Z_q$, and computes $T = g^{d'} \bmod p \, (\bmod q)$, then he registers his public key in CA as $y_a = (y_p^{-1} g^t)^{-h_1(m_w',T)^{-1}} \bmod p$, and computes proxy signature as follows, $s_1 = h_3((y_v^w \bmod p) \bmod q)$, $s_2 = w - (d' + t)h_2(m,T) \bmod q$. Finally the forged proxy signature is (m, m_w', s_1, s_2, T), and the U_a is the original signer, the U_p is the proxy signer. The following shows (m, m_w', s_1, s_2, T) can pass the verification equation. The designated receiver U_v, computes $R_1 = y_v^{s_2} \bmod p$, $R_2 = (Ty_p y_a^{-h_1(m_w',T)})^{x_v h_2(m,T)} \bmod p$, then

$R_1R_2 = y_v^{s_2}(Ty_p y_a^{-h_1(m_w',T)})^{x_v h_2(m,T)} \bmod p = y_v^{s_2}(g^{d'} g^t)^{x_v h_2(m,T)} \bmod p = y_v^{s_2 + (d'+t)h_2(m,T)} \bmod p = y_v^w \bmod p$, so $s_1 = h_3((y_v^w \bmod p) \bmod q) = h_3((R_1R_2 \bmod p) \bmod q)$.

Attack4: An adversary U_a, first forges a warrant m_w', which records U_0 being the original signer and himself U_a being the proxy signer, then he chooses three random number $w, t, d' \in_R Z_q$, and computes $T = g^{d'} \bmod p \, (\bmod q)$, then he registers his public key in CA as $y_a = y_0^{h_1(m_w',T)} g^t \bmod p$, and computes proxy signature as follows, $s_1 = h_3((y_v^w \bmod p) \bmod q)$, $s_2 = w - (d' + t)h_2(m,T) \bmod q$. Finally, the forged proxy signature is (m, m_w, s_1, s_2, T), and the U_0 is the original signer, the U_a is the proxy signer. The following shows (m, m_w', s_1, s_2, T) can pass the verification equation. The designated receiver U_v, computes $R_1 = y_v^{s_2} \bmod p$, $R_2 = (Ty_a y_0^{-h_1(m_w',T)})^{x_v h_2(m,T)} \bmod p$, then

$R_1R_2 = y_v^{s_2}(Ty_a y_0^{-h_1(m_w',T)})^{x_v h_2(m,T)} \bmod p = y_v^{s_2}(g^{d'} g^t)^{x_v h_2(m,T)} \bmod p = y_v^{s_2 + (d'+t)h_2(m,T)} \bmod p = y_v^w \bmod p$, so $s_1 = h_3((y_v^w \bmod p) \bmod q) = h_3((R_1R_2 \bmod p) \bmod q)$.

4. Our Improvement

To resist the above attacks, we propose an improved scheme in the following. It consists of six phases. Setup phase is the same as original scheme.

- **Proxy Credential Generation:** Let U_0 be an original user delegating his signing power to a proxy signer U_p. U_0 first chooses $d \in_R Z_q$ to compute

$$T = (g^d \bmod p) \bmod q, \qquad (1)$$

$$\sigma = d - x_0 h_1(m_w, T, y_0) \bmod q, \qquad (2)$$

Where m_w is a warrant consisting of the identifiers and public keys of original signer and proxy signer, the delegation duration and so on. (σ, m_w, T) is then sent to U_p publicly. Upon receiving (σ, m_w, T), U_p computes Z as Eq. (3) and performs Eq. (4) to check its validity.

$$Z = y_0^{h_1(m_w, T, y_0)} \bmod p, \qquad (3)$$

$$T = g^{\sigma} Z \bmod p \pmod q . \qquad (4)$$

If it does not hold, (σ, m_w, T) is requested to be sent again.

