ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
https://doi.org/10.17559/TV-20180910170418
Original scientific paper

Efficient and Provably-secure Certificateless Strong Designated Verifier Signature Scheme without Pairings

Meijiao DUAN, Jianming ZHU, Yang LI

Abstract: Strong designated verifier signature (generally abbreviated to SDVS) allows signers to obtain absolute control over who can verify the signature, while only the designated verifier other than anyone else can verify the validity of a SDVS without being able to transfer the conviction. Certificateless PKC has unique advantages comparing with certificate-based cryptosystems and identity-based PKC, without suffering from key escrow. Motivated by these attractive features, we propose a novel efficient CL-SDVS scheme without bilinear pairings or map-to-point hash operations. The proposed scheme achieves all the required security properties including EUF-CMA, non-transferability, strongness and non-delegatability. We also estimate the computational and communication efficiency. The comparison shows that our scheme outperforms all the previous CL-(S)DVS schemes. Furthermore, the crucial security properties of the CL-SDVS scheme are formally proved based on the intractability of SCDH and ECDL assumptions in random oracle model.

Keywords: certificateless cryptography; existential unforgeability; non-delegatability; strong designated verifier signature; without pairings

1 INTRODUCTION

Traditional digital signature scheme with the public-verifiable property does not fit the requirement of various scenarios, such as deniable authentication, e-bidding, e-voting. To settle this problem, Jakobsson [1] put forward a new concept called designated verifier signature (generally abbreviated to DVS). The most attractive property of DVS is called *non-transferability*, which ensures signers to obtain absolute control over who can verify the signature. In addition, the designated verifier cannot convince any third entity of the fact the DVS is indeed signed by the genuine signer. Meanwhile, a variant of DVS called Strong designated verifier signature (SDVS) is also proposed. None other than the designated verifier can verify the SDVS. We call this property the *strongness* property, which is usually achieved by forcing the verification process using the designated verifier's secret. The formalized definition was first formed by Saeednia [2] and extended by Laguillaumie [3]. Later on, Lipmaa et al. [4] pointed out that most previous (S)DVS schemes [2, 5-6] suffer from a new-type attack named delegatability, in which the signer could delegate his/her signing ability to any third entity by transferring a common value without revealing the secret key. To capture this issue, the security notion of *non-delegatability* was proposed and formalized. Due to these unique properties above, the SDVS can help resolve the conflict between authentication and privacy protection in digital signatures, thus fit into various cryptographic applications and some new fields, such as IoT communications [7], biometric authentication and identity-management [8], privacy preserving cloud computing, the next-generation network infrastructure [9], and so on.

Following Saeednia et al.'s work, many identity-based (S)DVS schemes were proposed [10-13]. And recently Shooshtari et al. [14] presented a code-based SDVS scheme against quantum attacks. Unfortunately, all the above identity-based (S)DVS schemes suffer from the inherent problem of key escrow in ID-based cryptosystems. Certificateless public key cryptography, which was originally introduced by Al-Riyami et al. [15], can address this problem. It relies on a semi-trusted KGC who owns the master secret key. The full secret key of arbitrary user was deduced from two parts: partial private key D_i supplied from KGC, and secret value x_i chosen by himself. Thus, it has unique advantages comparing with certificate-based cryptosystems and identity-based PKC, without suffering from the problem of key escrow.

Motivated by this attractive feature, Huang [16] presented the first construction of certificateless DVS scheme. However, their scheme is figured out to be vulnerable to malicious-but-passive KGC attacks. Later on, some efficient short certificateless DVS schemes were independently designed [17-19] and claimed to be secure. However, most of them [17-18] still need several pairing operations and the previous one does not resist the public key replace attack. Recently, several certificateless strong DVS schemes (CL-SDVS for short) have been proposed [20-24]. Unfortunately, some of them cannot achieve all the security properties as they claimed. According to [25], the scheme in Reference [22] is insecure against the malicious PKC attack and public key replacement attack. The CL-SDVS scheme proposed in Reference [24] is pointed vulnerable to concrete attack which shows the signature is forgeable [26]. Recalling the proofs of previous insecure schemes [16-18, 22], we find that there is an inappropriate restriction in their security models of unforgeability. The adversary cannot query target signer or the target verifier to sign arbitrary message, nor to check the validity of any signature. With these restrictions, they actually ignore such a practical attack in which the adversary can query arbitrary signature on arbitrary message with respect to arbitrary signer and verifier, or the adversary can eavesdrop all communications between signer and verifier. Thus, the previous models are not adequately considered and essentially limit the adversary's abilities. Therefore, we consider a strictly stronger security model in Section 4. Moreover, expensive operations such as bilinear pairing, pairing-based exponentiation, and map-to-hash operations are required in all existing schemes. And the computation cost is still maintained at a high level.

This paper presents a novel efficient certificateless SDVS scheme with shorter signature size, achieving all the required security properties of strongness, non-transferability, non-delegatability and EUF-CMA against both Type I and Type II forgers. The unforgeability for our scheme is formally proved based on Square Computational Diffie-Hellman (SCDH) assumption and Elliptic Curve Discrete Logarithm (ECDL) assumption in random oracle model, of which the SCDH assumption is equivalent to the CDH assumption [27].And the proof of non-delegatability is strictly following the formalized definition originally introduced by Lipmaa [4]. Furthermore, our scheme does not need expensive pairing operations, which are making it more efficient than all the existing CL-SDVS schemes proposed so far. Due to its security and efficiency, our CL-SDVS scheme can be deployed in various power-constrained applications, such as sensor networks, IoT and cyber-physical systems.

The rest of this paper is organized as follows. The next section introduces the formal definition of certificateless SDVS scheme. We also describe the formalized model of crucial security properties, including unforgeability and non-delegatability. In Section 3, we propose a novel efficient certificateless SDVS scheme. The performance and security are analyzed in the next two sections. The last section concludes this paper.

2 MODELS OF CL-SDVS SCHEMES

2.1 Definition of CL-SDVS

A certificateless strong designated verifier signature (CL-SDVS) scheme consists of eight algorithms shown as the following:

- ***Setup***: This PPT algorithm runs by the KGC. Taking the security parameter κ as input, it outputs the public parameters *params* and the master secret key *msk*.
- ***Partial-Private-Key-Extract***: This PPT algorithm runs by KGC. Taking *params*, *msk* and $ID \in \{0,1\}^*$ (which is the identity of a user) as input, it returns the corresponding partial private key d_{ID}.
- ***Set-Secret-Value***: This PPT algorithm runs by a user with identity *ID*. Taking *params* and *ID as input*, this algorithm randomly selects a secret value x_{ID} and returns to the user.
- ***Set-Public-Key***: This PPT algorithm runs by a user with identity *ID*. On input x_{ID}, the algorithm calculates the public key PK_{ID}.
- ***Set-Private-Key***: This PPT algorithm executes by a user with identity *ID*. On input d_{ID}, PK_{ID} and x_{ID}, this algorithm produces full private key SK_{ID}.
- ***Sign***: This PPT algorithm runs by a signer. It takes (ID_S, SK_{ID_S}), (ID_V, PK_{ID_V}) and $m \in \mathcal{M}$ as input, and returns $\sigma = Sign\left(SK_{ID_S}, ID_S, ID_V, PK_{ID_V}, m\right)$.
- ***Verify***: This deterministic algorithm runs by the designated verifier. Taking ID_V, SK_{ID_V}, ID_S, PK_{ID_S}, m and the purported signature σ as input, it returns 1 (*accept*)or 0 (*reject*).
- ***Simulation***: This algorithm runs by the designated verifier. Taking ID_V, SK_{ID_V}, ID_S, PK_{ID_S} and m as input, it returns a simulated signature σ' on m, *i.e.* $\sigma' = Simulation\ \left(SK_{ID_V}, ID_V, ID_S, PK_{ID_S}, m\right)$, which should be indistinguishable from signatures constructed by the real signer.

