Front. Comput. Sci., 2020, 14(4): 144806
https://doi.org/10.1007/s11704-019-8320-0

RESEARCH ARTICLE

Zero-pole cancellation for identity-based aggregators: a constant-size designated verifier-set signature

E CHEN[1], Yan ZHU (✉)[1], Changlu LIN[2,3], Kewei LV[4]

1 School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China
2 College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350117, China
3 Fujian Provincial Key Lab of Network Security & Cryptology, Fujian Normal University, Fuzhou 350007, China
4 Institute of Information Engineering, DCS Research Center, Chinese Academy of Sciences, Beijing 100093, China

Abstract In this paper we present a designated verifier-set signature (DVSS), in which the signer allows to designate many verifiers rather than one verifier, and each designated verifier can verify the validity of signature by himself. Our research starts from identity-based aggregator (IBA) that compresses a designated set of verifier's identities to a constant-size random string in cryptographic space. The IBA is constructed by mapping the hash of verifier's identity into zero or pole of a target curve, and extracting one curve's point as the result of aggregation according to a specific secret. Considering the different types of target curves, these two IBAs are called as zeros-based aggregator and poles-based aggregator, respectively. Based on them, we propose a practical DVSS scheme constructed from the zero-pole cancellation method which can eliminate the same elements between zeros-based aggregator and poles-based aggregator. Due to this design, our DVSS scheme has some distinct advantages: (1) the signature supporting arbitrary dynamic verifiers extracted from a large number of users; and (2) the signature with short and constant length. We rigorously prove that our DVSS scheme satisfies the security properties: correctness, consistency, unforgeability and exclusivity.

Keywords designated verifier-set signature, aggregator, unforgeability, exclusivity

Received September 20, 2018; accepted July 19, 2019

E-mail: zhuyan@ustb.edu.cn

1 Introduction

Jakobsson et al. [1] introduced the concept of designated verifier proof (DVP) in which only the verifier designated by the confirmer can obtain any conviction on the correctness. As a good example of such a situation, the designated verifier undeniable signatures (DVUS) were presented to guarantee that only the intended verifier can be convinced about the validity or invalidity of the signature for the confirmation protocol. Following the above work, Steinfeld et al. [2] and Saeednia et al. [3] introduced the designated verifier signatures (DVS), which provides message authentication but does not require public verification of traditional signatures. The purpose of doing so is to enable the signers to have complete control over their signatures. Such signature schemes have a broad and special application prospect in the e-commerce and e-government, such as electronic voting and electronic contract signing.

For some fair contracts and distributed applications, the DVS notion has been extended to multi-verifier settings. This new kind of DVS is usually called multi-designated verifiers signature (MDVS), in which the designated signature corresponds to a group of specific verifiers. Exactly, it differs from DVS in that the signer wants to convince multiple verifiers of the signature's validity in a cooperative way. There are MDVS constructions, for example, Ng et al.'s scheme [4] is

concentrated on the random oracle model, Shailaja et al.'s scheme [5] without random oracle, Chang's scheme [6] for multi-signer and multi-verifier. These MDVS schemes allow the signer or a signature holder to designate multiple verifiers, but it requires that all designated verifiers must cooperate to verify the validity of signature rather than a verifier.

Obviously, the cooperative work mode of MDVS is not realistic for some applications, e.g., each designated verifier expects to verify the validity of signature by oneself. The cooperative way also implies the verification of signature is not complete or cannot be executed correctly only if one of all verifiers is missing, refuses to take part in, or maliciously conflicts with the verification. In addition, these MDVS schemes have a complicated cryptographic framework, e.g., the Steinfeld et al.'s scheme [2] consists of seven algorithms and a verifier key-resignated protocol. This design makes it difficult to apply for normal document or data signature. Aimed at these above-mentioned problems, in this paper we intend to develop a new type of DVS, called designated verifier-set signature (DVSS), for supporting the individual verification of the designated verifiers in an independent way, i.e., each of the designed verifiers is able to verify the validity of signature. To provide ease of use, the DVSS scheme is expected to build on the identity-based cryptosystem (IBC), in which the public key is the owner's identity information, e.g., user's name and email address. This design called identity-based DVSS can simplify the management of public key and naturally solve the binding problem between public key and entity information.

Our contributions The challenge problem for constructing a practical DVSS scheme lies in how to compress a designated set V of arbitrary size into a constant-size element. Moreover, we require that this compression process works in an efficient and secure way in order to prevent the adversary from tampering with the designated verifier-set. To deal with this challenge, we refer to the compression method in existing literatures, e.g., Merkle-tree-based accumulator [7], which is a binary tree that hashes a set of inputs into a short, constant-size random string. In terms of this idea of accumulator, we provide a new approach that combines all elements of designated verifier-set into an algebra curve, rather than a binary tree. For the purpose of maximizing compression, we only pick up a determined point on the above curve as a compressed representation of given verifier-set. This compression process is called an aggregator. We require that the range of values of the aggregator should be large enough and its distribution is uniform, so as to achieve the purpose of difficult-to-temper.

This approach also leads to another challenge for the choice of algebraic curve. Aimed at this challenge, two kinds of curves would be constructed by mapping all elements in the designed set into the curve's zeros or poles, respectively. The advantage of generating zero-pole-based curves lies in the ability to eliminate the same elements between zero-based curve and pole-based curve by means of zero-pole cancellation. Furthermore, we make this method to detect the existence of element from a given set. This means we can verify whether the verifier belongs to a designated verifier-set.

In this paper, we intent to present a DVSS scheme by applying for the approach as mentioned above. In DVSS, the signer allows to designate many verifiers rather than one verifier, and each of designated verifiers is able to verify the validity of the signature by himself. To implement an effective DVSS scheme, our research starts from identity-based aggregator (IBA) that compresses a designated set of verifier's identities to a constant-size random string in cryptographic space. The IBA is constructed by mapping the hash of verifier's identity into zero or pole of a target curve (expressed by pseudorandom polynomial), and extracting one curve's point as the result of aggregation according to a specific secret. Considering the different types of target curves, we call two above-mentioned IBAs as zeros-based aggregator (ZerosAggr) and poles-based aggregator (PolesAggr), respectively.

In terms of ZerosAggr and PolesAggr, we further propose a practical DVSS scheme constructed from the zero-pole cancellation method. In DVSS, the signer encapsulates the result of PolesAggr for a given verifier set as the signature. For a given DVSS signature, the verifier firstly uses the public key to produce the result of ZerosAggr over the verifier-set except himself. And then, a commitment of pole (mapped from verifier-identity) with secret is retrieved by using Zero-Pole Cancellation and used to complete the final verification of signature. Due to this design, our DVSS scheme has some distinct advantages, such as, no limitation on the number of designated verifiers, constant-size signature (merely five elements) and user's private key, public key with user's profiles etc. Based on the random oracle model, we obtain:

- The first designated verifier-set signature for arbitrary dynamic verifiers from a large number of users, so far, all existing DVSS schemes only deal with static and fixed number of verifiers.
- The designated verifier-set signature with short and constant length.

Under the existing security models, our DVSS provably

satisfies the general security properties, e.g., correctness and consistency. Additionally, the unforgeability of our DVSS scheme can be hold under the strong Diffie-Hellman (SDH) assumption in the random oracle model. In addition, we intend to introduce a new security notion called exclusivity. Furthermore, we present a formal definition of exclusivity, and then rigorously prove that our DVSS scheme has exclusivity under the general Diffie-Hellman exponent (GDHE) assumption with collusion attacks.

Related work Jakobsson et al. [1] firstly introduced the designated verifier proofs, both interactive and non-interactive, based on trap-door commitment. Based on this concept, Steinfeld et al. [2] provided the first universal designated verifier signature (UDVS) construction. Saeednia et al. [3] also proposed a more intuitive definition and construction of strong DVS scheme based on the simulator in zero-knowledge proof. However, they only proved that the construction is existentially unforgeable under non-chosen message attacks.

After the identity-based encryption (IBE) were presented, most of the proposed DVSs, e.g., [8–10], were constructed on identity-based systems without public key certificate. For example, Susilo et al. [11] proposed a generic construction of identity-based strong DVS with low communication and computation cost. And then, the security of this construction had been extended to not only non-delegatability [12], but also delegatability [13]. There are some other schemes, the security of which is closely related to the bilinear Diffie-Hellman (BDH) problem [14] in the random oracle model, e.g., [15, 16]. Though most of identity-based DVSs claimed that the scheme satisfied the security requirements of all designated verifier signature, the proofs for these properties were not provided in the literature we reviewed. It is worth mentioning that an efficient strong DVS on identity-based setting was proposed by Sharma et al. [17], where the scheme was strong existentially unforgeable against adaptive chosen message attack (EUF-CMA) and adaptive chosen identity attack (EUF-CIA). The identity-based DVS solves the management problems of public key certificate in the traditional PKI, and it has broad application prospects.

Generally, a DVS signature can only be verified by a unique and specific user. Jakobsson et al. [1] have discussed how their results can be extended to multiple designated verifiers. And then, Laguillaumie and Vergnaud [18] put forward a formal definition of multi-designated verifiers signature and a security proof. Subsequently, the MDVS schemes were continuously developed, e.g., [4, 19]. Ming and Wang [20] presented a universal MDVS scheme according to the gap bilinear Diffie-Hellman assumption, and proved its security in the standard model. Seo et al. [21] proposed an identity-based universal MDVS scheme by extending a single verifier to a set of multi-verifiers. But it is a pity that these schemes require collaboration of designated verifiers to verify an ID-based MDVS signature. The other noteworthy work also includes [22–25]. However, neither of all mentioned schemes were designed to support a set of specific verifiers, which can independently verify the validity of signature.

Organization Section 2 overviews some basic notions and complexity assumptions. In Section 3, the definition of DVSS is addressed. In Section 4, we propose a practical DVSS scheme. And then, the security models are given in Section 5. Finally, we conclude this paper in Section 6.

2 Preliminaries

In this paper, define $\mathcal{U}$ be a full set of all users in a system and each user has a unique identity ID denoted by a string $\{0,1\}^*$. Denote $\mathcal{U}$ as the identity, that is, $\mathcal{U} = \{\mathrm{ID}_1, \ldots, \mathrm{ID}_n\}$. A function $f : \mathbb{N} \mapsto \mathbb{R}$ is considered as a negligible function, if for any $c > 0$ there exists $n_0 \in \mathbb{N}$ such that $f(n) < 1/n^c$ for all $n > n_0$. Also a probability function $g : \mathbb{N} \mapsto \mathbb{R}$ is considered as overwhelming if the function $h(n) = 1 - g(n)$ is a negligible function. For a probabilistic algorithm $\mathcal{A}$, $\mathcal{A}(x, r)$ is used to denote the output of $\mathcal{A}$ on input x with a random input r. If r is not explicitly specified, it do so with the understanding that r is chosen statistically independent of all other variables.

2.1 Bilinear pairing

The basic definition of bilinear pairings proposed by Boneh and Franklin [26, 27] is briefly reviewed. Define $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot, \cdot))$ be a bilinear group system if there exists an efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that satisfies the following definition.

Definition 1 (Bilinear pairing) Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two additive cyclic groups with prime order p, and define G, H as the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Let $\mathbb{G}_T$ be a multiplicative cyclic group of same order p using elliptic curve conventions. For any $G \in \mathbb{G}_1$, $H \in \mathbb{G}_2$ and all $a, b \in \mathbb{Z}_p$, a bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties.