- **Proxy Key Generation:** The proxy signer computes his proxy private key $sk_p = \sigma + x_p h_1(m_w, T, y_p) \bmod q$, the corresponding proxy public key is $pk_p = g^{sk_p} = T y_0^{-h_1(m_w, T, y_0)} y_p^{h_1(m_w, T, y_p)}$.
- **Proxy Signature Generation:** For signing a message $m \in_R \{0,1\}^*$ on behalf of the original signer U_0, and user U_v is the designated receiver. U_p chooses $w \in_R Z_q$ to compute

$$s_1 = h_3((y_v^w \bmod p) \bmod q), \ s_2 = w - sk_p h_2(m, T) \bmod q,$$

and then delivers (m, m_w) along with the SDVPS $\delta = (s_1, s_2, T)$ to the designated receiver U_v.

- **Proxy Signature Verification:** Upon receiving (m, m_w) and $\delta = (s_1, s_2, T)$, U_v first computes (R_1, R_2) as follows:

$$R_1 = y_v^{s_2} \bmod p, \ R_2 = pk_p^{x_v h_2(m,T)} = (T y_p^{h_1(m_w, T, y_p)} y_0^{-h_1(m_w, T, y_0)})^{x_v h_2(m,T)} \bmod p .$$

U_v then verifies the proxy signature by checking if $s_1 = h_3((R_1 R_2 \bmod p) \bmod q)$. If it holds, the SDVPS $\delta = (s_1, s_2, T)$ for m is valid.

- **Transcript Simulation:** After receiving a SDVPS (m, m_w, s_1, s_2, T), the designated verifier U_v cannot prove to any third party that the signature is generated by the proxy signer U_p, because he can also generate a computationally indistinguishable SDVPS as follows:

First, U_v randomly chooses $s_2^* \in Z_q$, and computes $R_1 = y_v^{s_2^*} \bmod p$, $R_2 = (T y_p^{h_1(m_w, T, y_p)} y_0^{-h_1(m_w, T, y_0)})^{x_v h_2(m,T)} \bmod p$, $s_1^* = h_3((R_1 R_2 \bmod p) \bmod q)$. Then $(m, m_w, s_1^*, s_2^*, T)$ is the simulated signature.

5. Security proof and comparison

5.1. Security of the improved scheme

We now analyze the unforgeability, non-transferability, and strong privacy of signer's identity of our improved scheme.

Theorem 1 For a security parameter k, in the random oracle model, under adaptive chosen message attack and adaptive chosen warrant attack, if a PPT attacker A has a non-negligible advantage to forge the improved SDVPS scheme, then there exists an adversary B, who can break DLP with a non-negligible advantage in polynomial time.

Proof: We show how to build an algorithm B that solves DLP by running the adversary A as a subroutine.

Given the public key $y_1 = g^{x_1} \bmod p$ of user 1, the adversary A registers other users's public keys, $y_2 = g^{x_2} \bmod p$,…, $y_n = g^{x_n} \bmod p$, and he is allowed not to know the corresponding private keys. The objective of B is by being given the public key y_1 of user 1 to obtain x_1 to solve the DLP, or by being given the public key $pk_p = g^{sk_p} = Ty_0^{-h_1(m_w,T,y_0)} y_p^{h_1(m_w,T,y_p)}$ of proxy signer to obtain sk_p to solve the DLP. B chooses $x_v \in_R Z_q$ to set $y_v = g^{x_v} \bmod p$, and (x_v, y_v) is as the designated verifier's key pair. Then B maintains ten lists $Warrp, Warro, Warrs, S_{1-qu}, S_{2-qu}, PS_{qu}, TS_{qu}, L_1, L_2, L_3$ while A is running. B plays the role of A's challenger and works by interacting with A in a game defined as follow:

(1) H_1 **queries:**

When a tuple (m_w, T, y_i) is submitted to the H_1 oracle, B first scans a list L_1 to check whether H_1 is already defined for that input. If it is, the previously defined value is returned. Otherwise, B picks a random $h_1 \in Z_q$, stores the tuple (m_w, T, y_i, h_1) in the list L_1, and returns h_1 to A.