2.2 Security Model of CL-SDVS Schemes

A secure certificateless SDVS scheme must achieve four crucial security properties: unforgeability, strongness, non-transferability, and non-delegatability.

Unforgeability. This property ensures that no one can forge a valid CL-SDVS with non-negligible probability without obtaining the private key of either the signer or the designated verifier. Depending on the different capabilities, we can divide the forger in certificateless SDVS schemes into two types. The description and formalization of these forgers is given below.

Type I Forger: Let $\mathcal{F}_I$ denote this type of forger. Forger $\mathcal{F}_I$ can request public keys, and even substitute public keys for some values he selects, but cannot obtain the master key *msk*. Forger $\mathcal{F}_I$ simulates a normal third-party attacker against the CL-SDVS scheme. To formalize the attack model, we demonstrate the following game performed between the challenger $\mathcal{C}$ and the forger $\mathcal{F}_I$.

- **Setup:** $\mathcal{C}$ runs the Setup algorithm to generate *msk* and *params*. Then the forger $\mathcal{F}_I$ can obtain *params*, the identity and public keys of the singer *S* and the specified verifier *V*.
- **Queries**: The forger $\mathcal{F}_I$ can get access to the following oracles.
    - ExtrPartSK(*ID*): On input of any user's identity *ID*, $\mathcal{C}$ produces the corresponding partial private key d_{ID}.
    - ExtrFullSK(*ID*): On input *ID*, $\mathcal{C}$ returns the corresponding full private key SK_{ID}.
    - ReqestPK(*ID*): On input *ID*, $\mathcal{C}$ returns the corresponding public key PK_{ID}.
    - ReplacePK(*ID*): The forger $\mathcal{F}_I$ can select a substitutional public key PK_{ID}' and replace the original public key PK_{ID}.
    - SIGN(*m*, ID_S, ID_V): On input *m* with ID_S and ID_V, it responds a valid signature σ on *m*.
    - VER(σ, *m*, ID_S, ID_V): On input of a signature σ on *m* with ID_S and ID_V, it executes the verify algorithm and returns True (*valid*) or False (*invalid*).
    - SIMUL(*m*, ID_S, ID_V): On input of a message *m* with designated verifier's identity ID_V and signer's identity ID_S, it responds a valid transcript simulation σ' on *m*.
- **Output:** Eventually, $\mathcal{F}_I$ produces a forgery $\left(m^*, \sigma^*\right)$ with the target signer ID_S^* and designated verifier ID_V^*. The forger $\mathcal{F}_I$ wins this game if:
    1. Algorithm ***Verify*** (σ^* , m^* , ID_S^* , ID_V^*) outputs *accept*;
    2. $\mathcal{F}_I$ has never queried ExtrPartSK (ID_V^*) or ExtrFullSK (ID_V^*) ;

3. $\mathcal{F}_I$ has never queried ExtrPartSK (ID_S^*) or ExtrFullSK(ID_S^*);
4. $\mathcal{F}_I$ has never queried SIGN(m, ID_S^*, ID_V^*) or SIMUL(m, ID_S^*, ID_V^*) to get the signature or a simulated one on m^* with ID_S^* and ID_V^*;
5. $\mathcal{F}_I$ has never queried SIGN(m, ID_V^*, ID_S^*) or SIMUL(m, ID_V^*, ID_S^*) to get the signature or a simulated one on m^* with ID_V^* and ID_S^*.

Type II Forger: Let $\mathcal{F}_{II}$ denote this type of forger. Forger $\mathcal{F}_{II}$ has access to the master-key *msk*, but could not substitute public keys. Forger $\mathcal{F}_{II}$ simulates a malicious KGC against the CL-SDVS scheme. To define the attack model, we demonstrate this game below performed between a challenger $\mathcal{C}$ and the forger $\mathcal{F}_{II}$.

- **Setup:** $\mathcal{C}$ executes the Setup algorithm of the CL-SDVS scheme to generate *msk* and *params*, which the forger $\mathcal{F}_{II}$ also can both obtain.
- **Queries:** The forger $\mathcal{F}_{II}$ can get access to the following oracles, which are the same as in Game I: ExtrFullSK(*ID*), ReqestPK(*ID*), SIGN(*m*, *ID_S*, *ID_V*), VER(σ, *m*, ID_S, ID_V), and SIMUL(*m*, ID_S, ID_V).
- **Output:** Eventually, $\mathcal{F}_{II}$ outputs his forgery $\left(m^*, \sigma^*\right)$ with the target ID_S^* and ID_V^*. The forger $\mathcal{F}_{II}$ wins this game if:
  1. Algorithm ***Verify*** (σ^*, m^*, ID_S^*, ID_V^*) outputs *accept*;
  2. $\mathcal{F}_{II}$ has never queried ExtrFullSK (ID_S^*) or ExtrFullSK(ID_V^*);
  3. $\mathcal{F}_{II}$ has never queried SIGN (m, ID_S^*, ID_V^*) or SIMUL (m, ID_S^*, ID_V^*) to get the signature or a simulated one on m^* with ID_S^* and ID_V^*;
  4. $\mathcal{F}_{II}$ has never queried SIGN(m, ID_V^*, ID_S^*) or SIMUL(m, ID_V^*, ID_S^*) to get the signature or a simulated one on m^* with ID_V^* and ID_S^*.

Strongness. The verification process needs to use the specified verifier's secret, without which any third entity cannot confirm whether a signature is valid.

Non-transferability. Given arbitrary message m and the corresponding signature σ generated by the signer, it is computationally indistinguishable from the simulation σ' produced by the specified verifier.

Non-delegatability. The signer cannot delegate his/her signing ability to any third entity by transferring a common value without revealing the secret key.

Assume that $\mathcal{F}$ denotes the delegated entity by the real signer, who can generate a valid signature in time τ with a non-negligible probability ε. Then we can construct an efficient black-box knowledge extractor E based on the algorithm $\mathcal{F}$. Suppose $\mathcal{F}$ can produce a valid signature on m for (PK_{ID_S}, SK_{ID_S}) and (PK_{ID_V}, SK_{ID_V}), where m is an arbitrary input message, (PK_{ID_S}, SK_{ID_S}) is the key pair of an arbitrary signer, (PK_{ID_V}, SK_{ID_V}) is the key pair of an arbitrary designated verifier. And the probability of $\mathcal{F}$ succeed is $\varepsilon > r$, which is related to the choice of the random coin toss and the hash function H. Let $\mathcal{F}_m$ represent $\mathcal{F}$ with m as its input. Then by taking $\mathcal{F}_m$ as a subroutine, the knowledge extractor E can deduce either SK_{ID_S} or SK_{ID_V} in expected time $\tau / (\varepsilon - r)$, where $r \in [0,1]$ is the knowledge error. The time cost making the oracle queries is negligible here. If such an E exists, the CL-SDVS scheme is (τ, r)-non-delegatable.