- Bilinearity: $e([a]G, [b]H) = e(G, H)^{ab}$.
- Non-degeneracy: $e(G, H) \neq 1$ unless $G = 1$ or $H = 1$.

- Computability: $e(G, H)$ is efficiently computable.

2.2 Security problems

The security of the presented DVSS scheme depends on the q-strong Diffie-Hellman (SDH) assumption and the general Diffie-Hellman exponent (GDHE) assumption. The SDH problem is defined as follows.

Definition 2 (q-SDH problem [28]) Define $\mathbb{G}, \mathbb{H}$ be two additive cyclic groups of prime p. Let g be a generator of $\mathbb{G}$ and h a generator of $\mathbb{H}$. The q-SDH problem in $(\mathbb{G}, \mathbb{H})$ is defined as follows. Given a $(q+2)$-tuple $(g, g^{\gamma}, g^{\gamma^2}, \ldots, g^{\gamma^q}, h) \in \mathbb{G}^{q+1} \times \mathbb{H}$, output a pair $(\zeta, g^{\frac{1}{\gamma+\zeta}})$, where $\zeta \in \mathbb{Z}_p^*$. An algorithm $\mathcal{A}$ has advantage ϵ in solving q-SDH in $(\mathbb{G}, \mathbb{H})$ if

$$\Pr[\mathcal{A}(g, g^{\gamma}, g^{\gamma^2}, \ldots, g^{\gamma^q}, h) = (\zeta, g^{\frac{1}{\gamma+\zeta}})] \geqslant \epsilon,$$

where the probability is over the random choice of $\gamma \in \mathbb{Z}_p^*$ and the random bits consumed by $\mathcal{A}$.

The problem is considered hard, such that the following assumption holds.

Definition 3 ((q, t, ϵ)-SDH assumption [28]) The (q, t, ϵ)-SDH assumption holds in $(\mathbb{G}, \mathbb{H})$ if no t-time algorithm has advantage at least ϵ in solving the q-SDH problem in $(\mathbb{G}, \mathbb{H})$.

The complexity of this problem has been elaborated in the following theorem.

Theorem 1 (Theorem 2 in [29]) Let g be an element of prime order p in an abelian group. Suppose that d is a positive divisor of $p+1$ and $g_i := G^{\gamma^i}$ for $i = 1, 2, \ldots, d$ are given. Then γ can be computed in $O(\log p \cdot (\sqrt{(p+1)/d} + d))$ group operations using $O(\max\{\sqrt{(p+1)/d}, \sqrt{d}\})$ memory.

Given $\langle g, g^{\gamma}, \ldots, g^{\gamma^d} \rangle$ for a positive divisor d of $p+1$, assume that d is large enough, this paper also indicates the strong Diffie-Hellman problem and its related problems have computational complexity reduced by $O(\sqrt{d})$ from that of the discrete logarithm problem for such primes.

The general Diffie-Hellman exponent (GDHE) problem is a more fundamental problem on the general bilinear map group system $\mathbb{S}$. Just consider the weakest case $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ in $\mathbb{S}$. Then, the GDHE framework is overviewed.

Definition 4 (GDHE problem [30]) Let $P, Q \in \mathbb{F}_p[X_1, \ldots, X_m]^l$ be two l-tuples of m-variate polynomials over $\mathbb{F}_p$, where $P = (p_1, \ldots, p_l), Q = (q_1, \ldots, q_l)$ and $l, m \in \mathbb{Z}^+$. Define $\hat{G}, \hat{H}$ be the generators of additive cyclic group $\mathbb{G}$ with prime order p. Given a vector $S = (\hat{G}^{P(x_1,\ldots,x_m)}, e(\hat{G}, \hat{H})^{Q(x_1,\ldots,x_m)}) \in \mathbb{G}^l \times \mathbb{G}_T^l$ and a random element T in $\mathbb{G}_T$, decide whether or not $T = e(\hat{G}, \hat{H})^{h(x_1,\ldots,x_m)}$, where the polynomial $h \in \mathbb{F}_p[X_1, \ldots, X_m]$.

Almost previous decisional Diffie-Hellman assumptions can be reduced into GDHE assumption. An algorithm $\mathcal{A}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving the GDHE problem if

$$Adv_{\mathrm{GDHE}}(\mathcal{A}) = |\Pr[\mathcal{A}(S, e(\hat{G}, \hat{H})^{h(x_1,\ldots,x_m)}) = 0] - \Pr[\mathcal{A}(S, T) = 0]| > \epsilon.$$

2.3 Forking lemma

The forking lemma applied in the presented DVSS scheme is introduced. In this lemma, considering that the ordinary three-move signature schemes on the input message M produce triplets (σ_1, h, σ_2) independent of previous signature. Those triplets can be described as a commit-challenge-response protocol, where σ_1 is a commitment, h is the hash value of (M, σ_1), and σ_2 just depends on M, σ_1, h.

Lemma 1 (Forking lemma [31]) Define $\mathcal{A}$ be a probabilistic polynomial time Turing machine whose only the public data be taken as input. Assume $\mathcal{A}$ produces a valid signature $(M, \sigma_1, h, \sigma_2)$ with non-negligible probability ϵ. Then there is another machine, which has control over $\mathcal{A}$, outputs two valid signatures $(M, \sigma_1, h, \sigma_2)$ and $(M, \sigma_1, h', \sigma_2')$ such that $h \neq h'$with probability ϵ' much smaller than ϵ.

This lemma uses the oracle replay attack with the same random tape and a different oracle, two signatures of a specific form can be obtained which open a way to solve the underlying hard problem.

3 Definition of the DVSS

In this section, the definition of DVSS over the set of identities is stated for supporting the independent verification of individual verifier. That is, any verifier is able to verify the validity of a signature. For sake of simplicity, suppose that each user has a unique identity ID (e.g., Email address) and the signer only needs to use the users' identities to define the designated verifier-set. Benefited from the identity-based cryptosystem (rather than the traditional CA's certificate), a definition of DVSS is simplified to make it closer to a common signature instead of a special CA's signature in the UDVS/DMVS schemes. Therefore, the DVSS consists of just four algorithms which are defined as follows.

Setup$(\mathbb{S}) \rightarrow (pk, msk)$**.** It takes as input a bilinear map group system $\mathbb{S}$ under the security parameter κ. It outputs a system's public key pk and a master secret key msk, where pk includes a list of users' profiles pp.

KeyGen$_{msk}(\mathrm{ID}_i) \rightarrow sk_i$**.** It takes as input msk and ith user's identity ID_i. It outputs the ith user's secret key sk_i and adds a user's profile pp_i to pp.

Sign$_{pk,sk_i}(M, V) \rightarrow \sigma$**.** It takes as input the public key pk, a signing private key sk_i, a message $M \in \{0,1\}^*$ and the set of designated verifiers $V = \{\mathrm{ID}_j\}_{j=1}^m$. It outputs a signature σ on the message M.

Verify$_{pk,sk_j}(\mathrm{ID}_i, M, \sigma, V) \rightarrow 0/1$**.** It takes as input pk, the jth verifier's secret key sk_j, a signature σ on a message M for a given signer ID_i, and the set of designated verifiers $V = \{\mathrm{ID}_j\}_{j=1}^m$. It outputs 0 (invalid) or 1 (valid).

The number of users in system is not limited, and a new user can be dynamically added to the system by using the **KeyGen** algorithm. The user's profile, some personal information and public cryptographic parameters, is used to omit "Verifier Key-Registration Protocol"[1)] in the UDVS [2]. Moreover, the definition of DVSS does not follow the procedure of the CA's certificate (used in the UDVS and DMVS schemes) because the notion of "Personal Public-key Certificate" has cleared away in the identity-based cryptosystem. Hence, the number of algorithms is reduced from seven [2] to four in the presented DVSS scheme. In addition, a DVSS scheme must meet the following properties, whose general definitions are presented as follows.

Correctness. The valid signature σ produced by the signer ID_i can be accepted as valid by the designated verifier $\mathrm{ID}_j \in V$, that is,

$$\Pr\left[\mathbf{Verify}_{pk,sk_j}(\mathrm{ID}_i, M, \sigma, V) = 1 \,\middle|\, \begin{array}{l} \exists \mathrm{ID}_j \in V, \\ \mathbf{Sign}_{pk,sk_i}(M, V) = \sigma] \end{array}\right] = 1.$$

Consistency. Given a signature σ on message M signed by ID_i, the outputs of all designated verifiers are consistent, that is, for any $\mathrm{ID}_j, \mathrm{ID}_k \in V (j \neq k)$, they yield the same verifcation results with the probability 1,

$$\Pr\left[b = b' \,\middle|\, \begin{array}{l} \mathbf{Verify}_{pk,sk_j}(\mathrm{ID}_i, M, \sigma, V) = b, \\ \mathbf{Verify}_{pk,sk_k}(\mathrm{ID}_i, M, \sigma, V) = b' \end{array}\right] = 1.$$

Unforgeability. For a given signer ID_i and a designated verifier-set $V = \{\mathrm{ID}_j\}_{j=1}^m$ ($\mathrm{ID}_i \notin V$), the successful probability of establishing a valid DVSS signature σ^* on message M^* is negligible, i.e.,

$$\Pr\left[\begin{array}{l} \mathbf{Verify}_{pk,sk_j}(\mathrm{ID}_i, \\ M^*, \sigma^*, V) = 1 \end{array} \middle| \begin{array}{l} (pk, msk) \leftarrow \mathbf{Setup}(\mathbb{S}), \\ sk_i \leftarrow \mathbf{KeyGen}_{msk}(\mathrm{ID}_i), \mathrm{ID}_j \in V, \\ \{sk_k\}_{\mathrm{ID}_k \in V} \leftarrow \mathbf{KeyGen}_{msk}(\mathrm{ID}_k \in V), \\ (M^*, \sigma^*) \leftarrow \mathcal{A}^{\mathbf{Sign}_{pk,sk_i}(\cdot,\cdot)}(pk, \{sk_k\}_{\mathrm{ID}_k \in V}) \end{array}\right] < \epsilon.$$

Exclusivity. For a given signer ID_i, the (valid or invalid) signature σ^* on the chosen message and a designated verifier-set $V = \{\mathrm{ID}_j\}_{j=1}^m$, the successful probability that the signature can be accepted as valid by the designated verifier $\mathrm{ID}_k \notin V$ is negligible even if the adversary $\mathcal{A}$ has the valid private key, that is,

$$\Pr\left[b = b^* \,\middle|\, \begin{array}{l} (pk, msk) \leftarrow \mathbf{Setup}(\mathbb{S}), \\ sk_i \leftarrow \mathbf{KeyGen}_{msk}(\mathrm{ID}_i \in V), \\ (M_0^*, M_1^*) \leftarrow \mathcal{A}^{\mathbf{KeyGen}_{msk}(\mathrm{ID}_k \notin V)}(\mathrm{ID}_i, V), \\ b \leftarrow \{0,1\}, \sigma^* \leftarrow \mathbf{Sign}_{pk,sk_i}(M_b^*, V), \\ b^* \leftarrow \mathcal{A}^{\mathbf{Sign}_{pk,sk_i}(\cdot,\cdot)}(M_1^*, \sigma^*) \end{array}\right] < \frac{1}{2} + \epsilon.$$

The generated signature needs the designated verifier's information derived from the verifier's identity and the corresponding profile information in the public key. The public key, considered as a management tool for a large number of users, consists of the master public key and a user's profile list. Exactly speaking, the joining and revoking operations on the users can be implemented by adding and deleting profiles in the public key. In addition, we can make use of these profiles to provide an on-line search service for querying a specified user.