(2) H_2 **queries:**

When a tuple (m, T) is submitted to the H_2 oracle, B first scans a list L_2 to check whether H_2 is already defined for that input. If it is, the previously defined value is returned. Otherwise, B picks a random $h_2 \in Z_q$, stores the tuple (m, T, h_2) in the list L_2, and returns h_2 to A.

(3) H_3 **queries:**

B searches (u, h_3) in the list L_3. If such a pair is found, B returns h_3 to A; otherwise B picks a random $h_3 \in Z_q$, stores the tuple (u, h_3) in the list L_3, and returns h_3 to A.

(4) i **designates 1 (** $i = 2,\ldots, n$ **):**

A produces a proxy credential (σ_i, m_{w_i}, T_i). B queries the H_1 oracle to get the tuple $(m_{w_i}, T_i, y_i, h_{1i})$, computes $Z = y_i^{h_{1i}} \bmod p$, and checks whether $T_i \overset{?}{=} (g^{\sigma_i} Z \bmod p) (\bmod q)$. If it is OK, B accepts it and puts (σ_i, m_{w_i}, T_i) into the list $Warrp$.

(5) 1 designates i **(** $i = 1,\ldots, n$ **) (including self-delegation):**

A chooses a warrant m_{w_i}, B proceeds as follows. B first randomly chooses $\sigma_i, h_{1i} \in Z_q$, computes $T_i = (g^{\sigma_i} y_1^{h_{1i}} \bmod p) (\bmod q)$, and sets $H_1(m_{w_i}, T_i, y_1) = h_{1i}$. If the list L_1 already contains a tuple $(m_{w_i}, T_i, y_1, h_{1i})$, then B repeats the process with another $\sigma_i, h_{1i} \in Z_q$. B adds the tuple $(m_{w_i}, T_i, y_1, h_{1i})$ into L_1, and returns (σ_i, m_{w_i}, T_i) to A. If it is user 1's self-delegation, B puts (σ_i, m_{w_i}, T_i) into list $Warrs$; else B puts (σ_i, m_{w_i}, T_i) into list $Warro$.

(6) Standard signature queries of scheme 1:

User U_1 is the signer. A chooses a message m, B proceeds as follows. B first randomly chooses $\sigma, h_1 \in Z_q$, computes $T = (g^{\sigma} y_1^{h_1} \bmod p) (\bmod q)$, and sets $H_1(m, T, y_1) = h_1$. If the list

L_1 already contains a tuple (m,T,y_1,h_1), then B repeats the process with another $\sigma,h_1 \in Z_q$. B adds the tuple (m,T,y_1,h_1) into L_1, and returns (σ,m,T) to A, and it appears as a valid signature from the A's point of view. At last, B puts m to the list S_{1-qu}.

(7) Standard signature queries of scheme 2:

User U_1 is the signer, and user U_v is the designated verifier. A chooses a message m, B proceeds as follows. B first randomly chooses $s_2,T \in Z_q$, retrieves h_2 from list L_2 by querying H_2 on (m,T), and computes $R_1 = y_v^{s_2} \bmod p$, $R_2 = y_1^{x_v h_2} \bmod p$, $s_1 = h_3((R_1R_2 \bmod p) \bmod q)$. Then B returns (m,s_1,s_2,T) to A, and it appears as a valid signature from the A's point of view. At last, B stores m to the list S_{2-qu}.