3 PROPOSED CL-SDVS SCHEME

This section presents an efficient Certificateless SDVS scheme without using any expensive pairing computation or map-to-point hash operation. Our scheme satisfies all the required security properties including unforgeability, non-transferability, strongness and non-delegatability. The signature length is also obviously reduced. The construction of our CL-SDVS scheme is as follows:

- **Setup.** Let $\mathbb{F}_q$ be a finite field and $\mathbb{E} / \mathbb{F}_p$ be an elliptic curve $y^2 = x^3 + ax + b$ over $\mathbb{F}_q$, where p and q are two large primes. Let $\mathbb{G}$ be a q-order subgroup of the additive group of points over $\mathbb{E} / \mathbb{F}_p$, P be a generator of $\mathbb{G}$, and $H_1(\cdot), H_2(\cdot): \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ be hash functions. The trusted KGC selects a random value $s \in \mathbb{Z}_q^*$ and calculates $P_{pub} = sP$. Finally, the public parameters $params = (p, q, \mathbb{E} / \mathbb{F}_p, \mathbb{G}, P, P_{pub}, H_1, H_2)$ is released, and master key $msk = s$ is kept secret by KGC.
- **Partial-Private-Key-Extract.** Given a user identity *ID*, KGC computes its partial private key as follows. First it chooses a random value $y \in \mathbb{Z}_q^*$ and sets $Y_{ID} = yP$. Then it calculates the partial private key $d_{ID} = y + H_1(P_{pub}, ID, Y_{ID})s$. KGC sends d_{ID} and Y_{ID} to the corresponding user via a secure channel. The correctness of the partial private key can be confirmed by verifying $Y_{ID} + H_1(P_{pub}, ID, Y_{ID}) P_{pub} = d_{ID}P$.
- **Set-Secret-Value.** The user with identity *ID* sets his secret value by randomly choosing $x_{ID} \in \mathbb{Z}_q^*$.
- **Set-Private-Key.** Combining d_{ID} and x_{ID}, the user with identity *ID* sets $sk_{ID} = d_{ID} + x_{ID} = w_{ID}$ as his full private key.
- **Set-Public-Key.** On input *params* and x_{ID}, the user with identity *ID* calculates $X_{ID} = x_{ID}P$, and sets $PK_{ID} = (X_{ID}, Y_{ID})$ as his public key.
- **Sign. For** arbitrary message m and the designated verifier V, the signer S calculates the signature σ using

its private key $SK_{ID_S} = w_S$ as follows. S gets V's identity ID_V and public key $PK_{ID_V} = (X_V, Y_V)$, and computes $H_V = H_1(P_{pub}, ID_V, Y_V)P_{pub}$. Then S selects three random values $t, r, k \in \mathbb{Z}_q^*$, calculates $T = tP$, $R = rP$ and

$$c_1 = (t + w_S)(X_V + Y_V + H_V)$$
$$c_2 = R + k(X_V + Y_V + H_V)$$
$$h = H_2(ID_S, ID_V, m, T, c_1, c_2)$$
$$z = t + (r + h)w_S \bmod q$$

The CL-SDVS is then $\sigma = (r, k, h, z)$.

There is a remain issue that should be noticed. Before computing z in the above process, it is necessary to make sure $r + h \neq 0 \bmod q$. If this inequation does not hold, S should re-select an alternative random $r' \in \mathbb{Z}_q^*$ which satisfies $r' + h \neq 0 \bmod q$. Then the signature can be produced subsequently.

- **Verify.** Given a CL-SDVS signature $\sigma = (r, k, h, z)$ on message m, the designated verifier V needs to check whether $r + h \neq 0 \bmod q$ holds in advance. If not, V outputs *"reject"*. Otherwise, it calculates

$$H_S = H_1(P_{pub}, ID_V, Y_S)P_{pub}$$
$$T = zP - (r + h)(X_S + Y_S + H_S)$$
$$c_1 = w_V(T + X_S + Y_S + H_S)$$
$$c_2 = rP + kw_V P$$

If and only if

$$h = H_2(ID_S, ID_V, m, T, c_1, c_2)$$

holds, V outputs *accept*. Otherwise, V outputs *"reject"*.

- **Simulation.** The specified verifier V with private key $SK_V = (d_V, x_V)$ can calculate a simulated signature σ', satisfying that it is indistinguishable from σ. V randomly selects three values $z', \beta, \alpha \in \mathbb{Z}_q^*$, and calculates

$$H_S = H_1(P_{pub}, ID_V, Y_S)P_{pub}$$
$$T' = z'P - \beta(X_S + Y_S + H_S)$$
$$c_1' = w_V(T' + X_S + Y_S + H_S)$$
$$c_2' = \alpha P$$
$$h' = H_2(ID_S, ID_V, m, T', c_1', c_2')$$
$$r' = (\beta - h') \bmod q$$
$$k' = w_V^{-1}(\alpha - r') \bmod q$$

Then the simulated CL-SDVS is $\sigma' = (r', k', h', z')$.

The correctness of the CL-SDVS scheme is proved following from the fact that

$$\begin{aligned}T &= zP - (r + h)(X_S + Y_S + H_S)\\ &= [t + (r + h)w_S]P - (r + h)(X_S + Y_S + H_S)\\ &= tP + (r + h)w_S P - (r + h)(X_S + d_S P)\\ &= tP + (r + h)w_S P - (r + h)(x_S + d_S)P\\ &= tP\\ c_1 &= w_V(T + X_S + Y_S + H_S)\\ &= w_V(T + X_S + Y_S + H_1(P_{pub}, ID_S, Y_S)P_{pub})\\ &= w_V(t + x_S + d_S)P = (t + w_S)w_V P\\ &= (t + w_S)(X_V + Y_V + H_1(P_{pub}, ID_V, Y_V)P_{pub})\\ &= (t + w_S)(X_V + Y_V + H_V)\end{aligned}$$

and

$$\begin{aligned}c_2 &= rP + kw_V P\\ &= R + k(X_V + Y_V + H_1(P_{pub}, ID_V, Y_V)P_{pub})\\ &= R + k(X_V + Y_V + H_V).\end{aligned}$$

4 SECURITY ANALYSIS

The computational assumptions on which the security proof relies are briefly described below.

Definition 1 (Square Computational Diffie-Hellman (SCDH) Problem): Given a q-order cyclic additive group $\mathbb{G}$ generated by an element P. On input P and aP, computing a^2P.

Definition 2 (SCDH Assumption): Let $\mathcal{A}$ be a SCDH-adversary who can succeed in solving the SCDH problem within time t with the probability of $Adv_{\mathbb{G}}^{scdh}(\mathcal{A})$. If $Adv_{\mathbb{G}}^{scdh}(t) = \max_{\mathcal{A}}\{Adv_{\mathbb{G}}^{scdh}(\mathcal{A})\}$ is negligible, the SCDH assumption holds in $\mathbb{G}$.