4 The DVSS scheme

In this section, we present an effective DVSS scheme to meet our design goals. Before describing the scheme, we first propose the concept of identity-based aggregator (IBA) and its two instances, which are core components in the DVSS scheme. The practical DVSS scheme is then presented in the random oracle model.

4.1 Identity-based aggregator

The core technical problem of designing a constant-size and short DVSS signature lies in how to compress a designated set V of any size into an $O(1)$-size representation in cryptography. From the recent researches, the answer of this problem is moving steadily closer to "the truth". For example, a trivial approach was presented by [32] in which each element in the set was mapped into one item in cryptosystem for authenticating membership, i.e., the size of mapping is $O(m)$ for a

[1)] The verifier key-registration protocol of UDVS is to force the verifier to know the secret-key corresponding to his public-key

set with m-size. In [33], the representation of compressed set can be optimized to $O(\sqrt{m})$ size by arranging the elements in $a * b$ array, where $m = ab$. In [7], a lattice-based accumulator is proposed to achieve $O(\log m)$ by using a binary tree structure. This result was heading towards the optimal result $O(1)$. From the above analysis, it is not hard to see that the structure, used to compress the set, is the key to realize an effective representation of set.

For a set of designated verifiers $V = \{\mathrm{ID}_j\}_{j=1}^m \subseteq \mathcal{U}$ with any size m, we present an algebraic curve approach to represent the set V. Ideally, such a representation of set has nothing to do with the size m of the set V, that is, the size of representation of V is $O(1)$. In this approach, we introduce a new cryptographic algorithm, called identity-based aggreagator (IBA), that can compress the public-key information of a set (of any size) into a constant-size value. Exactly, such a aggregator is stated as follows.

Definition 5 Let $\mathcal{PK}$ denote the public key space over a group $\mathbb{G}$. Given a set $\mathcal{U}$, the identity-based aggregator $Aggr : \mathcal{PK} \times 2^{\mathcal{U}} \mapsto \mathbb{G}$ is a deterministic polynomial time algorithm satisfying $Aggr(pk, V) = R_V$, where pk is a public key in $\mathcal{PK}$, a subset $V \subseteq \mathcal{U}$ and R_V is called the representation of set V.

The approach of constructing this algorithm is to build an algebraic curve $f(x)$ using a polynomial through all points that derived from the designated verifier-set. To do it, each element (user's identity ID_j) of V is firstly mapped into one random point in the curve space by using cryptographic hash function, i.e., $x_j = hash(\mathrm{ID}_j)$. And then, all these points $\{x_j\}_{\forall \mathrm{ID}_j \in V}$ are considered as zeros or poles to make up the expected curve $f(x)$. A rational polynomial function has the form $f(x) = \frac{P(x)}{Q(x)}$ that is the quotient of two polynomial $P(x)$ and $Q(x)$. We say the value z is a zero of $f(x)$ if $P(x) = 0$, and z is a pole of $f(x)$ if $Q(x) = 0$. Note that, the number of zeros or poles is unlimited in $f(x)$.

Next, a specific point can be picked out from the curve $f(x)$ as the IBA. This output point must be enough random to guarantee the unpredictability of IBA. We provide a simple way to do this, that is, the system manager chooses a secret γ and the corresponding value $f(\gamma)$ is regarded as the IBA of the set V. The randomness of $f(\gamma)$ can be ensured because of the uniform distribution of hash mapping of each element in V and the randomness of γ. Moreover, this way will be proved enough secure to prevent from forging signature in Section 5.

According to the above approach, two types of IBAs are presented. These two aggregators work on the following cryptographic surrounding. Let $\mathcal{PK}$ be a public key space over a multiplicative cyclic group $\mathbb{G}$ of prime order p and G, H are two generators in $\mathbb{G}$. In addition, define $\gamma \in \mathbb{Z}_p^*$ be an unknown secret and a cryptographic hash function $hash : \{0,1\}^* \mapsto \mathbb{Z}_p^*$, i.e., $x_i = hash(\mathrm{ID}_i)$.

I. Zeros-based aggregator (ZerosAggr) Given a public parameter $pk = \{G_i = G^{\gamma^i}\}_{i=1}^n \in \mathcal{PK}$ for a secret γ and a subset $V = \{\mathrm{ID}_j\}_{j=1}^m \subseteq \mathcal{U}$ $(m < n)$, there exists a polynomial-time algorithm **ZerosAggr** that outputs

$$G_V = \mathbf{ZerosAggr}(pk, V) = G^{\gamma \prod_{\mathrm{ID}_j \in V}(\gamma + x_j)}. \tag{1}$$

How to implement the **ZerosAggr** is shown below. Given all known $\{x_i\}_{i=1}^n$. Define a polynomial $f_V(x) = x \prod_{\mathrm{ID}_j \in V}(x + x_j) = \sum_{k=0}^m a_k x^{k+1} \pmod p$ of degree $m+1$ over V and compute the coefficients $a_k \in \mathbb{Z}_p^*$ for all $k \in [0, m]$. It uses pk to compute $G_V = G^{f_V(\gamma)} = G^{\gamma \prod_{\mathrm{ID}_j \in V}(\gamma + x_j)} = G^{\sum_{k=0}^m a_k \gamma^{k+1}} = \prod_{k=1}^{m+1} G_k^{a_{k-1}}$, where $G_k \in pk$. When $V = \emptyset$, we define $\mathbf{ZerosAggr}(pk, \emptyset) = G_1$.

II. Poles-based aggregator (PolesAggr) Given a public parameter $pk' = \{H_i = H^{\frac{1}{\gamma + x_i}}\}_{i=1}^n \in \mathcal{PK}$ for a secret γ and a subset $V = \{\mathrm{ID}_j\}_{j=1}^m \subseteq \mathcal{U}$ $(m < n)$, there exists a polynomial-time algorithm **PolesAggr** that outputs

$$H_V = \mathbf{PolesAggr}(pk', V) = H^{\prod_{\mathrm{ID}_j \in V} \frac{1}{(\gamma + x_j)}}. \tag{2}$$

A recursive algorithm is provided to implement the **PolesAggr** from $pk' = \{H_i\}_{i \in [1,n]}$ as follows. Define a polynomial $g_V(x) = \frac{1}{\prod_{\mathrm{ID}_j \in V}(x + x_j)} \pmod p$ of degree m over V. Given H_i and H_j, it is easy to obtain the equation $(H_j / H_i)^{\frac{1}{x_i - x_j}} = (H^{\frac{1}{\gamma + x_j}} / H^{\frac{1}{\gamma + x_i}})^{\frac{1}{x_i - x_j}} = H^{\frac{1}{(\gamma + x_i)(\gamma + x_j)}}$, where $x_i \neq x_j$ is a precondition for this equation. Next, this equation is expanded to multi-value cases. Define $B_{s,r} = H^{\prod_{k=s}^r \frac{1}{(\gamma + x_k)}}$, where $1 \leqslant s < r \leqslant m$. In the same way, the equation $B_{s,r+1} = (B_{s,r} / B_{s+1,r+1})^{\frac{1}{x_{r+1} - x_s}}$ can be computed. By using this equation, the output value $H_V = B_{1,m}$ can be completed in a recursive way: $B_{r,r} = H_r$ for $\forall r \in [1, m]$, and $B_{s,r} = (B_{s,r} / B_{s+1,r})^{\frac{1}{x_r - x_s}}$ for $s \in [1, m-1]$, $r \in [2, m]$, where $B_{r,r}$ is the initial input H_r for $r = [1, m]$.

In summary, the above aggregators do not have a limit to the size of the input set. More importantly, we can employ a zero-pole cancellation method to eliminate the same elements between **ZerosAggr** and **PolesAggr**. By using this method, we are able to construct a DVSS to meet our goals.

4.2 Our DVSS construction

A practical scheme for the identity-based DVSS on a bilinear map system $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot, \cdot))$ of prime order p is presented. In cryptosystem, given a full set $\mathcal{U} = \{\mathrm{ID}_i\}_{i=1}^n$ of all

users and a designated verifier-set $V = \{\mathrm{ID}_j\}_{j=1}^{m} \subset \mathcal{U}$, where the size of $\mathcal{U}$ and V are not restricted and $m < n$. The scheme adopts two hash functions: $hash_1 : \{0,1\}^* \to \mathbb{Z}_p^*$ which maps any identity ID described as a binary string to a random value $x_i \in \mathbb{Z}_p^*$ (i.e., $x_i = hash_1(\mathrm{ID}_i)$), and $hash_2 : \{0,1\}^* \to \mathbb{Z}_p^*$ which maps any string to a random value $u \in \mathbb{Z}_p^*$. [2] In order to complete the verification procedure, we use the Zero-Pole Cancellation method which eliminates the same elements in the exponential part between the encapsulated result of **PolesAggr** and the produced result of **ZerosAggr**.

Setup($\mathbb{S}$). This algorithm chooses two random generators G and H in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Let m be the maximum number of aggregated users in the **ZerosAggr**. Then, it randomly picks $\gamma, \varepsilon \leftarrow \mathbb{Z}_p^*$, and defines $R = e(G,H)^{\varepsilon}$, $H' = H^{\gamma}$ and $G_i = G^{\gamma^i}$ for $i \in [1,m]$. It outputs the master secret key $msk = (\gamma, \varepsilon, G)$ and the public key $pk = \{\mathbb{S}, H, H', R, \{G_i\}_{i\in[1,m]}, pp = \emptyset\}$, where pp is a list of users' profiles.

KeyGen$_{msk}(\mathrm{ID}_i)$. Given an $\mathrm{ID}_i \in \mathcal{U}$, the algorithm defines $x_i = hash_1(\mathrm{ID}_i)$ and computes the user's private-key $sk_i = G^{\frac{x_i\varepsilon}{\gamma+x_i}}$ and $H_i = H^{\frac{\varepsilon}{\gamma+x_i}}$. Furthermore, the user's profile $pp_i = (\mathrm{ID}_i, H_i)$ is appended into pp.

Sign$_{pk,sk_i}(M,V)$. The algorithm takes as input pk, a signer's private key sk_i, a message $M \in \{0,1\}^*$, and a set of designated verifiers V. It proceeds the following steps.

1) It invokes **PolesAggr**(pk, V) to produce H_V, picks two random integers λ and s in $\mathbb{Z}_p^*$, then computes the commitments

$$C_1 = H^{\lambda}, \qquad C_2 = sk_i^{\frac{s}{x_i}}, \qquad C_3 = (H_V)^{\lambda},$$

where $H_V = H^{\varepsilon \prod_{\mathrm{ID}_i\in V} \frac{1}{\gamma+x_i}}$ and $C_2 = G^{\frac{s\varepsilon}{\gamma+x_i}}$.

2) It computes a referred value $\bar{R} = R^{\lambda/s} = e(G,H)^{\varepsilon\lambda/s}$.

3) It calculates a challenge value $c = hash_2(C_1, C_2, C_3, M, V, \bar{R}) \in \mathbb{Z}_p^*$.

