1-2011

Efficient strong designated verifier signature schemes without random oracle or with non-delegatability

Qiong HUANG

Guomin YANG
Singapore Management University, gmyang@smu.edu.sg

Duncan S. WONG

Willy SUSILO

Int. J. Inf. Secur. (2011) 10:373–385
DOI 10.1007/s10207-011-0146-1

REGULAR CONTRIBUTION

Efficient strong designated verifier signature schemes without random oracle or with non-delegatability

Qiong Huang · Guomin Yang · Duncan S. Wong · Willy Susilo

Published online: 24 August 2011

Abstract Designated verifier signature (DVS) allows a signer to convince a designated verifier that a signature is generated by the signer without letting the verifier transfer the conviction to others, while the public can still tell that the signature must be generated by one of them. Strong DVS (SDVS) strengthens the latter part by restricting the public from telling whether the signature is generated by one of them or by someone else. In this paper, we propose two new SDVS schemes. Compared with existing SDVS schemes, the first new scheme has almost the same signature size and meanwhile, is proven secure in the standard model, while the existing ones are secure in the random oracle model. It has tight security reduction to the DDH assumption and the security of the underlying pseudorandom functions. Our second new scheme is the first SDVS supporting non-delegatability, the notion of which was introduced by Lipmaa, Wang and Bao in the context of DVS in ICALP 2005. The scheme is efficient and is provably secure in the random oracle model based on the discrete logarithm assumption and Gap Diffie–Hellman assumption.

Keywords Strong designated verifier signature · Non-delegatability · Non-transferability · Standard model · Signature scheme

Q. Huang (✉)
Department of Computer Science and Engineering,
South China Agricultural University, Guangzhou, China
e-mail: csqhuang@alumni.cityu.edu.hk

Q. Huang · D. S. Wong
Department of Computer Science, City University of Hong Kong,
Hong Kong, China

D. S. Wong
e-mail: duncan@cityu.edu.hk

G. Yang
Temasek Laboratories, National University of Singapore,
Singapore, Singapore
e-mail: tslyg@nus.edu.sg

W. Susilo
School of Computer Science and Software Engineering,
University of Wollongong, Wollongong, Australia
e-mail: wsusilo@uow.edu.au

1 Introduction

In undeniable signature [9], given a signature from Alice (the signer), Bob (the verifier) cannot check the validity of it by himself. Instead, Alice proves the validity (or invalidity) of the signature to Bob interactively. Alice can decide when to prove, but not whom to prove to. In Eurocrypt 1996, Jakobsson et al. [16] introduced the notion of *designated verifier proofs* (DVP), which allows Alice to designate a verifier, say Bob and proves the validity of a statement only to Bob. Bob cannot transfer this conviction to anyone else. This is called *non-transferability* and is usually achieved by proving either the validity of the statement or the knowledge of Bob's secret key, so that Bob is also able to produce the '*same*' proof transcript. Designated verifier signature (DVS) is the non-interactive version of DVP. A DVS is publicly verifiable and a valid DVS means that it is generated either by Alice or by Bob. Applications of DVS include voting [16], undeniable signature [12], deniable authentication [32] and several other cryptographic schemes where it is required that only certain parties can be convinced on something.

Also first pointed out in [16], in some applications, it is desirable if other than Alice and Bob, a third party cannot tell if a designated signature for Bob is indeed from Alice or from someone else. This motivates a variant of DVS called ***Strong Designated Verifier Signature*** (SDVS) [16]. An SDVS

requires that no third party can tell in a computationally indistinguishable way on which signer is the one who generated a given SDVS. SDVS was not formalized until 2004 [19] where Laguillaumie and Vergnaud defined a property called '*privacy of signer's identity*' which captures the 'strongness' of SDVS.

Jakobsson et al. [16] suggested a generic approach for constructing SDVS. Each user in the system holds two key pairs, one for DVS and the other for public key encryption. To sign a message M, Alice uses her signing key to generate a DVS signature on M for Bob and encrypts the signature under Bob's encryption key. The ciphertext is set as an SDVS. Upon receiving the SDVS, Bob decrypts it and checks the signature's validity using the DVS verification keys of both Alice and Bob. This approach is conceptually simple; however, it requires the full encryption of a DVS signature, which usually makes the scheme less efficient and the resulting signature bigger in size.

(*Non-delegatability*). In quite a number of DVS schemes such as [19,26,28,29], Alice can delegate her signing capability to a third party say Teddy, by giving Teddy a piece of information derived from her secret key without giving away her secret key. Teddy can then generate signatures on behalf of Alice. This property may be useful in some applications, e.g. proxy signature. However, there are applications in which this is not desirable, e.g. e-voting and online e-library subscription [22]. Suppose that Alice subscribes to an electronic library service provided by Bob. To access the service, Alice sends her DVS to Bob, so that Bob cannot show to others the access history of Alice. If the DVS scheme is delegatable, Alice can give the delegation information to Teddy so that Teddy can also access the service while Bob cannot tell whether the user is Alice or Teddy.

In ICALP 2005, Lipmaa et al. [22] introduced the notion called *non-delegatability* in the context of DVS. A DVS is *non-delegatable* if someone can generate a valid DVS signature on behalf of Alice for the designated verifier Bob, then it must '*know*' the secret key of either Alice or Bob. In other words, there exists an extractor such that given oracle access to the forger algorithm (and the message), it outputs the secret key of either the signer or the verifier. In [21,22], many DVS schemes were shown to be delegatable.

1.1 Our work

In the line of research on (S)DVS, there are some problems which remain unsolved since their introduction, for example:

1. constructing an SDVS without random oracles and with better efficiency than the generic construction [16];
2. constructing a non-delegatable SDVS; and
3. constructing a non-delegatable DVS in the standard model.

In this work, we give affirmative answers to the first two open problems. We propose a new SDVS scheme provably secure without random oracles. It is simple and more efficient than the aforementioned generic construction. Conceptually, our idea is to make signer S share a common key K with designated verifier V. To sign a message M for V, S uses K as the key to select a pseudorandom function PRF. The SDVS is simply the output of the PRF on input M. The security of the scheme, i.e. unforgeability and privacy of signer's identity, is tightly reduced to the pseudorandomness of the PRF and the intractability of Decisional Diffie–Hellman (DDH) problem. The scheme also enjoys perfect non-transferability, as S and V can produce the same signature on any message.

This new scheme is delegatable, because the key for the pseudorandom function is derived from the common key shared between S and V, and anyone with this key can sign messages on behalf of S. We then propose another efficient construction of SDVS, which to the best of our knowledge, is the *first* non-delegatable SDVS scheme. Our scheme makes use of the standard Fiat–Shamir heuristic to convert a proof of knowledge proof into a non-interactive one. The non-delegatability stems from the fact that the signature is a (non-interactive) proof of knowledge of the secret key of either S or V. The price we pay is that the security of the second scheme can only be proven in the random oracle model [4]. Unforgeability of the scheme is based on discrete logarithm (DL) assumption, while the privacy of signer's identity relies on the intractability of Gap Diffie–Hellman (GDH) problem.