Definition 3 (Elliptic Curve Discrete Logarithm (ECDL) Problem): Let $\mathbb{F}_p$ be a finite field, $\mathbb{E}$ be an elliptic curve defined over $\mathbb{F}_p$. On input P and $Q=aP$, where both of them are elements on q-order group $\mathbb{E}/\mathbb{F}_p$, compute a.

Definition 4 (ECDL assumption): For arbitrary polynomial time bounded adversary $\mathcal{A}$, the advantage of $\mathcal{A}$'s success in breaking the ECDLP problem is $Adv_{\mathbb{E}}^{ecdlp}(t) = \max_{\mathcal{A}}\{Adv_{\mathbb{E}}^{ecdlp}(\mathcal{A})\}$. If $Adv_{\mathbb{E}}^{ecdlp}(t)$ is negligible, the ECDLP assumption holds in $\mathbb{E}$.

Theorem 1. If the SCDH assumption holds, then our proposed CL-SDVS scheme is existential unforgeable against *Type I forger* under chosen message attacks in the random oracle model.

Proof. Given a SCDH instance $(A, B) = (aP, P)$, in which $a \in \mathbb{Z}_q^*$ is an unknown random value and $P \in \mathbb{G}$. Suppose $\mathcal{F}_I$ be a successful forger against our CL-SDVS scheme. Then an algorithm $\mathcal{C}$ can be constructed by running $\mathcal{F}_I$ as a subroutine to break the SCDH problem. The aim of $\mathcal{C}$ is to compute a^2P.

Algorithm $\mathcal{C}$ simulates the attack environment. And system parameters are published to $\mathcal{F}_I$, in which the master public key is $P_{pub} = A = aP$. In this formula, a is the master private key and should be kept secret to $\mathcal{C}$. The oracle queries of forger $\mathcal{F}_I$ are simulated as follows:

- $H_1(P_{pub}, ID_i, Y_i)$ query: $\mathcal{C}$ maintains a list H_1^{list} (initally empty) with entries of format $<P_{pub}, ID_i, Y_i, e_i>$. When $\mathcal{F}_I$ queries the H_1 Oracle with input (ID_i, Y_i), $\mathcal{C}$ responses as below.

- If (ID_i, Y_i) already consists in list H_1^{list} within a tuple <P_{pub}, ID_i, Y_i, e_i>, then $\mathcal{C}$ responds with $H_1(P_{pub}, ID_i, Y_i) = e_i$.
- Otherwise, $\mathcal{C}$ selects a random value $e \in \mathbb{Z}_q^*$, inserts <P_{pub}, ID_i, Y_i, e> into the list H_1^{list} and returns e.

• H$_2$ (ID_i, ID_j, m, T, c_1, c_2) query: $\mathcal{C}$ maintains a list H_2^{list} (initially empty) with entries of format <ID_i, ID_j, m, T, c_{1i}, c_{2i}, h_i> and proceeds as below.
- If a tuple indexed by (ID_i, ID_j, m, T, c_{1i}, c_{2i}) already appears on the list H_2^{list}, then $\mathcal{C}$ returns h_i.
- Otherwise, $\mathcal{C}$ randomly selects $h \in \mathbb{Z}_q^*$, inserts <ID_i, ID_j, m, T, c_{1i}, c_{2i}, h> into list H_2^{list} and responds with h.

• ExtrPartSK(ID_i) query: $\mathcal{C}$ maintains list $\mathcal{PSK}^{list}$ with entries of format <ID_i, Y_i, d_i, e_i> and responses to the queries.
- If $ID_i = ID_S^*$ or $ID_i = ID_V^*$, then abort the game.
- Otherwise, $\mathcal{C}$ randomly selects $d, e \in \mathbb{Z}_q^*$, sets $Y = dP - eP_{pub}$ and inserts tuple <ID_i,Y,d, e> into the list $\mathcal{PSK}^{list}$, $< ID_i, \cdot, Y >$ into the list $\mathcal{PK}^{list}$, and <P_{pub}, ID_i, Y, e> into the list H_1^{list}. Then $\mathcal{C}$ returns $PSK_{ID_i} = d$.

• ExtrFullSK(ID_i) query: $\mathcal{C}$ maintains list $\mathcal{SK}^{list}$ with entries of format <ID_i, d_i, x_i, w_i> and proceeds as follows to respond to the queries.
- If $ID_i = ID_S^*$ or $ID_i = ID_V^*$, then abort the game.
- Otherwise, $\mathcal{C}$ chooses a random $x \in \mathbb{Z}_q^*$, sets $X_i = xP$, $w = d_i + x$ and inserts tuple <ID_i, d_i, x, w> into the list $\mathcal{SK}^{list}$. Then $\mathcal{C}$ searches list $\mathcal{PK}^{list}$ for tuple $< ID_i, \cdot, Y_i >$. If such tuple exists, it refreshes this record with <ID_i, X_i, Y_i>.
- Otherwise, it inserts <ID_i, X_i, Y_i> in to the list $\mathcal{PK}^{list}$. Then $\mathcal{C}$ returns $SK_{ID_i} = w$.

• ReqestPK(ID_i) query: $\mathcal{C}$ maintains list $\mathcal{PK}^{list}$ with entries of format <ID_i, X_i, Y_i>. When $\mathcal{F}_I$ queries with ID_i, $\mathcal{C}$ proceeds as follows.
- If ID_i already consists in the list $\mathcal{PK}^{list}$ inside a tuple <ID_i, X_i, Y_i>, then $\mathcal{C}$ returns $PK_{ID_i} = (X_i, Y_i)$.
- Else if $ID_i = ID_S^*$, $\mathcal{C}$ randomly chooses $x, y \in \mathbb{Z}_q^*$ and sets $X = xP$, $Y = yP$, stores <ID_i, X, Y> in list $\mathcal{PK}^{list}$, and returns $PK_{ID_i} = (X, Y)$.
- Else if $ID_i = ID_V^*$, $\mathcal{C}$ generates $X = xP$, $Y = dP - eP_{pub}$ by randomly picking $x, d, e \in \mathbb{Z}_q^*$. Then it stores <$ID_i$, X, Y> in list $\mathcal{PK}^{list}$, inserts <P_{pub}, ID_i, Y, e> into the list H_1^{list} and returns in output $PK_{ID_i} = (X, Y)$.
- Otherwise, $\mathcal{C}$ makes an ExtrFullSK(ID_i) query on ID_i itself to refresh its list $\mathcal{PK}^{list}$ and returns in output $PK_{ID_i} = (X_i, Y_i)$.

• ReplacePK(ID_i, X_i', Y_i') query: When $\mathcal{F}_I$ inputs (ID_i, X_i', Y_i') to replace user ID_i 's public key, $\mathcal{C}$ searches the list $\mathcal{PK}^{list}$ to check whether an element $< ID_i, \cdot, \cdot >$ exists.
- If it does, $\mathcal{C}$ sets X_i= X_i', Y_i= Y_i' and refreshes the record with <ID_i, X_i', Y_i'>.
- Otherwise, $\mathcal{C}$ inserts tuple <ID_i, X_i', Y_i'> into the list $\mathcal{PK}^{list}$.