4) It yields $\mu = \frac{cs}{\lambda} + \frac{1}{s} \pmod p$, and outputs the final signature

$$\sigma \leftarrow (C_1, C_2, C_3, c, \mu).$$

Verify$_{pk,sk_j}(\mathrm{ID}_i, M, \sigma, V)$. The algorithm takes as input pk, the jth verifier's private key sk_j, a signature $\sigma = (C_1, C_2, C_3, c, \mu)$ on a message M for a given signer ID_i, and the set of designated verifiers V. Then it proceeds the following steps.

1) If the relation $\mathrm{ID}_j \in V$ holds, it sets $V_- = V\backslash\{\mathrm{ID}_j\}$, then invokes **ZerosAggr**(pk, V_-) to produce $G_{V_-} = G^{\gamma \prod_{\mathrm{ID}_i\in V_-}(\gamma+x_i)}$. For every designated verifier ID_j, the value $\bar{R'}$ is reconstructed as

$$\bar{R'} = \left(e(sk_j, C_1) \cdot e(G_{V_-}, C_3)\right)^{\mu} \cdot e(C_2, (H' \cdot H^{x_i})^c)^{-1}.$$

2) It retrieves $c' = hash_2(C_1, C_2, C_3, M, V, \bar{R'})$. If $c' = c$, accepts and outputs 1; otherwise, rejects and outputs 0.

Correctness. The correctness of verification procedure is demonstrated as follows.

- Assume the verifier $\mathrm{ID}_j \in V$ holds a private key $sk_j = G^{\frac{x_j\varepsilon}{\gamma+x_j}}$ and the signature $\sigma = (C_1, C_2, C_3, c, \mu)$ produced by the signer ID_i, where $\mu = \frac{cs}{\lambda} + \frac{1}{s} \pmod p$. If the relation $\mathrm{ID}_j \in V$ holds, the value $\bar{R'}$ that is equal to $\bar{R}$ is recovered by the following equation.

$$\begin{aligned}
\bar{R'} &= \left(e(sk_j, C_1) \cdot e(G_{V_-}, C_3)\right)^{\mu} \cdot e(C_2, (H' \cdot H^{x_i})^c)^{-1} \\
&= \frac{\left(e(G^{\frac{x_j\varepsilon}{\gamma+x_j}}, H^{\lambda}) \cdot e(G^{\gamma \prod_{\mathrm{ID}_i\in V_-}(\gamma+x_i)}, (H^{\varepsilon \prod_{\mathrm{ID}_i\in V} \frac{1}{\gamma+x_i}})^{\lambda})\right)^{\mu}}{e(G^{\frac{s\varepsilon}{\gamma+x_i}}, (H^{\gamma} \cdot H^{x_i})^c)} \\
&= \frac{e(G,H)^{\frac{x_j\varepsilon}{\gamma+x_j}\cdot\lambda\cdot\mu} \cdot e(G,H)^{\gamma\cdot\mu \frac{\prod_{\mathrm{ID}_i\in V}(\gamma+x_i)}{\gamma+x_j} \cdot \frac{\varepsilon\cdot\lambda}{\prod_{\mathrm{ID}_i\in V}(\gamma+x_i)}}}{e(G,H)^{\frac{s\varepsilon}{\gamma+x_i}\cdot(-c)(\gamma+x_i)}} \\
&= e(G,H)^{\frac{x_j\varepsilon\lambda\mu}{\gamma+x_j} + \frac{\gamma\varepsilon\lambda\mu}{\gamma+x_j} - cs\varepsilon} = e(G,H)^{\varepsilon\lambda\mu - cs\varepsilon} \\
&= e(G,H)^{\varepsilon\lambda/s} = \bar{R}.
\end{aligned}$$

- The hash value c' is equal to c such that the signature will be accepted.

Consistency. The consistency of the verification results for different verifiers is described. Assume ID_j and ID_k be two different verifiers in V, i.e., $\mathrm{ID}_j, \mathrm{ID}_k \in V$ and $j \neq k$, they hold two private keys sk_j and sk_k, respectively. Let $V'_- = V \setminus \{\mathrm{ID}_j\}$ and $V''_- = V \setminus \{\mathrm{ID}_k\}$. These two verifiers use the public key pk to compute $G_{V'_-} = G^{\gamma \frac{\prod_{\mathrm{ID}_i\in V}(\gamma+x_i)}{(\gamma+x_j)}}$ and $G_{V''_-} = G^{\gamma \frac{\prod_{\mathrm{ID}_i\in V}(\gamma+x_i)}{(\gamma+x_k)}}$, respectively. In terms of the similar analysis in the step 1) of the verification algorithm, the following equation holds.

$$e(sk_j, C_1) \cdot e(G_{V'_-}, C_3) = e(sk_k, C_1) \cdot e(G_{V''_-}, C_3) = e(G,H)^{\varepsilon\lambda}.$$

Meanwhile, the value of $e(C_2, (H' \cdot H^{x_i})^c)^{-1}$ yielded by ID_j is the same as that of ID_k. Therefore, they are able to retrieve the correct $\bar{R}$ and accept the signature.

Performance. In the presented DVSS scheme, the new user can be added into the system, and the total number of users is not restricted. The number of designated verifiers is also not

[2] We like to make such a distinction because two hash functions have the different security properties as discussed later

restricted in a DVSS signature as long as m is big enough. The verifier in the set V is dynamic. Moreover, the size of signature $\sigma = (C_1, C_2, C_3, c, \mu) \in \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_2 \times (\mathbb{Z}_p^*)^2$ is a constant $|\mathbb{G}_1| + 2|\mathbb{G}_2| + 2|\mathbb{Z}_p^*|$. This value is independent on the size of a designated verifier-set, which is an important property for the practical applications. In addition, the user's secret key sk_i is only an element in $\mathbb{G}_1$. In order to improve the performance of scheme, two IBAs can be pre-computed based on the public key pk for a given set V.

5 Security analysis

5.1 Security analysis of unforgeability

Since the forger may accuse the other verifiers in the designated verifier-set V of forging signatures, the presented DVSS does not allow the designated verifiers to produce signatures. A clear unforgeability definition for the DVSS scheme is firstly presented. According to the construction of DVSS, the hash functions are regarded as random oracles controlled by the challenger. The DVSS scheme can be proved to be existential unforgeable against weak chosen message attacks in the random oracle model. Therefore, a game between a challenger and an adversary $\mathcal{A}$ is defined as follows.

Setup For the set $\mathcal{U}$, the challenger takes $\mathbb{S}$ as input and runs **KeyGen** algorithm in the DVSS scheme to obtain a public key pk, a master private key msk and the user's private key sk_i. Then, the challenger gives pk to $\mathcal{A}$.

Learning Before the queries, $\mathcal{A}$ sends a list of l users $\mathscr{U} = \{\mathrm{ID}_k\}_{k=1}^{l}$ to the challenger, which will chosen a challenge identity $\mathrm{ID}_k \in \mathscr{U}$. State that $\mathcal{A}$ can query any message as he/she likes from the signer. And $\mathcal{A}$ performs a polynomially bounded number of queries, each query may depend on the responses of previous queries. Suppose $\mathcal{A}$ is given access at most q_h times to oracle $Hash(\cdot)$ and q_s times to oracle $Sign(\cdot)$, respectively. The types of queries are represented as follows.

1) $hash_1$-Queries. When $\mathcal{A}$ makes a query to the oracle, the challenger assigns the value to $\mathcal{A}$.
2) ExtractKey-Queries. When $\mathcal{A}$ arbitrary chooses a user $\mathrm{ID}_k \in \mathscr{U}$, the challenger runs $\mathbf{KeyGen}_{msk}(\mathrm{ID}_k) = sk_k$ but $\mathrm{ID}_k \neq \mathrm{ID}_i$ according to the DVSS scheme and sends the result to $\mathcal{A}$.
3) $hash_2$-Queries. When $\mathcal{A}$ queries the hash value to the oracle, the challenger outputs the hash value c if the query has existed.
4) Signing-Queries. $\mathcal{A}$ queries a signature on the message M and $V \subseteq \mathscr{U}$. When $\mathcal{A}$ queries a signer ID_i to sign it, the challenger computes the private key sk_i of user ID_i and generates a signature $\sigma_i = \mathbf{Sign}_{pk,sk_i}(M, V)$. Finally, the challenger returns the signature σ_i to $\mathcal{A}$.

Forgery $\mathcal{A}$ returns a forgery (M^*, σ^*) such that σ^* is a valid signature if $\sigma^* \notin \{\sigma_i\}_{i=1}^{l}$, where the signature σ_i is generated during the query phase.

Verify $\mathcal{A}$ sends a valid pair (M^*, σ^*) to the challenger. $\mathcal{A}$ wins the game if $\mathbf{Verify}_{pk,sk_j}(\mathrm{ID}_i, M^*, \sigma^*, V) = 1$, where sk_j is the private key of designated verifier $\mathrm{ID}_j \in V$.

We define the advantage $\mathrm{AdvSig}_{\mathcal{A}}$ of an adversary $\mathcal{A}$ in attacking the signature scheme as the probability that $\mathcal{A}$ wins the above game, where the advantage is taken over the random bits of the challenger and $\mathcal{A}$. A formal definition of existential unforgeability is presented as follows.

Definition 6 (Unforgeability) A forger $\mathcal{A}$ (t, q_h, q_s, ϵ)-weakly breaks a DVSS scheme if $\mathcal{A}$ runs in time at most t, $\mathcal{A}$ makes at most q_h hash queries and q_s signature queries, and $\mathrm{AdvSig}_{\mathcal{A}}$ is at least ϵ. Then, a DVSS scheme is (t, q_h, q_s, ϵ)-existentially unforgeable under a weak chosen message attack if there exists no forger that (t, q_h, q_s, ϵ)-weakly breaks it.

The following theorem shows that the DVSS scheme is existentially unforgeable under weak chosen message attacks, provided that the q-SDH assumption holds in $(\mathbb{G}, \mathbb{H})$.

Theorem 2 (Unforgeability of DVSS scheme) Suppose the (q, t', ϵ)-SDH assumption holds in $(\mathbb{G}, \mathbb{H})$. Then the proposed DVSS scheme is (t, q_h, q_s, ϵ)-secure against existential forgery under a weak chosen-message attack in the random oracle model provided that

$$t \leqslant t' - O(q^2), \quad q_s \leqslant q.$$

The full proof of this theorem is presented in Appendix A based on the q-SDH assumption in the random oracle model.

5.2 Security analysis of exclusivity

In the presented DVSS scheme, a set V of special verifiers is given. Each of the designed verifiers in the set is able to independently verify the validity of a signature. Meanwhile, the verification procedure is invalid for other members out of V even if they have many valid private keys issued by the algorithm manager. Such a property is called *exclusivity*.

A exclusivity definition for the presented DVSS scheme is firstly proposed. To prove that the scheme holds the semantic exclusivity against chosen message attack (Ex-CMA) with learning corrupted users $\mathcal{R} = \{\mathrm{ID}_k\}_{k=1}^{t}$, a game between a challenger and an adversary $\mathcal{A}$ is described as follows.

Setup The challenger takes as input a challenge set $\mathcal{Q} = \{\mathrm{ID}_k\}_{k=1}^{n-t} \subset \mathcal{U}$ chosen by $\mathcal{A}$, runs **Setup** algorithm in the DVSS scheme to obtain a public key pk and a master private key msk. Then, the challenger sends pk to $\mathcal{A}$.