(8) Proxy signature queries:

The original signer is user $i \in \{1,2,...,n\}$ (including self-delegation), the proxy signer is user 1, and the designated verifier is user U_v. A chooses a warrant m_{w_i} and a message m_i, then asks B to produce a proxy signature . It must satisfy $m_{w_i} \in \{Warrp \bigcup Warrs\}$, and m_i suits m_{w_i}. B searches m_{w_i} in the list $Warrp \bigcup Warrs$ to get the corresponding (σ_i,T_i), retrieves h_{1i} from list L_1 by querying H_1 on (m_{w_i},T_i,y_1), retrieves h_{2i} from list L_2 by querying H_2 on (m_i,T_i), randomly chooses $s_{2i} \in Z_q$, and computes $R_1 = y_v^{s_{2i}} \bmod p$, $R_2 = (g^{\sigma_i} y_1^{h_{1i}})^{x_v h_{2i}} \bmod p$, $s_{1i} = h_3((R_1R_2 \bmod p) \bmod q)$. Then, B returns $(m_i,m_{w_i},s_{1i},s_{2i},T_i)$ to A, and it appears as a valid proxy signature from the A's point of view. At last, B puts (m_{w_i},m_i) to the list PS_{qu}.

(9) Proxy signature verification queries:

A can request a proxy signature verification on a tuple $(m, m_w, p\sigma)$, where user 1 can be an original signer or a proxy signer, and user U_v is the designated verifier. Because B knows the private key of the user U_v, B can run the algorithm PV normally and returns 1 if it is correct, or 0 otherwise.

(10) Proxy signature simulation queries:

User 1 can be the original signer or the proxy signer, and user U_v is the designated verifier. A chooses a warrant m_{w_i} and a message m_i, then asks B to produce a proxy signature simulation. It must satisfy $m_w \in Warro$ (user 1 being the original signer) or $m_w \in \{Warrp \bigcup Warrs\}$ (user 1 being the proxy signer), and m_i suits m_{w_i}. Because B knows the private key of the user U_v, B can run algorithm TS normally to produce a proxy signature $p\sigma$, returns $p\sigma$ to A, and puts (m_{w_i},m_i) to the list TS_{qu}.

At last, the adversary A stops the above queries and outputs a forgery:

E_1: A outputs a forgery $\sigma_1 = (\sigma',m',T')$, where $V_1(y_1,m',\sigma_1)=1$, and $m' \notin S_{1-qu}$, i.e., A forges a user 1's standard signature of scheme 1. Because scheme 1 is a generic digital signature scheme, based on Forking Lemma [28], B can produce two valid signatures (σ',m',T') and (σ'',m',T'), where $h_1'(m',T',y_1) \neq h_1''(m',T',y_1)$, $\sigma' = d - x_1h_1'(m',T',y_1) \bmod q$, $\sigma'' = d - x_1h_1''(m',T',y_1) \bmod q$, so we can get $x_1 = (\sigma' - \sigma'')/(h_1''(m',T',y_1) - h_1'(m',T',y_1))$, i.e., B solves the DLP.

Claim 1. According to Forking Lemma [28], if A can forge a user 1's standard signature of scheme 1, with probability $\varepsilon \geq 10(q_{s1}+1)(q_{s1}+q_{h1})/2^k$ within time t, suppose A makes at most

q_{s1} standard signature queries of scheme 1, q_{h1} queries to random oracle H_1, then DLP can be solved with probability $\varepsilon' \ge 1/9$ in time $t' \le 120686 q_{h1} t/\varepsilon$.

E_2: A outputs a forgery $\sigma_2 = (m', s_1', s_2', T')$, where $V_2(y_1, m', \sigma_2) = 1$, and $m' \notin S_{2-qu}$, i.e., A forges a user 1's standard signature of scheme 2. According to Ref. [15], the Forking Lemma [28] is also suitable, then B can get two valid signatures (m', s_1', s_2', T') and (m', s_1', s_2'', T'), where $h_2'(m', T') \ne h_2''(m', T')$, $s_2' = w - x_1 h_2'(m', T') \bmod q$, $s_2'' = w - x_1 h_2''(m', T') \bmod q$, so we can get $x_1 = (s_2' - s_2'')/(h_2''(m', T') - h_2'(m', T'))$, i.e., B solves the DLP.