• SIGN(ID_i, ID_j, m) query: When $\mathcal{F}_I$ queries with signer ID_i, verifier ID_j, and message m, $\mathcal{C}$ first searches list $\mathcal{PK}^{list}$ for tuples <ID_i, X_i, Y_i> and <ID_j, X_j, Y_j>. If such tuples do not exist, it runs the above algorithms to generate keys for user ID_i and/or ID_j. Then it proceeds as follows.
- If $(ID_i, ID_j) = (ID_S^*, ID_V^*)$ or (ID_V^*, ID_S^*), and $m = m^*$, then it returns "$\perp$".
- Else if ID_i, $ID_j \neq ID_S^*$ or ID_V^*, $\mathcal{C}$ first finds <ID_i, w_i> in list $\mathcal{SK}^{list}$ and <P_{pub}, ID_j, Y_j, e_j> in list H_1^{list}. Then it selects $t, r, k \in \mathbb{Z}_q^*$, calculates T=tP, $c_1 = (t + w_i)(X_j + Y_j + e_j P_{pub})$, $c_2 = rP + kw_j P$ and obtains h by making a H$_2$(ID_i, ID_j, m, T, c_1, c_2) query. Finally $\mathcal{C}$ caculates $z = t + (h + r)w_i$ mod q, and responds with $\sigma = (r, k, h, z)$.
- Else if $ID_i = ID_S^*$, $ID_j \neq ID_V^*$ or $ID_i = ID_V^*$, $ID_j \neq ID_S^*$, $\mathcal{C}$ first finds <ID_j, w_j> in list $\mathcal{SK}^{list}$ and <P_{pub}, ID_i, Y_i, e_i> in list H_1^{list}. Then $\mathcal{C}$ randomly selects $z, \beta, \alpha \in \mathbb{Z}_q^*$, and computes $T = zP - \beta(X_i + Y_i + e_i P_{pub})$, $c_1 = w_j(T + X_i + Y_i + e_i P_{pub})$ and $c_2 = \alpha P$. Then it makes a H$_2$(ID_i, ID_j, m, T, c_1, c_2) query itself to get h and caculates $r = \beta - h \bmod q$, $k = w_j^{-1}(\alpha - r) \bmod q$. Finally $\mathcal{C}$ returns $\sigma = (r, k, h, z)$.
- Otherwise, if $ID_i = ID_S^*$, $ID_j = ID_V^*$ and $m = m^*$, $\mathcal{C}$ knows the corresponding w_j. Then it finds <P_{pub}, ID_i, Y_i, e_i> in list H_1^{list}, randomly chooses $z, \beta, \alpha \in \mathbb{Z}_q^*$ and proceeds in accordance with the above process. Finally $\mathcal{C}$ returns $\sigma = (r, k, h, z)$.

• SIMUL(ID_i, ID_j, m) query: $\mathcal{C}$ queries SIGN(ID_j, ID_i, m) by itself, and returns whatever it gets.

- VERI(ID_i, ID_j, m, σ): When $\mathcal{F}_I$ queries with input a signature $\sigma = (r,k,h,z)$, $\mathcal{C}$ first confirms whether $(ID_i, ID_j) = (ID_S^*, ID_V^*)$ or $(ID_i, ID_j) = (ID_V^*, ID_S^*)$ and $m = m^*$ holds.
  - If so, it returns "$\perp$".
  - Otherwise, $\mathcal{C}$ searches list $\mathcal{PK}^{list}$ to obtain tuple <ID_i, X_i, Y_i>, and list $\mathcal{SK}^{list}$ for tuple <ID_j, w_j>. Here notice, if $ID_j = ID_V^*$, it knows the corresponding w_j. If such tuples do not exist, it queries ReqestPK(ID_i) and/or ExtrFullSK(ID_j) to get them. Then it verifies whether σ is valid by running the Signature Verification algorithm, and returns the result it gets.

Eventually, $\mathcal{F}_I$ forges a valid signature $(ID_A, ID_B, m^*, \sigma = (r,k,h,z))$. If $(ID_A, ID_B) \neq (ID_S^*, ID_V^*)$, $\mathcal{C}$ outputs "$\perp$". Otherwise, $\mathcal{C}$ finds $< P_{pub}, ID_S^*, Y_S^*, e_S^* >$ in list H_1^{list} . Then $\mathcal{C}$ executes $\mathcal{F}_I$ again for a second run, while the input A=aP remain unchanged, and also keep the same coins used for both $\mathcal{C}$ and $\mathcal{F}_I$. The difference between the two runnings is that $\mathcal{C}$ changes the method to answer H$_1$ queries during the second run. Before a particular query of H$_1$(P_{pub}, ID_V^* , Y_V^*), $\mathcal{C}$ responds the queries identically as in the first run. To this query H$_1$(P_{pub}, ID_V^* , Y_V^*), $\mathcal{C}$ responds with a new independent value $e' \in \mathbb{Z}_q^*$. Subsequent queries to H$_1$ are also reponsed by randomly choosing from $\mathbb{Z}_q^*$.

Applying the "forking" technique formalized in Reference [28], $\mathcal{C}$ can get another valid forged signature tuple $(ID_A, ID_B, m^*, \sigma' = (r,k,h',z'))$, such that $h \neq h'$. If $\mathcal{F}_I$ can forge a correct valid signature successfully, then it must request the random oracle H$_2$ for the hash value h, and the input (T, c_1, c_2) must be correct.

Thus $\mathcal{C}$ can find the pair (T, c_1, c_2) and (T, c_1', c_2'). in list H_2^{list} . Then $\mathcal{C}$ computes the following values:

$$
\begin{aligned}
W_1 &= (e-e')^{-1}(c_1 - c_1') \\
&= (e-e')^{-1}(t + w_S^*)[(P_V^* + eP_{pub}) - (P_V^* + e'P_{pub})] \\
&= (e-e')^{-1}(t + w_S^*)(e-e')P_{pub} \\
&= (t + w_S^*)P_{pub} \\
W_2 &= (h + r - 1)^{-1}(zP_{pub} - W_1) \\
&= (h + r - 1)^{-1}[(t + (h+r)w_S^*)P_{pub} - W_1] \\
&= (h + r - 1)^{-1}[t + (h+r)w_S^* - t - w_S^*]P_{pub} \\
&= (h + r - 1)^{-1}(h + r - 1)w_S^* P_{pub} \\
&= w_S^* P_{pub} \\
W_3 &= e_S^{*\,-1}(W_2 - (x+y)P_{pub}) \\
&= e_S^{*\,-1}(W_S^* P_{pub} - xP_{pub} - yP_{pub}) \\
&= e_S^{*\,-1}[(y + e_S^* a)aP - yaP] \\
&= e_S^{*\,-1}(e_S^* a^2 P) \\
&= a^2 P
\end{aligned}
$$

Finally, $\mathcal{C}$ produces a correct answer W_3 to the SCDH challenge. Thus, if there exists a forger $\mathcal{F}_I$ who can break our CL-SDVS scheme, then there exists an algorithm who can solve the SCDH problem.

Theorem 2. If the SCDH assumption holds, then our proposed CL-SDVS scheme is existential unforgeable against *Type II forger* under chosen message attacks in the random oracle model.