1.2 Related work

DVS has attracted a lot of attention since its introduction, and many variants have been proposed. They include *ring signature* [25,27], in which the validity of a signature shows that the signature is from a group of signers but tells nothing about the actual identity of the real signer, *universal designated verifier signature* (UDVS) [14,18,28,31,33], in which someone who holds the signer's standard signature can designate any party as the verifier and transform the signature to a DVS signature for that party, *multi-designated verifiers signature* (MDVS) [16,20], in which the signer can designate multiple parties as verifiers for checking the validity of a signature and *identity-based designated verifier signature* (IBDVS) [5,11,15,30], which is a DVS in the identity-based setting. In the study of SDVS, Huang et al. [15] and Bhaskar et al. [5] independently proposed an efficient construction, which has short signatures. A signature in their scheme is simply the hash of a message and a common key shared

between S and V. The security of their scheme is analyzed in the random oracle model.

Since the introduction of non-delegatability [22] in the context of DVS, many schemes have been shown to be delegatable, e.g. [19,26,28,29]. In Lipmaa et al. [22], also proposed the first non-delegatable DVS with security based on DDH assumption in the random oracle model. Huang et al. [13] constructed a UDVS scheme with non-delegatability from Boneh–Lynn–Shacham signature [8]. Very recently, Huang et al. [11] proposed the first non-delegatable IBDVS scheme, which is based on Gentry and Silverberg's hierarchical identity-based encryption scheme [10].

1.3 Outline

In the next section, we give the formal definition of SDVS, as well as its security model. We then review in Sect. 3 the definition of PRFs and the assumptions used in our schemes. The two new SDVS schemes, one without random oracles and the other one with non-delegatability property, are presented in Sects. 4 and 5, respectively. We compare these two new schemes with some existing (S)DVS schemes in Sect. 6 and conclude the paper in Sect. 7.

2 Strong designated verifier signature

Definition 1 (*SDVS*) A *Strong designated verifier signature (SDVS)* scheme consists of the following (probabilistic) polynomial time (PPT) algorithms.

- Kg: takes as input 1^k where k is a security parameter and outputs a public/secret key pair, i.e. $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{Kg}(1^k)$.
- Sign: takes as input the secret key of signer S, public keys of S and V (the designated verifier) and message M, and outputs a signature σ, i.e. $\sigma \leftarrow \mathsf{Sign}(\mathtt{sk}_s, \mathtt{pk}_s, \mathtt{pk}_v, M)$.
- Ver: takes as input the secret key of V, public keys of S and V, message M and an alleged signature σ, and outputs a bit b, which is 1 for acceptance of σ as a valid signature, and 0 for rejection, i.e. $b \leftarrow \mathsf{Ver}(\mathtt{sk}_v, \mathtt{pk}_s, \mathtt{pk}_v, M, \sigma)$.

The **correctness** of SDVS requires that for any

$$(\mathtt{pk}_s, \mathtt{sk}_s) \leftarrow \mathsf{Kg}(1^k), (\mathtt{pk}_v, \mathtt{sk}_v) \leftarrow \mathsf{Kg}(1^k)$$

and any message $M \in \{0,1\}^*$, we have

$$\Pr[\mathsf{Ver}(\mathtt{sk}_v, \mathtt{pk}_s, \mathtt{pk}_v, M, \mathsf{Sign}(\mathtt{sk}_s, \mathtt{pk}_s, \mathtt{pk}_v, M)) = 1] = 1.$$

A secure SDVS should be unforgeable, non-transferable and satisfying the privacy of signer's identity. If an SDVS is said to be non-delegatable, it should also support non-delegatability.

Unforgeability It requires that no one other than the signer S and the designated verifier V can produce a valid signature. We consider the following game played between a challenger C and a PPT adversary $\mathcal{A}$.

1. C prepares the key pairs for S and V, i.e. $(\mathtt{pk}_s, \mathtt{sk}_s)$ and $(\mathtt{pk}_v, \mathtt{sk}_v)$, and gives $(\mathtt{pk}_s, \mathtt{pk}_v)$ to $\mathcal{A}$.
2. $\mathcal{A}$ issues queries to the following oracles.
   - $\mathcal{O}_{\mathsf{Sign}}$: Given a message M, it uses $\mathtt{sk}_s$ to generate a signature σ on M, which is valid w.r.t. $\mathtt{pk}_s$ and $\mathtt{pk}_v$, and returns it to $\mathcal{A}$.
   - $\mathcal{O}_{\mathsf{Sim}}$: Given a message M, it uses $\mathtt{sk}_v$ to generate a simulated signature σ on M, which is valid w.r.t. $\mathtt{pk}_s$ and $\mathtt{pk}_v$, and returns it to $\mathcal{A}$.
   - $\mathcal{O}_{\mathsf{Ver}}$: Given a query of the form (M, σ), the oracle returns a bit b which is 1 if σ is a valid signature on M w.r.t. $\mathtt{pk}_s$ and $\mathtt{pk}_v$, and 0 otherwise.
3. $\mathcal{A}$ outputs a forgery, (M^*, σ^*) and wins if
   (a) $\mathsf{Ver}(\mathtt{sk}_v, \mathtt{pk}_s, \mathtt{pk}_v, M^*, \sigma^*) = 1$, and
   (b) it did not query $\mathcal{O}_{\mathsf{Sign}}$ and $\mathcal{O}_{\mathsf{Sim}}$ on input M^*.

Definition 2 (*Unforgeability*) An SDVS scheme is $(t, q_{\mathsf{Sign}}, q_{\mathsf{Sim}}, q_{\mathsf{Ver}}, \epsilon)$-*unforgeable* if no adversary $\mathcal{A}$ which runs in time at most t, issues at most q_{Sign} queries to $\mathcal{O}_{\mathsf{Sign}}$, q_{Sim} queries to $\mathcal{O}_{\mathsf{Sim}}$ and q_{Ver} queries to $\mathcal{O}_{\mathsf{Ver}}$, can win the game with probability at least ϵ.

Non-transferability Given a message/signature pair (M, σ), it should be infeasible for any PPT distinguisher to tell whether σ was generated by the signer or simulated by the designated verifier. Formally, we consider the following definition.

Definition 3 (*Non-transferability*) An SDVS is *non-transferable* if there exists a PPT simulation algorithm Sim which takes as input $\mathtt{sk}_v, \mathtt{pk}_s, \mathtt{pk}_v$ and a message M and outputs a simulated signature that is indistinguishable from real signatures generated by the signer on the same message. That is, for any PPT distinguisher $\mathcal{D}$, any $(\mathtt{pk}_s, \mathtt{sk}_s) \leftarrow \mathsf{Kg}(1^k)$, $(\mathtt{pk}_v, \mathtt{sk}_v) \leftarrow \mathsf{Kg}(1^k)$ and any message $M \in \{0,1\}^*$, it holds that

$$\left| \Pr\left[\begin{array}{r} \sigma_0 \leftarrow \mathsf{Sign}(\mathtt{sk}_s, \mathtt{pk}_s, \mathtt{pk}_v, M), \\ \sigma_1 \leftarrow \mathsf{Sim}(\mathtt{sk}_v, \mathtt{pk}_s, \mathtt{pk}_v, M), \\ b \leftarrow_{\$} \{0,1\}, \\ b' \leftarrow \mathcal{D}(\mathtt{pk}_s, \mathtt{sk}_s, \mathtt{pk}_v, \mathtt{sk}_v, \sigma_b) \end{array} : b' = b \right] - \frac{1}{2} \right| < \epsilon(k),$$

where $\epsilon(k)$ is a negligible function[1] in the security parameter k, and the probability is taken over the randomness used in Kg, Sign and Sim, and the random coins consumed by $\mathcal{D}$. If the probability is equal to $1/2$, we say that the SDVS scheme is *perfectly non-transferable* (or *source hiding*).