Claim 2. According to Forking Lemma [28], if A can forge a user 1's standard signature of scheme 2, with probability $\varepsilon \ge 10(q_{s2}+1)(q_{s2}+q_{h2})/2^k$ within time t, suppose A makes at most q_{s2} standard signature queries of scheme 2, q_{h2} queries to random oracle H_2, then DLP can be solved with probability $\varepsilon' \ge 1/9$ in time $t' \le 120686 q_{h2} t/\varepsilon$.

E_3: A outputs a forgery $p\sigma = (m_{w_i}', m_i', s_{1i}', s_{2i}', T_i')$, where $PV(y_i, y_1, x_v, m_{w_i}', m_i', p\sigma) = 1$, $(i \in \{1,2,...,n\})$, and $(m_{w_i}', m_i') \notin (PS_{qu} \cup TS_{qu})$, i.e., A forges a proxy signature by user 1 on behalf of user $i \in \{1,2,...,n\}$, and user U_v is the designated verifier, including self-delegation. According to Ref. [15], the Forking Lemma [28] is also suitable, then B can get two valid proxy signatures $(m_{w_i}', m_i', s_{1i}', s_{2i}', T_i')$ and $(m_{w_i}', m_i', s_{1i}', s_{2i}'', T_i')$, where $h_{2i}'(m_i', T_i') \ne h_{2i}''(m_i', T_i')$, $s_{2i}' = w_i' - sk_p h_{2i}'(m_i', T_i') \bmod q$, $s_{2i}'' = w_i' - sk_p h_{2i}''(m_i', T_i') \bmod q$, so we can get $sk_p = (s_{2i}' - s_{2i}'')/(h_{2i}''(m_i', T_i') - h_{2i}'(m_i', T_i'))$, i.e., by being given the public key $pk_p = g^{sk_p} = T_i' y_i^{-h_{1i}'(m_{w_i}', T_i', y_i)} y_1^{h_{1i}'(m_{w_i}', T_i', y_1)}$ of proxy signer U_1, B obtains sk_p, so B solves the DLP, $h_{1i}'(m_{w_i}', T_i', y_i)$ and $h_{1i}'(m_{w_i}', T_i', y_1)$ can be gotten by query the H_1 oracle.

Claim 3. According to Forking Lemma [28], if A can forge a proxy signature by user 1 on behalf of user $i \in \{1,2,...,n\}$, and user U_v is the designated verifier, including self-delegation, with probability $\varepsilon \ge 10(q_{ps}+q_{ts}+1)(q_{ps}+q_{ts}+q_{h2})/2^k$ within time t, suppose A makes at most q_{ps} proxy signature queries of user 1 on behalf of user $i \in \{1,2,...,n\}$, q_{ts} transcript simulation queries of user 1 on behalf of user $i \in \{1,2,...,n\}$, q_{h2} queries to random oracle H_2, then DLP can be solved with probability $\varepsilon' \ge 1/9$ in time $t' \le 120686 q_{h2} t/\varepsilon$.

E_4: A outputs a forgery $p\sigma = (m_{w_i}', m_i', s_{1i}', s_{2i}', T_i')$, where $PV(y_1, y_i, x_v, m_{w_i}', m_i', p\sigma) = 1$, $(i \in \{2,...,n\})$, and $m_{w_i}' \notin Warro$, i.e., A forges a proxy signature by user $i \in \{2,...,n\}$ on behalf of user 1, and user U_v is the designated verifier. According to Ref. [15], the Forking Lemma [28] is also suitable, then B can get two valid proxy signatures $(m_{w_i}', m_i', s_{1i}', s_{2i}', T_i')$ and $(m_{w_i}', m_i', s_{1i}', s_{2i}'', T_i')$, where $h_{2i}'(m_i', T_i') \ne h_{2i}''(m_i', T_i')$, $s_{2i}' = w_i' - sk_p h_{2i}'(m_i', T_i') \bmod q$, $s_{2i}'' = w_i' - sk_p h_{2i}''(m_i', T_i') \bmod q$, so we can get $sk_p = (s_{2i}' - s_{2i}'')/(h_{2i}''(m_i', T_i') - h_{2i}'(m_i', T_i'))$, i.e., by being given the public key $pk_p = g^{sk_p} = T_i' y_1^{-h_{1i}'(m_{w_i}', T_i', y_1)} y_i^{h_{1i}'(m_{w_i}', T_i', y_i)}$ of proxy signer U_i, B obtains sk_p, so B solves the DLP, $h_{1i}'(m_{w_i}', T_i', y_1)$ and $h_{1i}'(m_{w_i}', T_i', y_i)$ can be gotten by query the H_1 oracle.