Learning The challenger defines the set of t corrupted users $\mathcal{R}$ such that $\mathcal{R} = \mathcal{U} \setminus \mathcal{Q}$, and sends it to $\mathcal{A}$. $\mathcal{A}$ performs a polynomially bounded number of repeated queries for a user's identity. Also, $\mathcal{A}$ can issue up to t times private key queries and $n - t$ times label queries to obtain the knowledge of this cryptosystem. The types of queries are described as follows.

1) PrivateKey-Queries ($\mathrm{ID}_k \in \mathcal{R}$). When $\mathcal{A}$ makes a query $\mathrm{ID}_k \in \mathcal{R}$, the challenger assigns the hash value $x_k = hash_1(\mathrm{ID}_k)$ to $\mathcal{A}$. Moreover, the challenger runs $\mathbf{KeyGen}_{msk}(\mathrm{ID}_k) = sk_k$ and H_k according to the DVSS, then sends sk_k and (ID_k, H_k) to $\mathcal{A}$.
2) PublicLabel-Queries ($\mathrm{ID}_k \notin \mathcal{R}$). When $\mathcal{A}$ makes a query $\mathrm{ID}_k \notin \mathcal{R}$, the challenger replies $x'_k = hash_1(\mathrm{ID}_k)$ to $\mathcal{A}$. And, the challenger merely runs **KeyGen** algorithm to compute H_k. Finally, the challenger sends (ID_k, H_k) to $\mathcal{A}$.
3) $hash_2$-Queries. When $\mathcal{A}$ queries the hash value, the challenger outputs c if the value has existed.
4) Signing-Queries. $\mathcal{A}$ queries a signature on the message M_i for a designated signer $\mathrm{ID}_i \in \mathcal{Q}$. The challenger runs the signing algorithm to generate the signature $\sigma_i = \mathbf{Sign}_{pk,sk_i}(M_i, V)$. Finally, the challenger returns the pair (M_i, σ_i) to $\mathcal{A}$.

Challenge After learning, $\mathcal{A}$ returns two messages M_0^* and M_1^*. The challenger flips a random coin $b \leftarrow \{0, 1\}$ and generates the signature $\sigma^* = \mathbf{Sign}_{pk,sk_i}(M_b^*, V)$ for the signer $\mathrm{ID}_i \in \mathcal{Q}$ and sends (M_1^*, σ^*) to $\mathcal{A}$.

Learning This is the same as the above learning process.

Guess $\mathcal{A}$ returns a guess $b^* \in \{0, 1\}$ to the challenger. If $b^* = b$, the challenger outputs 1 (True), otherwise, it outputs 0 (False).

We define the advantage of an adversary $\mathcal{A}$ in attacking the signature scheme as the probability that $\mathcal{A}$ wins the above game, where the advantage is taken over the random bits of the challenger and the adversary. The advantage of an adversary $\mathcal{A}$ in this game is defined as $Adv^{\text{Ex-CMA}}_{\text{DVSS},\mathcal{A}}(n,t) = |\Pr[b^* = b] - 1/2|$. A formal definition of exclusivity is introduced as follows.

Definition 7 (Exclusivity) Given a (valid or invalid) signature σ on a message M and a designated verifier-set V, it is computationally infeasible to verify the validity of the signature σ for a verifier $\mathrm{ID}_k \notin V$ even if the verifier ID_k has the valid private key sk_k, that is, the advantage $Adv^{\text{Ex-CMA}}_{\text{DVSS},\mathcal{A}}(n,t)$ that can be accepted as valid is negligible under the above game.

The semantic exclusivity of the DVSS scheme is proved under the assumption of GDHE problem [30, 34]. Let $f(x)$ and $g(y)$ be two known random polynomials of degree t and $n - t$ with pairwise distinct roots, respectively. Define $h(x, y, z) = f^2(x)g(x)yz$ be a three-variable polynomial in a bilinear group system $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$. Based on the above polynomials, a computational problem which is used to prove the semantic security of the presented DVSS scheme is defined as follows.

Theorem 3 ((n,t)-GDHE Problem [35]) Let $\gamma, \lambda, \varepsilon \in \mathbb{G}_p^*$ be three secret random variables and $\hat{G}, \hat{H}$ are two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Define $f(X) = \prod_{i=1}^{t}(X + x_i) = \sum_{i=0}^{t} a_i X^i$ and $g(X) = \prod_{i=1}^{n-t}(X + x'_i) = \sum_{i=0}^{n-t} b_i X^i$ are two polynomicals, where $a_i, b_i \in \mathbb{Z}_p^*$. Given the values in (F_1, F_2, F_3, h)-GDHE problem with

$$\begin{cases} F_1(\gamma,\lambda,\varepsilon) = \langle \hat{G}^{\varepsilon}, \hat{G}^{\gamma\varepsilon}, \ldots, \hat{G}^{\gamma^{t-1}\varepsilon}, \hat{G}^{\gamma f(\gamma)}, \ldots, \hat{G}^{\gamma^m f(\gamma)} \rangle, \\ F_2(\gamma,\lambda,\varepsilon) = \langle \hat{H}^{\varepsilon}, \hat{H}^{\gamma\varepsilon}, \ldots, \hat{H}^{\gamma^n\varepsilon}, \hat{H}^{f(\gamma)g(\gamma)}, \hat{H}^{\lambda f(\gamma)g(\gamma)}, \hat{H}^{\lambda\varepsilon f(\gamma)} \rangle, \\ F_3(\gamma,\lambda,\varepsilon) = e(\hat{G},\hat{H})^{\varepsilon f^2(\gamma)g(\gamma)}, \end{cases}$$

$h(\gamma, \lambda, \varepsilon) = \lambda\varepsilon f^2(\gamma)g(\gamma)$, and a random value $T \leftarrow \mathbb{G}_T$, decide whether or not $T = e(\hat{G}, \hat{H})^{h(\gamma,\lambda,\varepsilon)}$. For any algorithm $\mathcal{A}$ whicht makes a total of at most q queries to the oracle computing the group operation in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and the bilinear pairing, the advantage of $\mathcal{A}$ is $Adv_{\text{GDHE},\mathcal{A}}(n,t) \leqslant \frac{[q+2(n+t+m+4)+2]^2 \cdot 2n}{2p}$.

We provide the proof of this theorem in Appendix B of [35]. Based on (n,t)-GDHE problem, the exclusivity of the presented DVSS scheme satisfies the following theorem.

Theorem 4 (Exclusivity of DVSS scheme) Suppose the (n,t)-GDHE problem is hard in $\mathbb{S}$, then the DVSS scheme is semantically exclusive against chosen-message attack (Ex-CMA) with colluders.

The proof of this theorem is provided in Appendix B based on the GDHE assumption. Additionally, due to the number of corrupted users t is not restricted, the DVSS scheme is secure for arbitrary large collusion of corrupted users.

6 Conclusion

In view of the requirement that the generated signature can be verified by many parties, we present the first designated verifier-set signature with short and constant length for arbitrary dynamic verifiers from a large number of users. We firstly propose two identity-based aggregators that compress a designated set of verifier's identities to a constant-size random string in cryptographic space. Our main contribution is to propose a DVSS scheme constructed from the zero-pole cancellation method which can eliminate the same elements between zeros-based aggregator and poles-based aggregator. Simultaneously, we prove that the DVSS scheme is secure into the sense of unforgeability and exclusivity, respectively. For future work, we will improve the performance and apply our scheme into some practical applications, e.g., broadcasting, keyword searching, and voting.

Acknowledgements The work was supported by the National Key Technologies R&D Programs of China (2018YFB1402702 and 2017YFB0802500), the "13th" Five-Year National Cryptographic Development Foundation (MMJJ20180208), NSFC-Genertec Joint Fund For Basic Research (U1636104), and the National Natural Science Foundation of China (Grant Nos. 61572132, 61972032 and U1705264).

Appendixes

Appendix A. Proof of Theorem 2

Proof. Assume that $\mathcal{A}$ (t, q_h, q_s, ϵ)-weakly breaks the signature scheme, then there exists a polynomial time algorithm $\mathcal{B}$ which solves the q-SDH problem in time t' with non-negligible probability ϵ by interacting with $\mathcal{A}$ and using the forking lemma. Before the simulator $\mathcal{B}$ is executed, we suppose $\mathcal{A}$ sends a list of l users $\mathscr{U} = \{\mathrm{ID}_i\}_{i=1}^{l}$ ($l < q$, we assume q is large enough) to $\mathcal{B}$. Meanwhile, suppose the algorithm $\mathcal{B}$ takes as input a known random sequence $\langle g, g^{\gamma}, g^{\gamma^2}, \ldots, g^{\gamma^q}\rangle \in \mathbb{G}_1$, and expects to output $\langle \zeta, g^{\frac{1}{\gamma+\zeta}}\rangle$, where γ is unknown and $\zeta \in \mathbb{Z}_p^*$. $\mathcal{B}$ does so by interacting with the forger $\mathcal{A}$ as follows.

Setup $\mathcal{B}$ randomly chooses l values $\{x_i\}_{i=1}^{l}$ in $\mathbb{Z}_p^*$, which are stored as a part of public key. Also, $\mathcal{B}$ picks two random elements $\varepsilon, \lambda, \leftarrow \mathbb{Z}_p^*$, and constructs a polynomial: $f(x) = \prod_{i=1}^{l}(x + x_i) = \sum_{i=0}^{l} a_i x^i$, where the coefficient a_i can be calculated from $\{x_i\}_{i=1}^{l}$. Moreover, $\mathcal{B}$ defines

$$G = g^{f(\gamma)} = \prod_{k=0}^{l} g^{\gamma^k a_k}, H = h^{f(\gamma)} = (\varphi(g))^{f(\gamma)} = \varphi(G),$$
$$H' = H^{\gamma} = \varphi(\prod_{k=0}^{l} g^{\gamma^{k+1} a_k}), \quad R = e(G, H)^{\varepsilon},$$
$$G_i = G^{\gamma^i} = \prod_{k=0}^{l} g^{a_k \gamma^{i+k}}, H_i = H^{\frac{\varepsilon}{\gamma+x_i}} = \varphi(\prod_{k=0}^{l-1} g^{\gamma^k b_k \varepsilon}),$$

where the equation $l + m \leqslant q$ must be satisfied if $\{G_i\}_{i=1}^{m}$ is constructed, and the user's public profiles $pp = \{(x_i, H_i)_{i=1}^{l}\}$ is assigned by using $f_i'(x) = \frac{\prod_{k=1}^{l}(x+x_k)}{x+x_i} = \sum_{k=0}^{l-1} b_k x^k$ if $x_i \in \{x_k\}_{k=1}^{l}$. Then, $\mathcal{B}$ gives the public key $pk_i = \{H, H', R, \{G_i\}_{i=1}^{m}, \{(x_i, H_i)_{i=1}^{l}\}\}$ to $\mathcal{A}$. Since the relation $(\gamma + x_i)|f(\gamma)$ holds, the value $g^{\frac{f(\gamma)}{\gamma+x_i}} = \prod_{k=0}^{l-1} g^{d_k \gamma^k}$ can be computed, where the coefficient d_i can be calculated and $g, g^{\gamma}, \ldots, g^{\gamma^l}$ is known.

Learning The types of queries allowed in our DVSS are described as follows.