Privacy of Signer's Identity (PSI) First defined by Laguillaumie and Vergnaud [19], it formalizes the motivation of SDVS, i.e. no one can tell signatures generated by signer S_0 for V apart from signatures by S_1 for V, if it does not know the secret key of V. Below is the formal definition of PSI, modeled by a game played between the challenger C and a distinguisher $\mathcal{D}$.

1. C generates key pairs for signers S_0, S_1 and verifier V, i.e. $(\mathtt{pk}_{s_0}, \mathtt{sk}_{s_0})$, $(\mathtt{pk}_{s_1}, \mathtt{sk}_{s_1})$ and $(\mathtt{pk}_v, \mathtt{sk}_v)$, and invokes $\mathcal{D}$ on input $(\mathtt{pk}_{s_0}, \mathtt{pk}_{s_1}, \mathtt{pk}_v)$.
2. $\mathcal{D}$ issues queries adaptively as in the unforgeability game, except that now all the oracles take an additional input $d \in \{0, 1\}$ indicating which signer responds to the query. That is, the oracles generate and verify signatures w.r.t. $\mathtt{pk}_{s_d}$ and $\mathtt{pk}_v$.
3. $\mathcal{D}$ submits a message M^*. C tosses a coin $b \in \{0, 1\}$, computes the challenge signature $\sigma^* \leftarrow \mathsf{Sign}(\mathtt{sk}_{s_b}, \mathtt{pk}_{s_b}, \mathtt{pk}_v, M^*)$ and returns σ^* to $\mathcal{D}$.
4. $\mathcal{D}$ continues to issue queries as in Step 2. Finally it outputs a bit b' and wins the game if
   (a) $b' = b$; and
   (b) it did not query $\mathcal{O}_{\mathsf{Ver}}$ on input (d, M^*, σ^*) for any $d \in \{0, 1\}$.

Definition 4 (*Privacy of Signer's Identity*) An SDVS scheme is $(t, q_{\mathsf{Sign}}, q_{\mathsf{Sim}}, q_{\mathsf{Ver}}, \epsilon)$-*PSI-secure* if no adversary $\mathcal{D}$ which runs in time at most t, issues at most q_{Sign} queries to $\mathcal{O}_{\mathsf{Sign}}$, q_{Sim} queries to $\mathcal{O}_{\mathsf{Sim}}$ and q_{Ver} queries to $\mathcal{O}_{\mathsf{Ver}}$, can win the game above with probability that deviates from $1/2$ by more than ϵ.

Remark 1 If the signing/simulation algorithm of an SDVS scheme is deterministic, $\mathcal{D}$ in the game above should also be restricted from asking $\mathcal{O}_{\mathsf{Sign}}$ and $\mathcal{O}_{\mathsf{Sim}}$ for a (simulated) signature on M^* for any $d \in \{0, 1\}$. Otherwise, $\mathcal{D}$ breaks the PSI trivially.

Non-delegatability Intuitively, it requires that if one produces a valid signature on a message, it must 'know' the secret key of either S or V. So a signature itself is a proof of knowledge of the secret key of either S or V. Formally, we consider the following definition.

[1] A function $f : \mathbb{N} \to \mathbb{N}$ is *negligible* in the security parameter k if for every polynomial $q(\cdot)$, there exists some $k_0 \in \mathbb{N}$ such that for every $k > k_0$, $f(k) < 1/q(k)$.

Definition 5 (*Non-delegatability*) Let $\kappa \in [0, 1]$ be the knowledge error and $\mathcal{F}$ a forger algorithm. Let $\mathcal{F}_M$ be $\mathcal{F}$ with M as its input, and oracle calls to $\mathcal{F}_M$ be counted as one step. An SDVS scheme is *non-delegatable with knowledge error* κ if there exists a positive polynomial $\mathrm{poly}(\cdot)$ and a probabilistic oracle machine $\mathcal{K}$ such that for every PPT algorithm $\mathcal{F}$, machine $\mathcal{K}$ satisfies the following condition:

– Denote by ϵ the probability that $\mathcal{F}$ on input M produces a valid signature on M. For every $(\mathtt{pk}_s, \mathtt{sk}_s) \leftarrow \mathsf{Kg}(1^k)$, $(\mathtt{pk}_v, \mathtt{sk}_v) \leftarrow \mathsf{Kg}(1^k)$, and every message $M \in \{0, 1\}^*$, if $\epsilon > \kappa$, then on input M and on (black box) oracle access to $\mathcal{F}_M, \mathcal{K}$ produces either $\mathtt{sk}_s$ or $\mathtt{sk}_v$ in expected polynomial time with probability at least $(\epsilon - \kappa)/\mathrm{poly}(k)$.

Remark 2 Our definition of non-delegatability is different from the original one given in [22]. The definition in [22] emphasizes that $\mathcal{K}$ can output $\mathtt{sk}_s$ or $\mathtt{sk}_v$ in time polynomially proportional to the inverse of $\epsilon - \kappa$, while our definition requires that $\mathcal{K}$ outputs a secret key in polynomial time with probability polynomially related to $\epsilon - \kappa$. However, similar to the equivalence between the definitions of proofs of knowledge [2], it can be shown that Definition 5 is equivalent to that in [22].

Also notice that the non-delegatability defined above and that in [22] is for signature generation. Another interesting and related property is *non-delegatability for signature verification*. This property requires that in order to verify a designated signature, the verifier must 'know' the designated verifier's secret key. One can formally define non-delegatability for verification analogously to Definition 5. However, our scheme proposed in Sect. 5 can be only shown to be non-delegatable for signing but not for verification. The difficulty we encounter is that using forking lemma in the random oracle model (as we do in the proof of non-delegatability for signing of our second scheme) does not seem to help us extract the entire secret key from the verifier's one-bit output, because given a signature the output of the hash function has already been determined during the process of signature generation. If we rewind the verifier and assign a new hash value, this would make the verification equation no longer hold. We leave designing an SDVS scheme satisfying the non-delegatability for verification as an open problem.

3 Tools and assumptions

In this section, we introduce the underlying tools and assumptions used in our SDVS schemes.

Pseudorandom Function Ensemble (PRF) Intuitively, the output distribution of a pseudorandom function should be

indistinguishable from that of a truly random function, if the seed for the function is random.