Claim 4. According to Forking Lemma [28], if A can forge a proxy signature by user $i \in \{2,...,n\}$ on behalf of user 1, and user U_v is the designated verifier, with probability $\varepsilon \ge 10(q_{ts}+1)(q_{ts}+q_{h2})/2^k$ within time t, suppose A makes at most q_{ts} transcript simulation queries of user $i \in \{2,\ldots,n\}$ on behalf of user 1, and user U_v is the designated verifier, q_{h2} queries to random oracle H_2, then DLP can be solved with probability $\varepsilon' \geq 1/9$ in time $t' \leq 120686 q_{h2} t/\varepsilon$.

Theorem 2 The proposed SDVPS scheme satisfies the security requirement of non-transferability.

Theorem 3 The proposed SDVPS scheme satisfies the security requirement of strong privacy of signer's identity even under the key-compromise attack.

The above two theorems can be proven similarly to Ref. [15].

5.2. Comparison

The following notations are used in the comparison.

$|x|$: the bit-length of an integer x

T_h: the time for performing a one-way hash function h

T_m: the time for performing a modular multiplication computation

T_e: the time for performing a modular exponentiation computation

T_i: the time for performing a modular inverse computation

We compare the improved scheme with several other SDVPS schemes including Wang's (Wang for short) [10], Dai et al.'s (DYD for short) [7] and the original scheme (LWH for short) [15]. Table 1 shows the comparison. Note that the time complexities for performing the modular exponentiation, multiplication, and inverse operations over $GF(p)$ are $O(\log^3(p))$, $O(\log^2(p))$, and $O(\log^2(p))$, respectively [29], and a hash computation does not take longer time than that of a modular multiplication [30]. From table 1, we can see in PCG stage, the computational costs of the improved scheme are the same as the original scheme, and are less than the DYD scheme's and Wang's scheme's. In PSG stage, the improved scheme has one more modular multiplication and one more hash computation than the original scheme. In PSV stage, the improved scheme has one more modular exponentiation and one more hash computation than the original scheme. Totally, the improved scheme needs one less modular exponentiation than DYD scheme and Wang's scheme. So, in computational costs, the original scheme is the most efficient one, and the improved scheme is more efficient than DYD scheme and Wang's scheme. The ciphertext size of the improved scheme is the same as the original scheme's, and is shorter than the DYD scheme's and Wang's scheme's. To sum up, the improved scheme is not so efficient as the original scheme, but is more efficient than DYD scheme and Wang's scheme. It still remains a high efficient scheme and can be used in electronic commerce.

Table 1. Comparison with other SDVPS schemes.