1) $hash_1$-Queries. When $\mathcal{A}$ queries $\mathrm{ID}_i \in \mathscr{U}$, $\mathcal{B}$ directly assigns $x_i = hash_1(\mathrm{ID}_i)$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ selects a random element $\omega_i \in \mathbb{Z}_p^*$ and stores ω_i in case the same query is made again.
2) ExtractKey-Queries. When $\mathcal{A}$ makes a query $\mathrm{ID}_i \in \mathscr{U}$, the algorithm $\mathcal{B}$ computes $\mathbf{KeyGen}_{msk}(\mathrm{ID}_i) = sk_i = G^{\frac{x_i \varepsilon}{\gamma+x_i}}$ according to the DVSS scheme. Then, $\mathcal{B}$ gives sk_i to $\mathcal{A}$. Otherwise, $\mathcal{B}$ does not respond this query.
3) $hash_2$-Queries. When $\mathcal{A}$ queries the hash value on $(C_1, C_2, C_3, M, V, \bar{R})$, $\mathcal{B}$ directly outputs c if this query has already existed. Otherwise, $\mathcal{B}$ sends a random value $c \in \mathbb{Z}_p^*$ to $\mathcal{A}$ as the hash value, and stores the result c and $(C_1, C_2, C_3, M, V, \bar{R})$.
4) Signing-Queries. When $\mathcal{A}$ queries a signature on the message M and the set of designated verifiers V, $\mathcal{B}$ responds as follows. If $\mathcal{A}$ assigns the signer $\mathrm{ID}_i \in \mathcal{U}$ to sign the message M, $\mathcal{B}$ uses the isomorphism φ to compute

$$C_1^* = H^{\lambda} = \varphi[(g^{d_0} g^{d_1 \gamma} \cdots g^{d_l \gamma^l})^{\lambda}] = \varphi(\prod_{k=0}^{l} g^{d_k \gamma^k \lambda}).$$

Also, $\mathcal{B}$ obtains the private key sk_i of the user ID_i (as the ExtractKey-Queries described), then $\mathcal{B}$ chooses the random value $s \in \mathbb{Z}_p^*$ and computes

$$C_2^* = sk_i^{\frac{s}{x_i}} = G^{\frac{s\varepsilon}{\gamma+x_i}} = g^{\frac{f(\gamma)s\varepsilon}{\gamma+x_i}} = \prod_{k=0}^{l-1} g^{d_k \gamma^k s \varepsilon},$$

due to $(\gamma + x_i)|f(\gamma)$ is a computational polynomial with $l-1$ degrees. $\mathcal{B}$ invokes $\mathbf{PolesAggr}(pk, V)$ to produce $H_V = H^{\varepsilon \prod_{\mathrm{ID}_i \in V} \frac{1}{\gamma+x_i}}$ and uses the isomor-

phism φ to compute

$$C_3^* = (H_V)^\lambda = \varphi[(g^{d_0} g^{d_1\gamma} \cdots g^{d_{l-m}\gamma^{l-m}})^{\varepsilon\lambda}] = \varphi(\prod_{k=0}^{l-m} g^{d_k\gamma^k\varepsilon}).$$

Moreover, both c^* and $\bar{R}$ are obtained by running the $hash_2$-queries process. Always, $\mathcal{B}$ can outputs the random value $\mu^* \in \mathbb{Z}_p^*$. Then, $\mathcal{B}$ simulates the signing algorithm $\mathbf{Sign}_{pk,sk_i}(M, V)$ to obtain a signature $\sigma \leftarrow (C_1^*, C_2^*, C_3^*, c^*, \mu^*)$ and returns the signature σ to $\mathcal{A}$.

Forgery After the learning process, $\mathcal{A}$ returns a forgery (M^*, σ^*) such that σ^* is a valid signature, where the signer $\mathrm{ID}^* \notin \mathscr{U}$.

Verify $\mathcal{A}$ sends a valid pair (M^*, σ^*) to $\mathcal{B}$. If $\mathbf{Verify}_{pk,sk_j}(M^*, \sigma^*, V) = 1$, $\mathcal{A}$ wins the game.

Now we analyze the validate of $\mathcal{B}$ as follows. Firstly, according to $R = e(G,H)^\varepsilon$, $C_1^* = H^\lambda$ and $V^* \subseteq \mathscr{U}$, it invokes $\mathbf{PolesAggr}(pk, V^*)$ to produces $H_{V^*} = H^{\varepsilon\prod_{\mathrm{ID}_i\in V^*}\frac{1}{(\gamma+x_i)}}$. If $\mathrm{ID}_j \in V^*$ and $V_-^* = V^* \setminus \{\mathrm{ID}_j\}$, then it invokes $\mathbf{ZerosAggr}(pk, V_-^*)$ to produces $G_{V_-^*} = G^{\gamma\prod_{\mathrm{ID}_i\in V_-^*}(\gamma+x_i)}$, we have the following equation holds, that is,

$$\frac{R^\lambda}{e(G_{V_-^*}, C_3^*)} = \frac{R^\lambda}{e(G_{V_-^*}, (H_{V^*})^\lambda)} = e(G,H)^{\lambda\varepsilon}/e(G,H)^{\gamma\varepsilon\lambda\frac{1}{\gamma+x_j}} = e(G,H)^{\varepsilon\lambda\frac{x_j}{\gamma+x_j}} = e(sk_j, H^\lambda) = e(sk_j, C_1^*). \quad (3)$$

It demonstrates $sk_j = G^{\frac{x_j\varepsilon}{\gamma+x_j}}$ is a valid secret key of ID_j.

If the challenge signature (M^*, δ^*) is valid, it will pass the verification algorithm process. That is, according to the $hash_2$ query $c^* = hash_2(C_1^*, C_2^*, C_3^*, M^*, V^*, \bar{R})$, the step 2) of **Verify** algorithm will be successful. Next, considering the step 1) of **Verify**, we have

$$\bar{R} = \left(e(sk_j, C_1^*) \cdot e(G_{V_-^*}, C_3^*)\right)^{\mu^*} \cdot e(C_2^*, (H' \cdot H^{x_i})^{c^*})^{-1} = R^{\lambda\mu^*} \cdot e(C_2^*, (H' \cdot H^{x_i})^{c^*})^{-1}.$$

Therefore, $\mathcal{B}$ can determine $C_2^* = G^{\frac{s\varepsilon}{\gamma+x_i}}$, however, $\mathcal{B}$ cannot figure it out since the value s is unknown.

Then, according to the Forking Lemma [31], $\mathcal{B}$ computes s by the following process. There exists two probabilistic polynomial time Turing machine $\mathcal{A}_1$ and $\mathcal{A}_2$. $\mathcal{A}_1$ produces a commit $\sigma^* = (C_1^*, C_2^*, C_3^*)$ on the input message M^* that is from $\mathcal{A}_2$, then it sends the result to $\mathcal{A}_2$. $\mathcal{A}_2$ gives the hash $c^* = hash_2(C_1^*, C_2^*, C_3^*, M^*, V, \bar{R})$ to $\mathcal{A}_1$. At last, $\mathcal{A}_2$ receives the response σ'^* from $\mathcal{A}_1$, where σ_2 just depends on (M^*, σ^*). The idea here is to think of $\mathcal{A}_1$ and $\mathcal{A}_2$ as running two times in related executions, then produces a branching point. The adversary forges twice on the same message but with different random oracle outputs c^*, c'^*(that is, with $c^* \neq c'^*$) since both c^* and c'^* are chosen randomly, then the adversary obtains two good forgeries μ^*, μ'^*. In our DVSS scheme, $\mathcal{B}$ can achieve two valid signatures $(C_1^*, C_2^*, C_3^*, c^*, \mu^*)$ and $(C_1^*, C_2^*, C_3^*, c'^*, \mu'^*)$ such that $\mu^* = c^* s/\lambda + 1/s \pmod p$, and $\mu'^* = c'^* s/\lambda + 1/s \pmod p$. Since we can get c^*, c'^* from the different oracle query, we can always find c^*, c'^* such that $\gcd(c^* - c'^*, p) = 1$. Therefore, we have $s = (\mu^* - \mu'^*)(c^* - c'^*)^{-1}\lambda \pmod p$. Thus, $\mathcal{B}$ obtains the pair $\langle x_i, C_2^*\rangle = \langle x_i, G^{\frac{s\varepsilon}{\gamma+x_i}}\rangle$.

Finally, $\mathcal{B}$ begins to solve the q-SDH problem. Since $\mathrm{ID}^* \notin \mathscr{U}$, we have $(\gamma + x_i) \nmid f(\gamma)$, then we obtain the polynomial

$$f(\gamma) = \sum_{i=0}^{l} a_i\gamma^i = (\sum_{i=0}^{l-1} a_i'\gamma^i)(\gamma + x_i) + r_c \pmod p,$$

where, a_i' and r_c can be computed from $\{x_1, x_2, \ldots, x_l\}$. Therefore, $\mathcal{B}$ converts $\langle x_i, G^{\frac{s\varepsilon}{\gamma+x_i}}\rangle$ into the presentation of $\langle\zeta, g^{\frac{1}{\gamma+\zeta}}\rangle(\zeta \in \mathbb{Z}_p^*)$ as follows. Let $\zeta = x_i$. And the equation $C_2^* = G^{\frac{s\varepsilon}{\gamma+x_i}} = g^{\frac{s\varepsilon}{\gamma+x_i}\cdot f(\gamma)} = g^{\varepsilon s\cdot(\sum_{i=0}^{l-2} a_i'\gamma^i + \frac{r_c}{\gamma+x_i})} = (\prod_{i=0}^{l-2} g^{\gamma^i a_i'})^{\varepsilon s} \cdot g^{\varepsilon s\frac{r_c}{\gamma+\zeta}}$ holds. Therefore, $\mathcal{B}$ computes

$$g^{\frac{1}{\gamma+\zeta}} = (C_2^*)^{\frac{1}{\varepsilon s r_c}}/(\prod_{i=0}^{l-2}(g^{\gamma^i})^{a_i'})^{r_c}.$$

$\mathcal{B}$ outputs $\langle\zeta, g^{\frac{1}{\gamma+\zeta}}\rangle$, that is, $\mathcal{B}$ has the non-negligible advantage ϵ in solving the q-SDH problem in time t', which contradicts with q-SDH assumption. The claimed bound $t \leqslant t' - O(q^2)$ is obvious by the construction of the algorithm $\mathcal{B}$. Therefore, we complete the proof.

Appendix B. Proof of Theorem 4

Proof. Suppose an adversary $\mathcal{A}$ can break our scheme with the advantage $Adv^{\text{Ex-CMA}}_{\text{DVSS},\mathcal{A}}(n,t)$, then there exists a polynomical time algorithm $\mathcal{B}$ which solves the (n,t)-GDHE problem by using the advantage of $\mathcal{A}$. $\mathcal{A}$ randomly chooses a challenge subset $\mathcal{Q} \subset \mathcal{U}$ such that $\mathcal{R} = \mathcal{U} \setminus \mathcal{Q}$, and sends them to $\mathcal{B}$. The full proof is presented as follows.