Definition 6 (*PRF* [23]) Let $\{A_k, B_k\}_{k\in\mathbb{N}}$ be a sequence of domains and $\mathbf{F} = \{\mathrm{PRF}_k\}_{k\in\mathbb{N}}$ a function ensemble such that the random variable PRF_k assumes values in the set of $A_k \to B_k$ functions. $\mathbf{F}$ is called an *efficiently computable pseudorandom function ensemble* if

1. (*efficient computation*) There exist PPT algorithms $\mathcal{I}$ and $\mathcal{V}$, and a mapping from strings to functions, ϕ, such that $\phi(\mathcal{I}(1^k))$ and PRF_k are identically distributed and $\mathcal{V}(i, x) = (\phi(i))(x)$.
2. ((t, ϵ)-*pseudorandomness*) For every PPT oracle machine $\mathcal{D}$ which runs in time at most t,

$$\left|\Pr\left[\mathcal{D}^{\mathrm{PRF}_k}(1^k) = 1\right] - \Pr\left[\mathcal{D}^{\mathrm{RF}_k}(1^k) = 1\right]\right| < \epsilon,$$

where $\mathbf{R} = \{\mathrm{RF}_k\}_{k\in\mathbb{N}}$ is the uniform function ensemble (i.e. $\forall k$, RF_k is uniformly distributed over the set of $A_k \to B_k$ functions).

Assumptions Let p, q be two large primes such that $q \mid p-1$. Let $\mathbb{G}$ be a multiplicative cyclic group of order q and g be its generator.

Definition 7 (*DL Assumption*) The *discrete logarithm (DL)* assumption (t, ϵ)-holds in $\mathbb{G}$ if there is no algorithm $\mathcal{A}$ which runs in time at most t, and

$$\Pr[x \leftarrow_{\$} \mathbb{Z}_q; x' \leftarrow \mathcal{A}(g, g^x) \ : \ x' = x] \geq \epsilon,$$

where the probability is taken over the random choice of x and the random coins used by $\mathcal{A}$.

Definition 8 (*DDH Assumption*) The *Decisional Diffie–Hellman (DDH)* assumption (t, ϵ) holds in $\mathbb{G}$ if there is no algorithm $\mathcal{A}$ which runs in time at most t, and

$$\Pr[a, b, c \leftarrow_{\$} \mathbb{Z}_q; Z_0 \leftarrow g^{ab}; Z_1 \leftarrow g^c; d \leftarrow_{\$} \{0, 1\};$$
$$d' \leftarrow \mathcal{A}(g, g^a, g^b, Z_d) : d' = d] \geq \epsilon,$$

where the probability is taken over the random choices of $a, b, c \in \mathbb{Z}_q$, $d \in \{0, 1\}$ and random coins used by $\mathcal{A}$.

Definition 9 (*GDH Assumption*) The *Gap Diffie–Hellman (GDH)* assumption (t, ϵ) holds in $\mathbb{G}$ if there is no algorithm $\mathcal{A}$ that given access to a DDH oracle $\mathcal{O}_{\mathrm{ddh}}$ which on input $(g_1, g_2, h_1, h_2) \in \mathbb{G}^4$ tells if $\log_{g_1} h_1 = \log_{g_2} h_2$, runs in time at most t, and

$$\Pr[a, b \leftarrow_{\$} \mathbb{Z}_q; Z \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{ddh}}}(g, g^a, g^b) : Z = g^{ab}] \geq \epsilon,$$

where the probability is taken over the random choices of $a, b \in \mathbb{Z}_q$ and the random coins used by $\mathcal{A}$.

4 SDVS without random oracle

4.1 The scheme

We now propose an SDVS scheme which is secure without random oracles. It is conceptually simple. A signature is the output of a pseudorandom function on a message under a key derived from the public key of the signer or the verifier and the secret key of the other.

Let $\mathbb{G}$ be a cyclic multiplicative group of large prime order q and g be its generator. Let $\lambda \stackrel{\mathrm{def}}{=} \lambda(\cdot)$, $\ell \stackrel{\mathrm{def}}{=} \ell(\cdot)$ and $\gamma \stackrel{\mathrm{def}}{=} \gamma(\cdot)$ be positive polynomials in the security parameter k. Let $\mathbf{F} = \{\mathrm{PRF}_K\}_{K\in\{0,1\}^\gamma}$ be a family of pseudorandom functions $\mathrm{PRF}_K : \{0, 1\}^\lambda \to \{0, 1\}^\ell$ with key space $\{0, 1\}^\gamma$. We assume that the message space is $\{0, 1\}^\lambda$. This assumption can be removed if we use a collision-resistant hash function to map arbitrarily long messages to $\{0, 1\}^\lambda$. We also assume that elements of $\mathbb{G}$ can be encoded to γ-bit strings. This new SDVS scheme is described in Fig. 1, where for simplicity, we omit the usage of the encoding, and simply use the shared key (i.e. $K = g^{x_s x_v}$) as the key for the pseudorandom function.

Difference from [5,15] The SDVS scheme in [5,15] has short signature and shares a similar structure to that of ours. In [5,15], a signature is the output of a hash function on M and a common key shared between S and V. The security is shown in the random oracle model based on the Gap Diffie–Hellman assumption (see Table 1, Sect. 6 for a detailed comparison). It is also suggested in [15] to replace the hash function with a symmetric encryption SE for obtaining a scheme secure in the standard model. However, it is not clear if we can obtain the provable security of the resulting scheme based on the security of SE. The indistinguishability under chosen-ciphertext attacks (IND-CCA) does not guarantee that one cannot compute the encryption of a new message without the knowledge of the secret key. Even if we assume that SE is an authenticated encryption [3] which has both indistinguishability under chosen-ciphertext attacks and ciphertext integrity (IND-CTXT), the resulting SDVS scheme may not satisfy all the security properties. Specifically, the IND-CCA security does not imply that ciphertexts generated using two different keys are indistinguishable. An adversary may be able to learn some information about signers' keys from issuing signing queries, which avails it to distinguish the two signers' signatures. Hence, the PSI of the SE-based SDVS scheme may not be guaranteed.

In our construction instead, we find that if the output of the signing algorithm of an SDVS scheme looks indistinguishable from random strings, it also turns out that two different signers' signatures would be indistinguishable to each other. As demonstrated in the proof of Theorem 3 (to be shown later), the pseudorandomness of the SDVS signatures is a sufficient condition to guarantee the PSI property. However,

$\mathsf{Kg}(1^k)$:
$x \leftarrow_{\$} \mathbb{Z}_q$
return $(\mathtt{pk}, \mathtt{sk}) := (g^x, x)$

$\mathsf{Sign}(\mathtt{sk}_s, \mathtt{pk}_s, \mathtt{pk}_v, M)$:
parse $\mathtt{sk}_s$ as x_s
set $K \leftarrow \mathtt{pk}_v^{x_s}$
return $\sigma \leftarrow \mathrm{PRF}_K(M)$

$\mathsf{Ver}(\mathtt{sk}_v, \mathtt{pk}_s, \mathtt{pk}_v, M, \sigma)$:
parse $\mathtt{sk}_v$ as x_v
set $K \leftarrow \mathtt{pk}_s^{x_v}$
return $\sigma \stackrel{?}{=} \mathrm{PRF}_K(M)$