Proof. Given an ECDLP instance (A, B) = (aP, P), in which $a \in \mathbb{Z}_q^*$ is an unknown random value and $P \in \mathbb{E}$. Suppose $\mathcal{F}_{II}$ be a successful forger against our CL-SDVS scheme. Then an algorithm $\mathcal{C}$ can be constructed by running $\mathcal{F}_{II}$ as a subroutine to break the ECDLP problem. The aim of $\mathcal{C}$ is to compute a.

Algorithm $\mathcal{C}$ simulates the attack environment as follows. It sets msk by selecting a random value $s \in \mathbb{Z}_q^*$, and caculates master public key P_{pub}=sP. Then public parameters are released, and msk =s is sent to $\mathcal{F}_{II}$, simulates the oracle queries of $\mathcal{F}_{II}$ .We only describe ReqestPK(ID_i) oracle below without losing generality, since other oracles are essentially the same as in **Theorem 1**.

- ReqestPK(ID_i) query: $\mathcal{C}$ maintains list $\mathcal{PK}^{list}$ with entries of format<ID_i, X_i, Y_i>. When $\mathcal{F}_{II}$ makes this query on ID_i, $\mathcal{C}$ proceeds as follows.
  - If ID_i already consists in the list $\mathcal{PK}^{list}$ in a tuple <ID_i,X_i, Y_i>, then $\mathcal{C}$ returns $PK_{ID_i} = (X_i, Y_i)$.
  - Else if $ID_i = ID_S^*$, $\mathcal{C}$ generates $Y = dP - eP_{pub}$ by randomly picking $d, e \in \mathbb{Z}_q^*$, and computes X=A-Y, where (A, B) is the ECDLP challenge instance. Then it stores <ID_i, X, Y> in list $\mathcal{PK}^{list}$, inserts <P_{pub}, ID_i, Y, e> into the list H_1^{list} , and returns in output $PK_{ID_i} = (X, Y)$.
  - Else if $ID_i = ID_V^*$, $\mathcal{C}$ chooses $x, d, e \in \mathbb{Z}_q^*$ and caculates $X = xP$, $Y = dP - eP_{pub}$. Then stores <ID_i, X, Y> in list $\mathcal{PK}^{list}$, inserts <P_{pub}, ID_i, Y, e> into the list H_1^{list} , and returns $PK_{ID_i} = (X, Y)$.
  - Otherwise, $\mathcal{C}$ makes an ExtrFullSK(ID_i) query on ID_i itself to refresh its list $\mathcal{PK}^{list}$ and returns in output $PK_{ID_i} = (X_i, Y_i)$.

Eventually, $\mathcal{F}_{II}$ produces its forgery $(ID_S^*, ID_V^*, m^*, \sigma = (r,k,h,z))$. $\mathcal{C}$ runs $\mathcal{F}$ again for a second time just as described in the proof of **Theorem 1** to get another valid forgery $(ID_S^*, ID_V^*, m^*, \sigma' = (r, k, h', z'))$ such that $z \neq z'$. Then $\mathcal{C}$ calculates the following values:

$$\begin{aligned} W &= (h-h')^{-1}(z-z') - es \\ &= (h-h')^{-1}[t+(r+h)w_S^* - t - (r+h')w_S^*] - es \\ &= (h-h')^{-1}(h-h')w_S^* - es \\ &= x_S^* + y_S^* + es - es = x_S^* + y_S^* \\ &= a \end{aligned}$$

Finally, $\mathcal{C}$ produces a correct answer W to the ECDLP challenge. Thus, if there exists a forger $\mathcal{F}_{II}$ who can break our CL-SDVS scheme, the ECDLP problem would be solved.

Theorem 3. Suppose $\mathcal{F}$ be a delegated signer who can generate valid signatures on message $m \in \mathcal{M}$ in polynomial time τ. The probability ε that $\mathcal{F}$ succeeds is not negligible. Then the CL-SDVS scheme is ($56\tau/\varepsilon$, $1/q$)-non-delegatable under the random oracle model.

Proof. If there exists a delegated signer $\mathcal{F}$, then we can construct an efficient knowledge extractor E. On input signature σ, E can get access to $\mathcal{F}$ as a black-box oracle, and produce the private key of either ID_S or ID_V in expected time $\tau' \le 56\tau/\varepsilon$. We assume $\varepsilon > k$ where $k=1/q$. The probability that E succeeds is 1.

Suppose $\mathcal{F}_m$ be $\mathcal{F}$ with input m. The extractor E executes $\mathcal{F}_m$ twice with the same random inputs. Except for the responses to the hash query $H_2(ID_S, ID_V, m, T, c_1, c_2)$, E executes $\mathcal{F}_m$ step-by-step in both runnings. In the second running, E returns a different random value as the answer. However, the tuple (T, c_1, c_2) must be equal in both runnings, since it is under the hash-function H_2. If both signatures are valid, we should have $t = t'$, i.e. $z-(r+h)w_S = z'-(h'+r')w_S$ mod q, and $c_2 = c_2'$, i.e. $r + k \cdot w_V = r' + k' \cdot w_V$ mod q. Here (r, k, h, z) is the signature in the first run, and (r', k', h', z'), is the signature in the second run, where $h \neq h'$.

Now we show how to extract user's private key. If $r = r'$ mod q, then $h + r \neq h' + r'$ mod q due to the fact that $h \neq h'$. Thus we can produce $w_S = (z-z')/(h-h')$ mod q, derived from the equation $z-(r+h)w_S = z'-(h'+r')w_S$ mod q. If $r \neq r'$ mod q, one can find $w_V = (r-r')/(k'-k)$ mod q derived from the equation $r + k \cdot w_V = r' + k' \cdot d_V$ mod q. Thus the private key (w_S or w_V) is sucessfully extracted with probability 1.

There is a remain issue how to generate two different valid signatures on m, i.e. (r, k, h, z) and (r', k', h', z'), satisfying that $h \neq h'$ but $(T, c_1, c_2) = (T, c_1', c_2')$. Now we construct an algorithm *Rewind* to settle it. *Rewind* can get access to $\mathcal{F}_m$ as a black-box oracle. If every request to $\mathcal{F}_m$ is counted as one step, then *Rewind* runs in expected time $56/\varepsilon$. Algorithm $\mathcal{F}_m$ can produce a valid signature (d, k, h, z) with probability at least ε. The procedure is related to the random outputs of $H_2(ID_S, ID_V, m, T, c_1, c_2)$ and the random coins toss.

To respond with the two different challenges correctly, we execute the prover and use rewinding. The random coins are supplied in the following way. First, we construct a matrix $\mathcal{H}$. The row of $\mathcal{H}$ is the probable random coins for $\mathcal{F}_m$, while the column of $\mathcal{H}$ is each probable H_2 value h. Then we probe the entries we need in $\mathcal{H}$, by taking $\mathcal{F}_m$ as a black-box oracle. If $\mathcal{F}_m$ responds correctly, we record 1 in the corresponding entry. If not, we record 0. The aim is to discover two 1's in the same row, which is equivalent to finding two different valid signatures. The probability ε equals the ratio of 1-entries in $\mathcal{H}$. According to Reference [4], algorithm *Rewind* can discover the required 1 entries in time $56/\varepsilon$.