	PCG	PSG	PSV	Total	Ciphertext Size
DYD	$3T_e + 2T_m + 2T_h$	$2T_e + T_m + T_h$	$4T_e + 4T_m + 2T_h$	$9T_e + 7T_m + 5T_h$	$3\|p\| + \|q\|$
Wang	$4T_e + 4T_m + 2T_h$	$T_e + 2T_m + T_h + T_i$	$4T_e + 4T_m + 2T_h$	$9T_e + 10T_m + 5T_h + T_i$	$\|p\| + 3\|q\|$
LWH	$3T_e + 2T_m + 2T_h$	$T_e + T_m + 2T_h$	$3T_e + 4T_m + 3T_h$	$7T_e + 7T_m + 7T_h$	$3\|q\|$
Ours	$3T_e + 2T_m + 2T_h$	$T_e + T_m + 2T_h$	$4T_e + 4T_m + 4T_h$	$8T_e + 7T_m + 8T_h$	$3\|q\|$

6. Conclusion

Designated verifier proxy signature can combine the merits of proxy signature and designated verifier signature. Recently, Lin et al. proposed a SDVPS scheme, which has not only shorter signature length, but also lower computational costs. In this paper, we point out their scheme is forgeable by giving four attacks, and further we propose an improvement of their scheme. We also give out a new formal definition and a new security model of existential unforgery of SDVPS scheme, and prove our improved scheme in the new security model, and there is no need for a secure channel between an original signer and a proxy signer when producing the proxy credential. At last, we give out performance analysis, which shows the improved scheme is efficient and suitable for electronic commerce. The future work is to design lattice-based [31] proxy signature or proxy signature in multivariate public key cryptography.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant No. 61462048) and the key program of Jiujiang University under Grant No. 2013ZD02.

This is the full version of an extended abstract presented at ISA 2014.

References

[1] M. Mambo, K. Usuda and E. Okamoto, “Proxy signatures for delegating signing operation,” In Proceedings of the 3rd ACM Conference on Computer and Communication Security, **(1996),** pp. 48–57.

[2] M. A. Jabri and S. Matsuoka, “Dealing with grid-computing authorization using identity-based certificateless proxy signature,” In 2011 11th IEEE/ACM International Symposium on CC Grid, **(2011),** pp. 544-553.

[3] X. Q. Cai and S. Y. Wang, “Cryptanalysis of efficient threshold proxy signature protocol for mobile agents,” In 2011 IEEE International Conference on CSAE, vol. 1, **(2011),** pp. 726-728.

[4] S. Kim, S. Park and D. H. Won, “Proxy signatures, revisited,” In Proc. of ICICS’1997, **(1997),** pp. 223-232.

[5] M. Jakobsson, K. Sako and R. Impagliazzo, “Designated verifier proofs and their applications,” EUROCRYPT’1996, LNCS 1070, **(1996),** pp. 143-154.

[6] S. Saeednia, S. Kremer and O. Markowitch, “An efficient strong designated verifier signature scheme,” in Proceedings of the 6th International Conference on Information Security and Cryptology, **(2003),** pp. 40-54.

[7] J. Z. Dai, X. H. Yang and J. X. Dong, “Designated receiver proxy signature scheme for e-commerce”, In Proc. of IEEE International Conference on System, Man and Cybernetics, vol. 1, **(2003),** pp. 384-389.

[8] G. L. Wang, “Designated-verifier proxy signatures for e-commerce”, In the IEEE 2004 International Conference on Multimedia and Expo, **(2004),** pp. 1731-1734.

[9] X. X. Li, K. F. Chen and S. Li, “Designated verifier proxy signatures for commerce from bilinear pairings”, In Proc. of 16th Int. Conf. on Computer Communication, **(2004),** pp. 1249-1252.

[10] G. L. Wang, “Designated-verifier proxy signature schemes,” Security and Privacy in the Age of Ubiquitous Computing, vol. 181, **(2005),** pp. 409-423.

[11] X. Y. Huang, Y. Mu, W. Susilo and F. T. Zhang, “Short designated verifier proxy signature from pairings,” The First International Workshop on Security in Ubiquitous Computing Systems, LNCS 3823, **(2005),** pp. 835-844.

[12] R. X. Lu and Z. F. Cao, “Designated verifier proxy scheme with message recovery,” Applied Mathematics and Computation, vol. 169, **(2005),** pp. 1237-1246.