Fig. 1 Our SDVS scheme in the standard model, SDVS_1

Table 1 Comparison with other schemes

	[16]	[20]	[22]	[5,15]	SDVS_1	SDVS_2
Type	DVS	MDVS	DVS	SDVS	SDVS	SDVS
PK size	$1\mathbb{Z}_p$	$1\mathbb{G}$	$2\mathbb{Z}_p$	$1\mathbb{Z}_p$	$2\mathbb{Z}_p$	$1\mathbb{Z}_p$
Signature size	$2\mathbb{Z}_p + 3\mathbb{Z}_q$	$n_r + 1\mathbb{G}_T$	$4\mathbb{Z}_q$	$\lvert \mathrm{H}(\cdot) \rvert$	$\lfloor \frac{k}{2} \rfloor$	$4\mathbb{Z}_q$
Sign cost	4E	4E	4E	1E	$O(k)$E	4E
Verification cost	6E	1E + 3P	6E	1E	$O(k)$E	5E
PSI	×	√	×	√	√	√
Non-delegatability	×	×	√	×	×	√
Standard model	×	×	×	×	√	×
Assumptions	DL	CDH + GBDH	DDH	GDH	DDH	DL + GDH

it is not known if it is also a necessary condition. We leave it as our future work to find a new SDVS construction which uses a primitive weaker than PRF.

4.2 Security analysis

As a pseudorandom function is a (deterministic) message authentication code with strong unforgeability under chosen message attacks [3], we have that after obtaining many signatures on messages of its choice, the adversary cannot forge a new pair. Also, the pseudorandomness of the function tells that after obtaining many outputs, the output of the function on a new input still looks random to the distinguisher. Hence, the adversary against the PSI cannot distinguish which signer produced the challenge signature. Formally, we have the following theorems, which together show that the security of SDVS_1 is tightly reduced to the pseudorandomness of PRF and the hardness of DDH problem.

Theorem 1 *Suppose that $\mathcal{A}$ is an adversary that $(t, q_{\mathsf{Sign}}, q_{\mathsf{Ver}}, \epsilon)$-breaks the unforgeability of SDVS_1. There exists an algorithm $\mathcal{A}_1$ that $(t_1, \epsilon_{\mathrm{ddh}})$-breaks the DDH assumption and an algorithm $\mathcal{A}_2$ that $(t_2, \epsilon_{\mathrm{prf}})$-breaks the pseudorandomness of $\mathbf{F}$ with $t_1, t_2 \approx t$ and $\epsilon_{\mathrm{ddh}} + \epsilon_{\mathrm{prf}} > \epsilon - (q_{\mathsf{Sign}} + q_{\mathsf{Sim}} + q_{\mathsf{Ver}})2^{-\ell}$.*

As we shall see later (Theorem 2), the scheme SDVS_1 is perfectly non-transferable. Hence, it suffices to consider the case in which the adversary in the games of unforgeability and privacy of signer's identity does not make query to $\mathcal{O}_{\mathsf{Sim}}$, as the queries can perfectly be handled by $\mathcal{O}_{\mathsf{Sign}}$.

Proof We prove the theorem by a series of games. Let $\mathcal{A}$ be an adversary against the unforgeability of the SDVS_1 above. Let G_i be the i-th game, and X_i be the event that $\mathcal{A}$ outputs a valid forgery in Game G_i without violating the constraints.

G_0: This is the original game. The challenger C chooses $a, b \leftarrow_{\$} \mathbb{Z}_q$ and invokes $\mathcal{A}$ on input $(\mathtt{pk}_s, \mathtt{pk}_v) = (g^a, g^b)$. Let $K := \mathtt{pk}_s^b = \mathtt{pk}_v^a = g^{ab}$. For each signing query M and each verification (M, σ), C simulates the corresponding answer using g^{ab} as the key for the pseudorandom function. The adversary finally outputs (M^*, σ^*) and wins the game if M^* is new and $\sigma^* = \mathrm{PRF}_{g^{ab}}(M^*)$. By definition, we have that

$$\Pr[X_0] = \epsilon. \tag{1}$$

G_1: This game differs from Game G_0 in that when handling the adversary's queries, the key for the pseudorandom function is chosen at random from $\mathbb{G}$, i.e. $K \leftarrow_{\$} \mathbb{G}$. Besides, the validity of the adversary's forgery is checked w.r.t. this random key. If the success probabilities of the adversaries in games G_1 and G_0 differ non-negligibly, it leads to an algorithm for breaking the DDH assumption. Therefore, we have that

$$|\Pr[X_1] - \Pr[X_0]| \leq \epsilon_{\mathrm{ddh}} \tag{2}$$

for some algorithm $\mathcal{A}_1$ that $(t_1, \epsilon_{\mathrm{ddh}})$-breaks the DDH assumption with $t_1 \approx t$. $\mathcal{A}_1$ works as below.

Given a DDH problem instance, i.e. $\mathbb{G}, g, q, g^a, g^b, Z$ where Z is either equal to g^{ab} or a random element of $\mathbb{G}$, $\mathcal{A}_1$ sets $\mathtt{pk}_s := g^a$ and $\mathtt{pk}_v := g^b$ and invokes $\mathcal{A}$ on input $(\mathtt{pk}_s, \mathtt{pk}_v)$ and the group description $\mathbb{G}, g, q$. To answer a signing query on message M, $\mathcal{A}_1$ computes and returns $\sigma \leftarrow \mathrm{PRF}_Z(M)$; to answer a verification query (M, σ), it computes and returns $\sigma \stackrel{?}{=} \mathrm{PRF}_Z(M)$. Finally, $\mathcal{A}$ outputs its forgery, (M^*, σ^*). The adversary $\mathcal{A}_1$ then tests whether

$\sigma^* = \mathrm{PRF}_Z(M^*)$. If so, it outputs 1, meaning that $Z = g^{ab}$; otherwise, it outputs 0, meaning that Z is randomly chosen from $\mathbb{G}$. If $Z = g^{ab}$, the game simulated by $\mathcal{A}_1$ is Game G_0, and $\mathcal{A}$ outputs a valid forgery with probability $\Pr[X_0]$. If Z is a random element in $\mathbb{G}$, the game simulated by $\mathcal{A}_1$ is Game G_1, and $\mathcal{A}$ outputs a valid forgery with probability $\Pr[X_1]$. Let b be the bit output by $\mathcal{A}_1$. Then, we have

$$\begin{aligned}&|\Pr[b = 1 | Z = g^{ab}] - \Pr[b = 1 | Z \leftarrow_{\$} \mathbb{G}]| \\ &= |\Pr[X_0] - \Pr[X_1]|.\end{aligned}$$

By the DDH assumption, we then have Eq. (2).

G_2: In Game G_1, the key for the pseudorandom function is independent from the two public keys, and the adversary can obtain information about the function only via issuing queries. We now modify the game so that the pseudorandom function is replaced with a truly random function. That is, the signature on a message is now randomly chosen from $\{0, 1\}^\ell$. We have

$$|\Pr[X_2] - \Pr[X_1]| \leq \epsilon_{\mathrm{prf}}. \tag{3}$$

To see the equation above, we construct an algorithm $\mathcal{A}_2$ to $(t_2, \epsilon_{\mathrm{prf}})$-break the pseudorandomness of $\mathbf{F}$ with $t_2 \approx t$.