Theorem 4. Our CL-SDVS scheme satisfies non-transferability.

Proof. By executing ***Sign*** algorithm, the signer ID_S can produce valid signature $\sigma = (r, k, h, z)$ on arbitrary m, where $h = H_2(ID_S, ID_V, m, T, c_1, c_2)$. Meanwhile, by executing ***Simulation*** algorithm, the designated verifier ID_V can also compute valid signatures. For arbitrary probabilistic polynomial-time distinguisher, it is infeasible to discriminate whether the signatrure is caculated by a real signer, or simulated by the specified verifier.

Theorem 5. Our CL-SDVS scheme satisfies strongness.

Proof. The ***Verify*** algorithm in our CL-SDVS scheme needs to use the private key $SK_{ID_V} = w_V$ of the designated verifier. This would guarantee that any third entity cannot verify the validity or invalidity of the signature, without obtaining the corresponding private key. More precisely, any third party cannot distinguish the signature from a random string with identical length and distribution. Thus our proposed scheme inherently satisfies the security property of strongness.

Table 1 Security comparison

Schemes	Strongness	Unforgeable against Type I forger	Unforgeable against Type II forger	Non-transferability	Nondelegatable	Assumptions
Our scheme	Y	Y	Y	Y	Y	SCDH&ECDL
H-scheme	N	Y	N	Y	N	GBDH
D-scheme	N	N	Y	Y	N	GBDH
C-scheme	N	Y	Y	Y	N	CDH&GBDH
Y-scheme	Y	Y	Y	Y	--	CDH&BDH
X-scheme	Y	N	N	Y	--	CDH&BDH
I-scheme	Y	Y	Y	--	--	CDH&BDH

We provide a security comparison of our proposed scheme with known existing CL-DVS schemes (H-scheme [16], D-scheme [18], C-scheme [17], Y-scheme [23], X-scheme [22] and I-scheme [21]) in Tab. 1, which demonstrates that most existing schemes cannot satisfy all the required security properties, while ours can achieve all.

We also provide formal security proofs of unforgeability against both *Type I* and *Type II* forgers, non-transferability, strongness and non-delegatability above. To the best of our knowledge, it is the first CL-SDVS scheme without pairings that provides formal security proof of non-delegatability.

5 PERFORMANCE EVALUATION

As shown in Tab. 2, we present a performance comparison of the proposed CL-SDVS with H-scheme [16], C-scheme [17], Y-scheme [23], X-scheme [22] and I-scheme [20]. The analysis inculdes the required computational cost of signing or verifying a siganature, and the communication cost which is measured by the signature length. Here we omit the cost of conventional hash operations and point addition operations for simplicity. Comparing with bilinear pairings and scalar multiplications, these operations are much more efficient and negeleted. Denote P be bilinear pairing operation, E be pairing-based scalar multiplication, E_M be ECC-based scalar multiplication, E_X be exponentiation in group $\mathbb{G}$, H be map-to-point hash operation, I be inverse operation, and *"pre"* be pre-computed operation. Assume the bit-length of group element in $\mathbb{G}$ is 512 bits, and the goup order q = 160 bits.

Table 2 Performance comparison

Schemes	Sign-Cost	Verify-Cost	Length(bits)
Our scheme	$3E_M$+ (2*pre* E_M)	6 E_M	4| q | = 640 bits
H-scheme	$1P + 1E + 1I + 1H$	$3P + 1E$	| q | = 160 bits
C-scheme	$1P + 1E + 1H$	$1P + 1E + 1H$	| q | = 160 bits
Y-scheme	$1P + 4E + 1H$	$1P + 2E + 1H$	2| G | = 1024 bits
X-scheme	$3E + 1I + 1H$	$2P + 1E + 1E_X + 2H$	2| G | = 1024 bits
I-scheme	(2*pre* P)+ $3E + 2E_X$	$1P + 1E$	2| G | = 1024 bits

To evaluate the computational efficiency, we follow the experimental results made in Reference [29], which estimated the running time of representative cryptographic operations. The time for executing the bilinear piring operation is approximately equal to 20.01 ms, i.e. $T_P \approx 20.01$ ms . Similarly, $T_E \approx 6.38$ ms , $T_{E_M} \approx 0.83$ ms , $T_{E_X} \approx 11.20$ ms . Fig. 1 shows the computational cost of signature creation and verification for all the compared schemes, while the time needed to execute the pre-computed operations is neglected.Without using any expensive computations like bilinear parings, our scheme is the most computationally efficient.

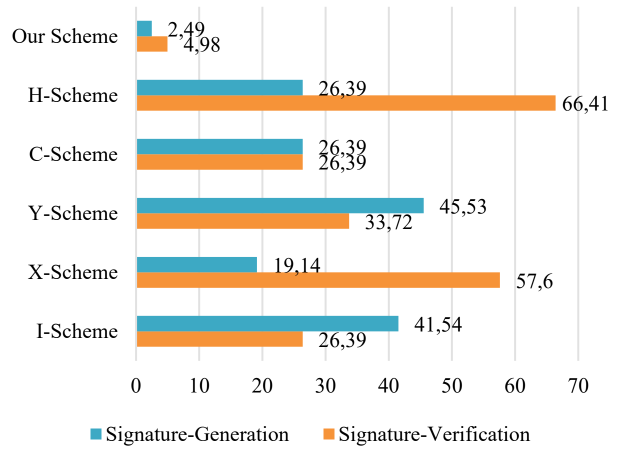

Figure 1 Evaluation of approximate running time.

6 CONCLUSIONS

This paper presented the first certificateless strong designated verifier signature scheme without using pairing or map-to-point hash operations. Our scheme satisfies all the required security properties of CL-SDVS, that is unforgeability against both Type I and Type II forgers, non-transferability, strongness and the stronger notion non-delegatability. The formal security proofs were based on the intractability of SCDH and ECDL assumptions in the random oracle model. Furthermore, our CL-SDVS scheme achieves higher efficiency and outperforms all the known CL-(S)DVS schemes in this literature. The signature length is also very short.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61702570 and U1509214, the National Key Research and Development Program of China under Grant No. 2017YFB1400701, and the Foundation of Science and Technology on Information Assurance Laboratory under Grant No.KJ-14-008.