Setup The algorithm takes as input a challenge set $\mathcal{Q} = \{\mathrm{ID}_i\}_{i=1}^{n-t} \subset \mathcal{U}$ and an (n,t)-GDHE instance, where $|\mathcal{Q}| = n - t$. From the instance, γ, λ and ε are unknown, but random integers $x_i, a_i \in \mathbb{Z}_p^*$ in $f(x)$ and $x_i', b_i \in \mathbb{Z}_p^*$ in $g(x)$ are known. Besides, any pairwise (x_i, x_i') are not equal to each other. Set $G = \hat{G}^{f(\gamma)}$ (unknown) and $H = \hat{H}^{f(\gamma)g(\gamma)}$. And, we compute H, R, G_k from the (n,t)-GDHE instance. Above all, it outputs a part of

the public parameter pk,

$$pk = \begin{cases} H = \hat{H}^{f(\gamma)g(\gamma)}, \\ R = e(G,H)^{\varepsilon} = e(\hat{G},\hat{H})^{\varepsilon f^2(\gamma)g(\gamma)}, \\ G_k = G^{\gamma^k} = \hat{G}^{\gamma^k f(\gamma)}, \ \ k = [1,m]. \end{cases} \tag{4}$$

Then, $\mathcal{B}$ sends the $pk = \{H, R, \{G_k\}_{k=1}^m\}$ to the $\mathcal{A}$.

Learning The algorithm defines $\mathcal{R} = \{\mathrm{ID}_i\}_{i=1}^t$, which indicates at most t corrupted users. The types of queries are described as follows.

1) PrivateKey-Queries ($\mathrm{ID}_i \in \mathcal{R}$). $\mathcal{B}$ randomly chooses an user $\mathrm{ID}_i \in \mathcal{R}$, computes $x_i = hash_1(\mathrm{ID}_i)$. For $i \in [1,t]$, $\mathcal{B}$ defines the polynomials $f_i(X) = \frac{f(X)}{X+x_i} = \prod_{k=1,k\neq i}^{t}(X+x_k) = \sum_{k=0}^{t-1} a'_k X^k$, and $f_i(X)g(X) = \sum_{k=0}^{n-1} b'_k X^k$, where a'_k and b'_k are known integers. Based on the known values $(\hat{G}^{\varepsilon}, \hat{G}^{\gamma\varepsilon}, \ldots, \hat{G}^{\gamma^{t-1}\varepsilon})$, $\mathcal{B}$ generates the private key of the corrupted users as $sk_i = G^{\frac{x_i\varepsilon}{\gamma+x_i}} = \hat{G}^{x_i\varepsilon f_i(\gamma)} = \hat{G}^{x_i\varepsilon\sum_{k=0}^{t-1} a'_k\gamma^k} = \prod_{k=0}^{t-1}(\hat{G}^{\gamma^k\varepsilon})^{a'_k x_i}$. Furthermore, $\mathcal{B}$ uses the known values $(\hat{H}^{\varepsilon}, \hat{H}^{\gamma\varepsilon}, \ldots, \hat{H}^{\gamma^n\varepsilon})$ to compute $H_i = H^{\frac{\varepsilon}{\gamma+x_i}} = \hat{H}^{\varepsilon f_i(\gamma)g(\gamma)} = \prod_{k=0}^{n-1}(\hat{H}^{\gamma^k\varepsilon})^{b'_k}$. Finally, $\mathcal{B}$ sends sk_i and $pp_i = (\mathrm{ID}_i, H_i)$ to $\mathcal{A}$.

2) PublicLabel-Queries ($\mathrm{ID}_i \notin \mathcal{R}$). $\mathcal{B}$ chooses randomly a user $\mathrm{ID}_i \notin \mathcal{R}$, then $\mathcal{B}$ replies $x'_i = hash_1(\mathrm{ID}_i)$ without the private key. For $i \in [1, n-t]$, $\mathcal{B}$ defines the polynomials $g_i(X) = \frac{g(X)}{X+x'_i} = \prod_{k=1,k\neq i}^{n-t}(X+x'_k) = \sum_{k=0}^{n-t-1} d_k X^k$, $f(X)g_i(X) = \sum_{k=0}^{n-1} d'_k X^k$, where d_k and d'_k are known integers. By using the above-mentioned approach, $\mathcal{B}$ computes $H_i = H^{\frac{\varepsilon}{\gamma+x'_i}} = \hat{H}^{\varepsilon f(\gamma)g_i(\gamma)} = \prod_{k=0}^{n-1}(\hat{H}^{\gamma^k\varepsilon})^{d'_k}$ in terms of $(\hat{H}^{\varepsilon}, \hat{H}^{\gamma\varepsilon}, \ldots, \hat{H}^{\gamma^n\varepsilon})$. Finally, $\mathcal{B}$ sends $pp_i = (\mathrm{ID}_i, H_i)$ to $\mathcal{A}$.

3) $hash_2$-Queries. When $\mathcal{A}$ queries the hash value on $(C_1, C_2, C_3, M, V, \bar{R})$, $\mathcal{B}$ directly outputs c if this query has already existed. Otherwise, $\mathcal{B}$ sends a random value $c \in \mathbb{Z}_p^*$ to $\mathcal{A}$ as the hash value, and stores the result c and $(C_1, C_2, C_3, M, V, \bar{R})$.

4) Signing-Queries. $\mathcal{A}$ queries a signature on the message M_i for a signer ID_{i^*} and the set of designated verifiers V. $\mathcal{B}$ generates the signature $\sigma_i = Sign_{pk,sk_{i^*}}(M_i, V)$ and $V \subset \mathcal{Q}$. Finally, the challenger returns the pair (M_i, σ_i) to $\mathcal{A}$.

Challenge After learning, $\mathcal{A}$ returns two messages M_0^* and M_1^*, as well as a verifier-set $V^* \subset \mathcal{Q}$. $\mathcal{B}$ flips a random coin $b \leftarrow \{0,1\}$, and chooses $\lambda \in \mathbb{Z}_p^*$ from the (n,t)-GDHE instance to compute $C_1^* = H^{\lambda} = \hat{H}^{\lambda f(\gamma)g(\gamma)}$. Next, in terms of the private key sk_{i^*} of the corrupted users (as the PrivateKey-Queries ($\mathrm{ID}_{i^*} \in \mathcal{R}$) described), $\mathcal{B}$ chooses a random value $s \in \mathbb{Z}_p^*$ and computes

$$C_2^* = sk_{i^*}^{\frac{s}{x_{i^*}}} = (\hat{G}^{x_i\varepsilon\sum_{k=0}^{t-1} a'_k\gamma^k})^{\frac{s}{x_{i^*}}} = \prod_{k=0}^{t-1}(\hat{G}^{\gamma^k\varepsilon})^{a'_k s}.$$

Furthermore, $\mathcal{B}$ invokes $\mathbf{PolesAggr}(pk, V^*)$ to produce $H_{V^*} = H^{\varepsilon\prod_{\mathrm{ID}_j\in V^*}\frac{1}{\gamma+x_j}}$, and computes $C_3^* = (H_{V^*})^{\lambda} = \hat{H}^{\varepsilon\lambda f(\gamma)\cdot\frac{g(\gamma)}{\prod_{\mathrm{ID}_j\in V^*}\gamma+x_j}} = \hat{H}^{\varepsilon\lambda f(\gamma)\cdot\prod_{\mathrm{ID}_j\in\mathcal{Q}\setminus V^*}\gamma+x_j}$. Then, $\mathcal{B}$ chooses two random integer $c^*, \mu^* \in \mathbb{Z}_p^*$, and simulates the algorithm $\mathbf{Sign}_{pk,sk_{i^*}}(M_b^*, V^*)$ to obtain a signature $\sigma^* \leftarrow (C_1^*, C_2^*, C_3^*, c^*, \mu^*)$. Finally, $\mathcal{B}$ sends (M_0^*, M_1^*, σ^*) to $\mathcal{A}$.

$hash_2$-Queries. When $\mathcal{A}$ queries the hash value on $(C_1, C_2, C_3, M, V, \bar{R})$, $\mathcal{B}$ directly outputs c if this query has already existed. Otherwise, $\mathcal{B}$ outputs $c^* \leftarrow hash_2(C_1, C_2, C_3, M, V, \bar{R})$ if the query satisfies the conditions: $C_1 = C_1^*$, $C_2 = C_2^*$, $C_3 = C_3^*$, $M = M_b^*$, $V = V^*$, and $\bar{R} = T^{\mu^*} \cdot e(C_2^*, (H' \cdot H^{x_{i^*}})^{c^*})^{-1}$. $\mathcal{B}$ sends a random value $c^* \in \mathbb{Z}_p^*$ to $\mathcal{A}$ as the hash value, and stores them.

Guess $\mathcal{A}$ returns a guess $b^* \in \{0,1\}$ to $\mathcal{B}$. If $b^* = b$, $\mathcal{B}$ outputs 1 (True), otherwise, it outputs 0 (False).

Now we analyze the validate of $\mathcal{B}$ as follows. First of all, if the given value T is valid, the challenge signature (M_b^*, δ^*) will be valid. According to $G_{V_-^*} = \hat{G}^{\gamma f(\gamma)\prod_{\mathrm{ID}_i\in V_-^*}(\gamma+x_i)}$, $V_-^* = V^* \setminus \{\mathrm{ID}_j\}$ and $\mathrm{ID}_j \in V^* \subseteq \mathcal{Q}$, we have the relation

$$\begin{aligned} e(G_{V_-^*}, C_3^*) &= e(\hat{G}^{\gamma f(\gamma)\prod_{ID_i\in V_-^*}(\gamma+x_i)}, \hat{H}^{\varepsilon\lambda f(\gamma)\cdot\frac{g(\gamma)}{\prod_{\mathrm{ID}_i\in V^*}\gamma+x_i}}) \\ &= e(\hat{G},\hat{H})^{\gamma\varepsilon\lambda\frac{f^2(\gamma)g(\gamma)}{\gamma+x_j}}. \end{aligned}$$

If T is valid, we have $T = e(\hat{G},\hat{H})^{\lambda\varepsilon f^2(\gamma)g(\gamma)}$ and $C_1^* = \hat{H}^{\lambda f(\gamma)g(\gamma)}$, such that the following equation holds, i.e.,

$$\begin{aligned} \frac{T}{e(G_{V_-^*}, C_3^*)} &= e(\hat{G},\hat{H})^{\lambda\varepsilon f^2(\gamma)g(\gamma)} / e(\hat{G},\hat{H})^{\gamma\varepsilon\lambda\frac{f^2(\gamma)g(\gamma)}{\gamma+x_j}} \\ &= e(\hat{G},\hat{H})^{\varepsilon\lambda f^2(\gamma)g(\gamma)\frac{x_j}{\gamma+x_j}} = e(sk_j, \hat{H}^{\lambda f(\gamma)g(\gamma)}) \\ &= e(sk_j, C_1^*). \end{aligned} \tag{5}$$

This means that $sk_j = \hat{G}^{\varepsilon f(\gamma)\frac{x_j}{\gamma+x_j}} = G^{\frac{x_j\varepsilon}{\gamma+x_j}}$ is a valid secret key of ID_j. However, this value could be computed from GDHE problem because of $\mathrm{ID}_j \notin \mathcal{R}$ and $(\gamma + x_j) \nmid f(\gamma)$. In terms of Eq. (5), we know that the equation

$$\begin{aligned} T &= e(\hat{G},\hat{H})^{\lambda\varepsilon f^2(\gamma)g(\gamma)} = e(\hat{G}^{\lambda f(\gamma)}, \hat{H}^{\varepsilon f(\gamma)g(\gamma)}) \\ &= e(sk_j, C_1^*) \cdot e(G_{V_-^*}, C_3^*), \end{aligned}$$

holds if the given value T is valid.