Given an oracle function $F(\cdot)$ which is either a pseudorandom function chosen from $\mathbf{F}$ or a truly random function, the adversary $\mathcal{A}_2$ randomly selects a group $(\mathbb{G}, g, q)$ and chooses $a, b \leftarrow \mathbb{Z}_q$ and invokes $\mathcal{A}$ on input $(\mathtt{pk}_s, \mathtt{pk}_v) = (g^a, g^b)$ and the group description. $\mathcal{A}_2$ maintains a table T, which is initially empty. On input a signing query M, if there is a tuple (M, σ) in T, $\mathcal{A}_2$ returns σ. Otherwise, it submits M to function F and obtains an answer σ. It returns σ to $\mathcal{A}$ and stores (M, σ) in T. On input a verification query (M, σ), if (M, σ) is in table T, $\mathcal{A}_2$ returns 1; otherwise, it submits M to F and obtains σ'. It stores (M, σ') in T and returns 1 if $\sigma' = \sigma$ and 0 otherwise. Since the pseudorandom function is deterministic, there is only one signature on each message. Hence, the simulation of oracle $\mathcal{O}_{\mathsf{Sign}}$ is perfect. Finally, $\mathcal{A}$ outputs a forgery (M^*, σ^*) where M^* is distinct from all messages it has ever submitted to $\mathcal{A}_2$. $\mathcal{A}_2$ submits M^* to F and obtains σ'^*. If $\sigma'^* = \sigma^*$, it outputs 1, indicating that its oracle function is a pseudorandom function from $\mathbf{F}$; otherwise, it outputs 0, indicating that its oracle is a truly random function. If F is chosen from $\mathbf{F}$, $\mathcal{A}_2$ perfectly simulated Game G_1; if F is truly random, $\mathcal{A}_2$ perfectly simulated Game G_2. Therefore, we obtain Eq. (3).

Note that in Game G_2, the signature on a message is a random string from $\{0, 1\}^\ell$; therefore, after issuing q_{Sign} signing queries, q_{Sim} simulation queries and q_{Ver} verification queries, the probability that the forgery output by the adversary is valid in Game G_2 is upper bounded by

$$\begin{aligned}\Pr[X_2] &\leq (2^\ell - q_{\mathsf{Sign}} - q_{\mathsf{Sim}} - q_{\mathsf{Ver}})^{-1} \\ &< (q_{\mathsf{Sign}} + q_{\mathsf{Sim}} + q_{\mathsf{Ver}}) 2^{-\ell}.\end{aligned} \tag{4}$$

Combining Eqs. (1)–(4), we get that

$$\begin{aligned}\epsilon &= \Pr[X_0] \\ &\leq |\Pr[X_0] - \Pr[X_1]| + |\Pr[X_1] - \Pr[X_2]| + \Pr[X_2] \\ &< \epsilon_{\mathrm{ddh}} + \epsilon_{\mathrm{prf}} + (q_{\mathsf{Sign}} + q_{\mathsf{Sim}} + q_{\mathsf{Ver}}) 2^{-\ell}.\end{aligned}$$

This completes the proof. □

Theorem 2 SDVS_1 *is perfectly non-transferable.*

Proof To simulate the signer's signature on M, the designated verifier does the following:

$$\text{set } K \leftarrow \mathtt{pk}_s^{x_v} \text{ and return } \sigma \leftarrow \mathrm{PRF}_K(M).$$

Since both the signer and the verifier can compute the same key $K = g^{x_s x_v}$, they can generate the same signature on any message M, i.e. $\sigma = \mathrm{PRF}_K(M)$ for each message M. □

Theorem 3 *Suppose that there is an adversary $\mathcal{D}$ that $(t, q_{\mathsf{Sign}}, q_{\mathsf{Ver}}, \epsilon)$-breaks the PSI of SDVS_1. Then, there exists an adversary $\mathcal{A}_1$ that $(t_1, \epsilon_{\mathrm{ddh}})$-breaks DDH assumption and an adversary $\mathcal{A}_2$ that $(t_2, \epsilon_{\mathrm{prf}})$-breaks the pseudorandomness of $\mathbf{F}$ with $t_1, t_2 \approx t$ and $\epsilon_1 + \epsilon_2 \geq \epsilon/2$.*

Proof Let $\mathcal{D}$ be the distinguisher against privacy of signer's identity. We consider the following games and denote by X_i the event that the adversary outputs the correct bit in Game G_i.

G_0: This is the original game. C chooses $a_0, a_1, c \in \mathbb{Z}_q$ at random and invokes the adversary $\mathcal{D}$ on input $(\mathtt{pk}_{s_0}, \mathtt{pk}_{s_1}, \mathtt{pk}_v) = (g^{a_0}, g^{a_1}, g^c)$. It simulates oracles for $\mathcal{D}$ using a_0, a_1, c. Let $K_0 = g^{a_0 c}$ and $K_1 = g^{a_1 c}$. The real keys used in the simulation of oracles are K_0 and K_1, and K_b is used in the generation of the challenge signature σ^* for a random $b \in \{0, 1\}$. By definition, we have

$$\Pr[X_0] = \epsilon + \frac{1}{2}. \tag{5}$$

G_1: This game differs from Game G_0 in that now the key K_0 is chosen at random from $\mathbb{G}$. The difference between $\mathcal{D}'s$ success probabilities in Game G_0 and Game G_1 is then bounded by the intractability of DDH problem, and we have

$$|\Pr[X_1] - \Pr[X_0]| \leq \epsilon_{\mathrm{ddh}}. \tag{6}$$

To see the equation above, we use $\mathcal{D}$ to build another algorithm $\mathcal{D}_1$ to $(t_1, \epsilon_{\mathrm{ddh}})$-break the DDH assumption with $t_1 \approx t$.

Given a random instance of DDH problem, i.e. $(\mathbb{G}, g, q, g^{a_0}, g^c, Z)$ where a_0, c are random elements of $\mathbb{Z}_q$ unknown

to it, $\mathcal{D}_1$ randomly chooses $a_1 \in \mathbb{Z}_q$ and invokes $\mathcal{D}$ on input $(\mathtt{pk}_{s_0}, \mathtt{pk}_{s_1}, \mathtt{pk}_v) = (g^{a_0}, g^{a_1}, g^c)$ and the group description. It also sets $K_0 := Z$ and $K_1 = (g^c)^{a_1}$.

Given a signing query (d, M), $\mathcal{D}_1$ computes and returns $\sigma \leftarrow \mathrm{PRF}_{K_d}(M)$. Given a verification query (d, M, σ), it computes and returns $\sigma \stackrel{?}{=} \mathrm{PRF}_{K_d}(M)$. When $\mathcal{D}$ submits its challenge message M^*, $\mathcal{D}_1$ chooses a random bit b and returns $\sigma^* \leftarrow \mathrm{PRF}_{K_b}(M^*)$. The successive queries issued by $\mathcal{D}$ are handled as above. Finally, $\mathcal{D}$ outputs a bit b'. Then, $\mathcal{D}_1$ outputs 1 if $b' = b$, indicating that $Z = g^{a_0 c}$ and 0 otherwise, indicating that Z is a random element of $\mathbb{G}$.