7 REFERENCES

[1] Jakobsson, M., Sako, K. & Impagliazzo, R. (1996). Designated verifier proofs and their applications. *EUROCRYPT*, LNCS 1070, Springer, 143-154. https://doi.org/10.1007/3-540-68339-9_13

[2] Saeednia, S., Kremer, S., & Markowitch, O. (2003). An Efficient Strong Designated Verifier Signature Scheme. In *6th International Conference on Information Security and Cryptology* (*ICISC*), LNCS 2971, Springer, 40-54. https://doi.org/10.1007/978-3-540-24691-6_4

[3] Laguillaumie, F. & Vergnaud, D. (2004). Designated verifier signatures: anonymity and efficient construction from any bilinear map. *Security in Communication Networks*, LNCS 3352, Springer, Berlin, 105-119. https://doi.org/10.1007/978-3-540-30598-9_8

[4] Lipmaa, H., Wang, G., & Bao F. (2005). Designated verifier signature schemes: attacks, new security notions and a new construction. In *32nd International Colloquium on Automata, Languages and Programming*, Springer, 459-471. https://doi.org/10.1007/11523468_38

[5] Steinfeld, R., Bull, L., Wang, H., & Pieprzyk, J. (2003). Universal Designated-Verifier Signatures. *ASIACRYPT*, LNCS 2894, Springer, Heidelberg, 523-542. https://doi.org/10.1007/978-3-540-40061-5_33

[6] Steinfeld, R., Wang, H., & Pieprzyk J. (2004). Efficient Extension of Standard Schnorr/RSA Signatures into Universal Designated-Verifier Signatures. *International Workshop on Public Key Cryptography*, 86-100. https://doi.org/10.1007/978-3-540-24632-9_7

[7] Yang, X. D., Xiao, L.K., Chen, C. L., & Wang, C. F. (2018). A Strong Designated Verifier Proxy Re-Signature Scheme for IoT Environments. *Symmetry*, 10(11), 580. https://doi.org/10.3390/sym10110580

[8] Kuchta, V., Sahu, R. A., Saraswat, V., Sharma, G., Sharma, N., & Markowitch, O. (2018). Anonymous yet Traceable Strong Designated Verifier Signature. *International Conference on Information Security*. Springer, 403-421. https://doi.org/10.1007/978-3-319-99136-8_22

[9] Khan, A. U., Ratha, B. K., & Mohanty, S. (2017). A Timestamp-Based Strong Designated Verifier Signature Scheme for Next-Generation Network Security Services. *Advances in Computer and Computational Sciences*. Springer, 311-320. https://doi.org/10.1007/978-981-10-3770-2_29

[10] Duan, M. J., Xu J., & Feng D. G. (2013). Efficient identity-based strong designated verifier signature schemes. *Security and Communication Networks, 6*(7), 902-911. https://doi.org/10.1002/sec.645

[11] Huang, Q., Yang, G., Wong, D. S., & Susilo, W. (2011). Identity-Based Strong Designated Verifier Signature Revisited. *The Journal of Systems and Software, 84*(1), 120-129. https://doi.org/10.1016/j.jss.2010.08.057

[12] Susilo, W., Zhang, F., & Mu, Y. (2004). Identity-based strong designated verifier signature schemes. *9th Australasian Conference on Information Security and Privacy* (ACIS'04), LNCS 3108, 313-324. https://doi.org/10.1007/978-3-540-27800-9_27

[13] Zhang, Y., Zhang, Y., Li, Y., & Wang, C. (2015). Strong Designated Verifier Signature Scheme Resisting Replay Attack. *Information Technology & Control, 44*(2), 165-171. https://doi.org/10.5755/j01.itc.44.2.7625

[14] Shooshtari, M. K., Ahmadian-Attari, M., & Aref, M. R. (2017). Provably secure strong designated verifier signature scheme based on coding theory. *International Journal of Communication Systems, 30*(7), e3162. https://doi.org/10.1002/dac.3162

[15] Al-Riyami, S. S. & Paterson, K. G. (2003). Certificateless Public Key Cryptography. *ASIACRYPT*, LNCS 2894, 452-473. https://doi.org/10.1007/978-3-540-40061-5_29

[16] Huang, X., Susilo, W., Mu, Y., & Zhang, F. (2006). Certificateless Designated Verifier Signature Schemes. *20th International Conference on Advanced Information Networking and Applications*, AINA'06, IEEE, 15-19. https://doi.org/10.1109/AINA.2006.124

[17] Chen, H., Song, R., Zhang, F., & Song, F. (2008). An efficient certificateless short designated verifier signature scheme. *Wireless Communications, Networking and Mobile Computing (WiCOM)*, IEEE, 1-6. https://doi.org/10.1109/WiCom.2008.1107

[18] Du, H. & Wen, Q. (2009). Efficient and provably-secure certificateless short signature scheme from bilinear pairings. *Computer Standards & Interfaces, 31*(2), 390-394. https://doi.org/10.1016/j.csi.2008.05.013

[19] He, D. & Chen J. (2013). An Efficient Certificateless Designated Verifier Signature Scheme. *International Arab Journal of Information Technology, 10*(4), 389-396.

[20] Islam, S. H. & Biswas, G. P. (2012). Certificateless Strong Designated Verifier Multisignature Scheme Using Bilinear Pairings. *International conference on advances in computing, communications and informatics* (*ICACCI)*, ACM, 540-546. https://doi.org/10.1145/2345396.2345485

[21] Islam, S. H. & Biswas, G.P. (2013). Provably secure certificateless strong designated verifier signature scheme based on elliptic curve bilinear pairings. *Journal of King Saud University-Computer and Information Sciences*, 2013, 25(1), 51-61. https://doi.org/10.1016/j.jksuci.2012.06.003

[22] Xiao, Z., Yang, B., & Li, S. (2010). Certificateless strong designated verifier signature scheme. *Proceedings of the 2nd International Conference on e-Businessand Information System Security*, EBISS 2010, IEEE, 1-5. https://doi.org/10.1109/EBISS.2010.5473435

[23] Yang, B., Hu, Z., & Xiao, Z. (2009). Efficient certificateless strong designated verifier signature scheme. *Computational Intelligence and Security (CIS)*, IEEE, 432-436. https://doi.org/10.1109/CIS.2009.191

[24] Chen, Y., Zhao, Y., Xiong, H., & Feng, Y. (2017). A Certificateless Strong Designated Verifier Signature Scheme with Non-delegatability. *IJ Network Security*, 19(4), 573-582. https://doi.org/10.6633/IJNS.201707.19(4).10

[25] Zhang, J. & Xie, J. (2011). Breaking a certificateless strong designated verifier signature scheme. *Consumer Electronics, Communications and Networks (CECNet)*, IEEE, 130-133. https://doi.org/10.1109/CECNET.2011.5768814

[26] Pakniat, N. (2018). On the Security of a Certificateless Strong Designated Verifier Signature Scheme. *eprint-2018-915.*

[27] Bao, F., Deng, R. H., & Zhu, H. (2003). Variations of Diffie-Hellman problem. *Information and Communications Security(ICICS)*, LNCS 2836, Springer, 301-312. https://doi.org/10.1007/978-3-540-39927-8_28

[28] Pointcheval, D. & Stern, J. (1996). Security Proofs for Signature Schemes. *EUROCRYPT*, Springer, Berlin, Heidelberg, 387-398. https://doi.org/10.1007/3-540-68339-9_33

[29] Cao, X., Kou, W., & Du, X. (2010). A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges. *Information Sciences, 180*(15), 2895-2903. https://doi.org/10.1016/j.ins.2010.04.002

Contact information:

Meijaio DUAN, Dr.
School of Information, Central University of Finance and Economics
39 South College Road, Haidian District, Beijing, P. R. China (100081)
E-mail: duanmeijiao@cufe.edu.cn

Jianming ZHU, Prof.
School of Information, Central University of Finance and Economics
39 South College Road, Haidian District, Beijing, P. R. China (100081)
E-mail: tyzjm65@163.com

Yang LI, Assoc. Prof.
School of Information, Central University of Finance and Economics
39 South College Road, Haidian District, Beijing, P. R. China (100081)
E-mail: liyang@cufe.edu.cn