[13] T. Cao, D. Lin and R. Xue, “ID-based designated-verifier proxy signatures,” IEE Proc. Commun., vol. 152, **(2005),** pp. 989-994.

[14] K. A. Shim, “Short designated verifier proxy signature,” Computers and Electrical Engineering, vol. 37, **(2011),** pp. 180-186.

[15] H. Y. Lin, T. S. Wu and S. K. Huang, “An efficient strong designated verifier proxy signature scheme for electronic commerce”, Journal of Information Science and Engineering, vol. 28, **(2012),** pp. 771-785.

[16] C. L. Hsu and H. Y. Lin, “Pairing-based strong designated verifier proxy signature scheme with low cost,” Security and Communication Networks, vol. 5, **(2012),** pp. 517-522.

[17] A. Boldyreva, A. Palacio and B. Warinschi, “Secure proxy signature schemes for delegation of signing rights,” eprint.iacr.org, **(2003)**.

[18] J. Herranz and G. Saez, “Revisiting fully distributed proxy signature schemes,” eprint.iacr.org, **(2003)**.

[19] T. Malkin, S. Obana and M. Yung, "The hierarchy of key evolving signatures and a characterization of proxy signatures," Eurocrypt 2004, LNCS 3027, **(2004),** pp. 306-322.
[20] Q. Wang and Z. F. Cao, "Security Arguments for Partial Delegation with Warrant Proxy Signature Schemes," eprint.iacr.org, **(2004)**.
[21] J. C. N. Schuldt, K. Matsuura and K. G. Paterson, "Proxy signatures secure against proxy key exposure," PKC 2008, LNCS 4939, **(2008),** pp. 141-161.
[22] W. Wu, Y. Mu, W. Susilo, J. Seberry and X. Y. Huang, "Identity-based proxy signature from pairings," Proc. of ATC 2007, LNCS, vol. 4610, **(2007),** pp. 22-31.
[23] L. Zhang, F. T. Zhang and Q. H. Wu, "Delegation of signing rights using certificateless proxy signatures," Information Science, vol. 184, **(2012),** pp. 298-309.
[24] N. Y. Lee, T. Hwang and C. H. Wang, "On zhang's nonrepudiable proxy signature schemes," In Proc. of ACISP 1998, LNCS 1438, **(1998),** pp. 415-422.
[25] K. Zhang, "Threshold proxy signature schemes," In Proc. of Information Security Workshop, **(1997),** pp. 191-197.
[26] F. G. Li, M. Shirase and T. Takagi, "Cryptanalysis of efficient proxy signature schemes for mobile communication," Science China: Information Science, vol. 53, **(2010),** pp. 2016-2021.
[27] R. X. Lu, X. L. Dong and Z. F. Cao, "Designing efficient proxy signature schemes for mobile communication," Science China: Information Science, vol. 51, **(2008),** pp. 183-195.
[28] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," Journal of Cryptology, vol. 13, **(2000),** pp. 361-396.
[29] J. Camenisch, "Group signature schemes and payment systems based on the discrete logarithm problem," in: ETH Series in Information Security and Cryptography, vol. 2, **(1998),** pp. 11-12.
[30] C. J. Mitchell, F. Piper and P. Wild, "Digital signature," In: Contemporary Cryptology: The Science of Information Integrity. IEEE Press, New York, **(1992)**.
[31] C. S. Gu and J. X. Gu, "Cryptanalysis of smart-vercauteren and Gentry-Halevi's fully homomorphic encryption", International Journal of Security and Its Applications, vol. 6, **(2012),** pp. 103-108.

Authors

Caixue Zhou, received his B.A. degree in Computer Science Department from Fudan University, Shanghai, China in 1988, and his M.S. degree in Space College of Beijing University of Aeronautics and Astronautics, Beijing, China in 1991. He is an Association Professor in the School of Information Science and Technology, University of Jiujiang, China since 2007. He is a member of the CCF (China Computer Federation). His research interests include applied cryptography and network security.