It is readily seen that if $Z = g^{a_0 c}$, the game simulated by D is identical to Game G_0, and $\mathcal{D}$ outputs the correct answer with probability $\Pr[X_0]$; on the other hand, if Z is a random element of $\mathbb{G}$, the game is identical to Game G_1, and $\mathcal{D}$ outputs the correct bit with probability $\Pr[X_1]$. Let the bit output by $\mathcal{D}_1$ be δ. We have that

$$\begin{aligned}&|\Pr[\delta = 1 | Z = g^{a_0 c}] - \Pr[\delta = 1 | Z \leftarrow_{\$} \mathbb{G}]| \\ &\quad = |\Pr[X_0] - \Pr[X_1]|.\end{aligned}$$

Therefore, we obtain the Eq. (6).

G_2: Now, we replace K_1 with a random element of $\mathbb{G}$ as well. Note that in this game, both K_0 and K_1 are randomly chosen from $\mathbb{G}$. Similar to Game G_1, we have that

$$|\Pr[X_2] - \Pr[X_1]| \leq \epsilon_{\mathrm{ddh}}. \tag{7}$$

G_3: We modify the game so that the function PRF_{K_0} is now replaced with a truly random function. That is, for each message M, the signer S_0's signature is now chosen at random from $\{0,1\}^\ell$ instead of being computed as $\mathrm{PRF}_{K_0}(M)$. We have

$$|\Pr[X_3] - \Pr[X_2]| \leq \epsilon_{\mathrm{prf}}. \tag{8}$$

To prove the equation above, we construct an algorithm $\mathcal{D}_2$ to $(t_2, \epsilon_{\mathrm{prf}})$-break the pseudorandomness of $\mathbf{F}$ with $t_2 \approx t$.

Given an oracle function $F(\cdot)$, $\mathcal{D}_2$ randomly chooses a group $(\mathbb{G}, g, q)$ and selects $a_0, a_1, c \in \mathbb{Z}_q$ and $K_1 \in \mathbb{G}$ uniformly. It invokes $\mathcal{D}$ on input $(\mathtt{pk}_{s_0}, \mathtt{pk}_{s_1}, \mathtt{pk}_v) = (g^{a_0}, g^{a_1}, g^c)$. $\mathcal{D}_2$ maintains a table T containing entries of the form (d, M, σ), which is initially empty.

Given a signing query (d, M), if there is a tuple (d, M, σ) in table T, $\mathcal{D}_2$ returns σ. Otherwise, if $d = 0$, $\mathcal{D}_2$ forwards M to F and obtains σ; if $d = 1$, $\mathcal{D}_2$ computes $\sigma \leftarrow \mathrm{PRF}_{K_1}(M)$. In either case, $\mathcal{D}_2$ stores (d, M, σ) in T and returns σ to $\mathcal{D}$. Given a verification query (d, M, σ), if it is contained in T, $\mathcal{D}_2$ returns 1. Otherwise, if $d = 0$, it submits M to F, obtains σ' and returns $\sigma' \stackrel{?}{=} \sigma$; if $d = 1$, it computes and returns $\sigma \stackrel{?}{=} \mathrm{PRF}_{K_1}(M)$. In either case, $\mathcal{D}_2$ stores (d, M, σ') in T.

When $\mathcal{D}$ submits its challenge message M^*, $\mathcal{D}_2$ tosses a coin b. If the output is 0, it sends M^* to F and obtains σ^*; if the outputs is 1, it computes $\sigma^* \leftarrow \mathrm{PRF}_{K_1}(M^*)$. $\mathcal{D}_2$ then returns σ^* to the adversary. All the successive queries are handled as above. Finally, $\mathcal{D}$ outputs b'. If $b' = b$, $\mathcal{D}_2$ outputs 1 indicating that F is a pseudorandom function chosen from $\mathbf{F}$; otherwise, it outputs 0 indicating that F is a truly random function.

Clearly, if F is chosen from $\mathbf{F}$, $\mathcal{D}_2$ perfectly simulated Game G_2 and $\mathcal{D}$ outputs the correct bit with probability $\Pr[X_2]$; if it is a truly random function, $\mathcal{D}_2$ perfectly simulated Game G_3 and $\mathcal{D}$ outputs the correct bit with probability $\Pr[X_3]$. Therefore, we obtain the Eq. (8).

G_4: In this game, the function $\mathrm{PRF}_{K_1}(\cdot)$ is also replaced with a truly random function. That is, the signer S_1's signatures are now chosen at random from $\{0,1\}^\ell$ as well. Similarly, we have that

$$|\Pr[X_4] - \Pr[X_3]| \leq \epsilon_{\mathrm{prf}}. \tag{9}$$

Note that in Game G_4 both of the signers' signatures are randomly chosen from $\{0,1\}^\ell$, including the challenge signature. Thus, the answers to all the queries issued by $\mathcal{D}$ provide no help to it. That is, $\mathcal{D}$ does not obtain any information about the bit b at all from its queries and the challenge signature. Hence, we have that

$$\Pr[X_4] = \frac{1}{2}. \tag{10}$$

Combining Eqs. (5)–(10), we obtain that

$$\begin{aligned}\Pr[X_0] &\leq \sum_{i=1}^{4} |\Pr[X_i] - \Pr[X_{i-1}]| + \Pr[X_4] \\ &\leq 2\epsilon_{\mathrm{ddh}} + 2\epsilon_{\mathrm{prf}} + \frac{1}{2}.\end{aligned}$$

This completes the proof. □

Naor–Reingold PRF [23] Let $IG(1^k)$ be an efficient algorithm that outputs (p, q, g) where p is a k-bit prime, q is a prime such that $p = 2q + 1$ and g is an element of order q in $\mathbb{Z}_p^*$. Let H_k be a family of pairwise independent hash functions from $\{0,1\}^k$ to $\{0,1\}^{\lfloor k/2 \rfloor}$. Define the function ensemble $\tilde{F} = \{\tilde{F}_k\}_{k \in \mathbb{N}}$. For every k, a key K of a function in $\tilde{F}_k$ consists of an output (p, q, g) of $IG(1^k)$, a random sequence $\mathbf{a} = (a_0, a_1, \ldots, a_k)$ of $k+1$ elements of $\mathbb{Z}_q$ and a hash function h from H_k. For any k-bit input $x = x_1 x_2, \ldots, x_n$, the function $\tilde{f}_K$ is defined by:

$$\tilde{f}_K \stackrel{\mathrm{def}}{=} h\left(g^{a_0 \prod_{i=1}^{k} a_i^{x_i}} \bmod p\right).$$

Naor and Reingold [23] proved that if DDH problem is hard, the function ensemble above is a pseudorandom function

ensemble. They also showed how to build a pseudorandom generator (PRG) from DDH assumption. Their PRG works as follows:

$$\mathrm{PRG}_{p,q,g,g^a}(b) = \left(g^b, g^{ab}\right),$$

where g is of prime order q and $a, b \in \mathbb{Z}_q$. The PRG doubles the length of the input. If we instantiate the pseudorandom function in SDVS_1 with Naor–Reingold PRF, we should include the description of a pseudorandom generator in the system parameter, which expands the Diffie–Hellman key of S and V to the randomness used for selecting a function in $\tilde{F}_k$. To do this, the signer (resp. verifier) applies a universal one-way hash function to $g^{x_s x_v}$, which maps to an element of $\mathbb{Z}_q$, and then applies Naor–Reingold PRG to the hash value. It then repeats the process above to the right half of the PRG's output, until it obtains enough (pseudo)random numbers for invoking the IG algorithm and for picking the random vector $\mathbf{a}$ for the PRF. The complexity of this process is $O(k)$. Therefore, we obtain the following corollary.