Next, considering the step 1) of **Verify** algorithm, we have

$$\begin{aligned}\bar{R} &= \left(e(sk_j, C_1^*) \cdot e(G_{V_-^*}, C_3^*)\right)^{\mu^*} \cdot e(C_2^*, (H' \cdot H^{x_i})^{c^*})^{-1} \\ &= T^{\mu^*} \cdot e(C_2^*, (H' \cdot H^{x_i})^{c^*})^{-1}.\end{aligned}$$

Further, according to the $hash_2$-queries $c^* \leftarrow hash_2(C_1^*, C_2^*, C_3^*, M_b^*, V^*, \bar{R})$ after **Challenge**, the step 2) of **Verify** algorithm will be successful, that is, the challenge signature (M_b^*, δ^*) will be valid.

Secondly, we analyze the advantage of the simulator $\mathcal{B}$. Considering that b is chosen randomly, we have $\Pr[b = 1] = \Pr[b = 0] = \frac{1}{2}$ and obtain the success probability of $\mathcal{B}$ under the condition $Cdt : \bar{R} = T^{\mu^*} \cdot e(C_2^*, (H' \cdot H^{x_{i^*}})^{c^*})^{-1}$ as follows.

$$\begin{aligned}\Pr[b = \mathcal{A}(M_0^*, M_1^*, \sigma^*)|Cdt] &= \Pr[b = b^*|Cdt] \\ &= \Pr[b^* = 1|b = 1 \wedge Cdt] \Pr[b = 1] \\ &+ \Pr[b^* = 0|b = 0 \wedge Cdt] \Pr[b = 0] \\ &= \frac{1}{2} \Pr[b^* = 1|b = 1 \wedge Cdt] \\ &+ \frac{1}{2} \Pr[b^* = 0|b = 0 \wedge Cdt].\end{aligned}$$

Moreover, $\mathcal{A}$ picks a random bit b with $\frac{1}{2}$ possibility, such that $\Pr[b = b^*|T] = \Pr[b \neq b^*|T] = \frac{1}{2}$. Above all, we have

$$\begin{aligned}&Adv_{\mathrm{GDHE},\mathcal{B}}(n,t) = |Pr[b = b^*|Cdt] - Pr[b = b^*|T]| \\ &= \left|\frac{1}{2} \Pr[b^* = 1|b = 1 \wedge Cdt] + \frac{1}{2} \Pr[b^* = 0|b = 0 \wedge Cdt] - \frac{1}{2}\right| \\ &= \frac{1}{2} \left|\Pr[b^* = 1|b = 1 \wedge Cdt] - \Pr[b^* = 1|b = 0 \wedge Cdt]\right|.\end{aligned}$$

In terms of the advantage $Adv_{\mathrm{DVSS},\mathcal{A}}^{\text{Ex-CMA}}(n,t)$ that $\mathcal{A}$ wins this game, we have the following equation under the precondition of Cdt.

$$\begin{aligned}&Adv_{\mathrm{DVSS},\mathcal{A}}^{\text{Ex-CMA}}(n,t) = |Pr[b^* = b] - 1/2| \\ &= \left|\frac{1}{2}(\Pr[b^* = 1|b = 1] + \Pr[b^* = 0|b = 0] - 1)\right| \\ &= \frac{1}{2} \left|\Pr[b^* = 1|b = 1] - \Pr[b^* = 1|b = 0]\right|.\end{aligned}$$

Then, we obtain that the equation $Adv_{\mathrm{GDHE},\mathcal{B}}(n,t) = Adv_{\mathrm{DVSS},\mathcal{A}}^{\text{Ex-CMA}}(n,t)$ holds. According to Theorem 3, we have $Adv_{\mathrm{DVSS},\mathcal{A}}^{\text{Ex-CMA}}(n,t) \leqslant \frac{[q+2(n+t+m+4)+2]^2 \cdot 2n}{2p}$. This implies the adversary $\mathcal{A}$ solves (n,t)-GDHE problem with non-negligible probability ϵ. However, the (n,t)-GDHE problem is hard in $\mathbb{S}$, the conclusion would contradict with the assumption. Therefore, we complete the proof, that is, we proof Theorem 4.

References

1. Jakobsson M, Sako K, Impagliazzo R. Designated verifier proofs and their applications. In: Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques. 1996, 143–154
2. Steinfeld R, Bull L, Wang H, Pieprzyk J. Universal designated-verifier signatures. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security. 2003, 523–542
3. Saeednia S, Kremer S, Markowitch O. An efficient strong designated verifier signature scheme. In: Proceedings of International Conference on Information Security and Cryptology. 2003, 40–54
4. Ng C Y, Susilo W, Mu Y. Universal designated multi verifier signature schemes. In: Proceedings of the 11th International Conference on Parallel and Distributed Systems. 2005, 305–309
5. Shailaja G, Kumar K P, Saxena A. Universal designated multi verifier signature without random oracles. In: Proceeding of the 9th International Conference on Information Technology. 2006, 168–171
6. Chang T Y. An ID-based multi-signer universal designated multi-verifier signature scheme. Information and Computation, 2011, 209(7): 1007–1015
7. Libert B, Ling S, Nguyen K, Wang H. Zero-knowledge arguments for lattice-based accumulators: logarithmic-size ring signatures and group signatures without trapdoors. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques. 2016, 1–31
8. Steinfeld R, Wang H, Pieprzyk J. Efficient extension of standard Schnorr/RSA signatures into universal designated-verifier signatures. In: Proceedings of International Workshop on Public Key Cryptography. 2004, 86–100
9. Kang B, Boyd C, Dawson E. Identity-based strong designated verifier signature schemes: attacks and new construction. Computers and Electrical Engineering, 2009, 35(1): 49–53
10. Khan A U, Ratha B K, Mohanty S. A timestamp-based strong designated verifier signature scheme for next-generation network security services. In: Bhatia S, Mishra K, Tiwari S, Singh V, eds. Advances in Computer and Computational Sciences. Springer, Singapore, 2017, 311–320
11. Susilo W, Zhang F, Mu Y. Identity-based strong designated verifier signature schemes. In: Proceedings of the Australasian Conference on Information Security and Privacy. 2004, 313–324
12. Tian H, Chen X, Li J. A short non-delegatable strong designated verifier signature. In: Proceedings of Australasian Conference on Information Security and Privacy. 2012, 261–279
13. Shim K A. On delegatability of designated verifier signature schemes. Information Sciences, 2014, 281: 365–372
14. Zhu Y, Gan G, Guo R, Huang D. Dual-mode broadcast encryption. Science China Information Sciences, 2018, 61(11): 118101
15. Zhang F, Susilo W, Mu Y, Chen X. Identity-based universal designated verifier signatures. In: Proceedings of the 2005 International Conference on Embedded and Ubiquitous Computing. 2005, 825–834
16. Zhang J, Mao J. A novel ID-based designated verifier signature scheme. Information Sciences, 2008, 178(3): 766–773
17. Sharma N, Sahu R A, Saraswat V, Sharma B K. Adaptively secure strong designated signature. In: Proceedings of International Conference on Cryptology in India. 2016, 43–60
18. Laguillaumie F, Vergnaud D. Multi-designated verifiers signatures. In: Proceedings of the 6th International Conference on Information and Communications Security. 2004, 495–507
19. Laguillaumie F, Vergnaud D. Multi-designated verifiers signatures: anonymity without encryption. Information Processing Letters, 2007, 102(2–3): 127–132
20. Ming Y, Wang Y. Universal designated multi verifier signature scheme without random oracles. Wuhan University Journal of Natural Sci-

ences, 2008, 13(6): 685–691
21. Seo S H, Hwang J Y, Choi K Y, Lee D H. Identity-based universal designated multi-verifiers signature schemes. Computer Standards and Interfaces, 2008, 30(5): 288–295
22. Lin C, Wu W, Huang X, Xu L. A new universal designated verifier transitive signature scheme for big graph data. Journal of Computer and System Science, 2017, 83(1): 73–83
23. Shi Y, Fan H, Liu Q. An obfuscatable designated verifier signature scheme. IEEE Transactions on Emerging Topics in Computing, 2017, 5(2): 271–285
24. Zhu Y, Gan G, Guo R, Huang D. PHE: an efficient traitor tracing and revocation for encrypted file syncing-and-sharing in cloud. IEEE Transactions on Cloud Computing, 2016, 6(4): 1110–1124
25. Zhu Y, Ahn G J, Hu H, Yau S S. An H G, Hu C. Dynamic audit services for outsourced storages in clouds. IEEE Transactions on Services Computing, 2013, 6(2): 227–238
26. Boneh D, Franklin M. Identity-based encryption from the weil pairing. In: Proceedings of Annual International Cryptology Conference. 2001, 213–229
27. Boneh D, Franklin M. Identity-based encryption from the weil pairing. SIAM Journal on Computing, 2003, 32(3): 586–615
28. Boneh D, Boyen X. Short signatures without random oracles. In: Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques. 2004, 56–73
29. Cheon J H. Security analysis of the strong Diffie-Hellman problem. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques. 2006, 1–11
30. Boneh D, Boyen X, Goh E J. Hierarchical identity based encryption with constant size ciphertext. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques. 2005, 440–456
31. Pointcheval D, Stern J. Security arguments for digital signatures and blind signatures. Journal of Cryptology, 2000, 13(3): 361–396
32. Schechter S, Parnell T, Hartemink A. Anonymous authentication of membership in dynamic groups. In: Proceedings of International Conference on Financial Cryptography. 1999, 184–195
33. Boneh D, Gentry C, Waters B. Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Proceedings of Annual International Cryptology Conference. 2005, 258–275
34. Delerablée C. Identity-based broadcast encryption with constant size ciphertexts and private keys. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security. 2007, 200–215
35. Zhu Y, Wang X, Ma D, Guo R. Identity-set-based broadcast encryption supporting cut-or-select with short ciphertext. In: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security. 2015, 191–202

E Chen received the BS degree from the department of School of Mathematics and Physics, University of Science and Technology Beijing, China. She is currently a PhD candidate with the department of School of Computer and Communication Engineering, University of Science and Technology Beijing, China. Her research interests include attribute based system and lattice based cryptography.

Yan Zhu was an associate professor of Computer Science with the Institute of Computer Science and Technology, Peking University, China, from 2007 to 2013. He was a visiting associate professor with the department of Computer Science and Engineering, Arizona State University, USA from 2008 to 2009. He was a visiting research investigator with the department of Computer and Information Science, University of Michigan-Dearborn, USA in 2012. He is currently a professor with the department of School of Computer and Communication Engineering, University of Science and Technology Beijing, China. His research interests include cryptography, secure computation, and network security.

Changlu Lin received the BS degree and MS degree in mathematics from the Fujian Normal University, China in 2002 and 2005, respectively, and received the PhD degree in information security from the state key laboratory of information security, Graduate University of Chinese Academy of Sciences, China in 2010. He works currently for the College of Mathematics and Informatics, and the Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, China. He is interested in cryptography and network security, and has conducted research in diverse areas, including secret sharing, multi-party computation, public key cryptography, and their applications.

Kewei Lv received his BSc and MSc degree in Math. from Qufu Normal University, China in 1992 and 1995, PhD in Math. from Peking University, China in 1999. He was appointed associate professor at Graduate University of Chinese Academy of Sciences, China in 2001 and associate professor at Institute of Information Engineering, Chinese Academy of Sciences, China in 2012. His research interests involves in theoretic cryptography, bit security, computational complexity, and Secure Multiparty Computation. In 2004, he won the second prize of Beijing Science and Technology Award.