Corollary 1 *If DDH assumption holds,* SDVS_1 *is a secure strong designated verifier signature scheme in the standard model.*

5 SDVS with non-delegatability

5.1 The scheme

The SDVS_1 scheme above is *delegatable*: signer S or verifier V can simply release $K = g^{x_s x_v}$ to any third party so that it can sign messages on behalf of S for V. As explained in [22], this capability may not be desirable in some applications. In this section, we propose another SDVS construction, called SDVS_2, which is non-delegatable but at the price of provable security in the random oracle model only. The definition of non-delegatability (Definition 5) requires that the signature is a non-interactive proof of knowledge of the secret key of either S or V. Therefore, it is natural to use the Fiat–Shamir heuristic to convert an interactive proof of knowledge of the secret key of either S or V into a signature of knowledge. To make the scheme a **strong** DVS, we choose to include the common key shared between S and V, i.e. $K = \mathtt{pk}_v^{x_s} = \mathtt{pk}_s^{x_v} = g^{x_s x_v}$, into the message to be signed. Thus, the signature of knowledge becomes

$$\sigma = \mathrm{SPK}\left\{x : \mathtt{pk}_s = g^x \vee \mathtt{pk}_v = g^x\right\}(M \| K).$$

Let $\mathbb{G}$ be a multiplicative cyclic group of large prime order q, and g be its generator. Let $\mathtt{H} : \{0,1\}^* \times \mathbb{G}^4 \to \mathbb{Z}_q$ be a collision-resistant hash function, which will be modeled as a random oracle in the security proofs. Our scheme SDVS_2 is described in Fig. 2.

5.2 Security analysis

A signature in our scheme is a proof of knowledge of the secret key, which is the discrete logarithm of the corresponding public key. Using the Forking lemma [24], we can show that the unforgeability of SDVS_2 could be based on the discrete logarithm assumption. Regarding the PSI property, as the key $K = g^{x_s x_v}$ is included in the input to the hash function, which is the CDH solution to the public keys of S and V, we are able to show that the intractability of GDH problem tightly reduces to the PSI of SDVS_2. The non-delegatability stems from the fact that the signature is a (non-interactive) proof of knowledge of the secret key. The role of W in the input to the hash function is to prevent the verifier from releasing K to a third party so that it can verify signatures designated to the verifier. Formally, we have the following theorems.

Theorem 4 *If DL assumption* $(t_{\mathrm{dl}}, \epsilon_{\mathrm{dl}})$*-holds in* $\mathbb{G}$, SDVS_2 *is* $(t_{\mathrm{uf}}, q_{\mathtt{H}}, q_{\mathsf{Sign}}, q_{\mathsf{Sim}}, q_{\mathsf{Ver}}, \epsilon_{\mathrm{uf}})$*-strongly unforgeable in the random oracle model, where* $t_{\mathrm{uf}} \approx t_{\mathrm{dl}}$ *and* $\epsilon_{\mathrm{uf}} \leq \sqrt{32 q_{\mathtt{H}} \epsilon_{\mathrm{dl}}}$.

Proof Again, as shown in Theorem 5, SDVS_2 is perfectly non-transferable, we do not need to handle the adversary's queries to oracle $\mathcal{O}_{\mathsf{Sim}}$. It is sufficient for us to consider how to deal with signing queries, verification queries and hash queries in the proof.

Let $\mathcal{F}$ be a forger against the unforgeability of SDVS_2, which runs in time at most t_{uf}, makes at most $q_{\mathtt{H}}$ hash queries, q_{Sign} signing queries, q_{Sim} simulation queries and q_{Ver} verification queries and wins the unforgeability game with probability at least ϵ_{uf}. We use it as a subroutine to build another algorithm $\mathcal{A}$ for solving the DL problem.

Given a random instance of DL problem, i.e. $(\mathbb{G}, g, q, y)$, where $y = g^x$ for some unknown $x \in \mathbb{Z}_q$, $\mathcal{A}$ randomly chooses $\bar{x} \in \mathbb{Z}_q$ and tosses a coin. If the outcome is head, it sets $\mathtt{pk}_s := g^{\bar{x}}$ and $\mathtt{pk}_v := y$; otherwise, it sets $\mathtt{pk}_s := y$ and $\mathtt{pk}_v = g^{\bar{x}}$. For simplicity, we assume that the outcome is head. The rest simulation in the other case can be done similarly. $\mathcal{A}$ then invokes $\mathcal{F}$ on input $(\mathtt{pk}_s, \mathtt{pk}_v)$ and then simulates oracles for it. It maintains a hash table HT, which is initially empty.

Hash Queries Given a query (M, K, R_s, R_v), if there is a tuple $((M, K, R_s, R_v), c)$ in HT, $\mathcal{A}$ returns c; otherwise, it randomly selects a fresh $c \in \mathbb{Z}_q$, stores $((M, K, R_s, R_v), c)$ in HT and returns c.

Signing Queries On input a message M, $\mathcal{A}$ computes the answer by calling the Sign algorithm using its knowledge of $\bar{x}$. That is, $\mathcal{A}$ randomly selects $r_s, z_v, c_v, w \in \mathbb{Z}_q$ and computes $R_s \leftarrow g^{r_s}$, $R_v \leftarrow g^{z_v} \mathtt{pk}_v^{-c_v}$, $W \leftarrow g^w$ and $U \leftarrow \mathtt{pk}_v^w$. It sets $c \leftarrow \mathtt{H}(M, \mathtt{pk}_v^{\bar{x}}, R_s, R_v, U)$ and $c_s \leftarrow c - c_v$, $z_s \leftarrow r_s + c_s \bar{x}$. The signature $\sigma := (c_s, z_s, c_v, z_v, W)$ is then returned to $\mathcal{F}$.

$\mathsf{Kg}(1^k)$:
choose $x \leftarrow_{\$} \mathbb{Z}_q$
return $(\mathtt{pk}, \mathtt{sk}) := (g^x, x)$

$\mathsf{Ver}(\mathtt{sk}_v, \mathtt{pk}_s, \mathtt{pk}_v, M, \sigma)$:
parse σ as (c_s, z_s, c_v, z_v, W), $\mathtt{sk}_v$ as x_v
set $R_s \leftarrow g^{z_s}\mathtt{pk}_s^{-c_s}$, $R_v \leftarrow g^{z_v}\mathtt{pk}_v^{-c_v}$
return $c_s + c_v \stackrel{?}{=} \mathtt{H}(M, \mathtt{pk}_s^{x_v}, R_s, R_v, W^{x_v})$

$\mathsf{Sign}(\mathtt{sk}_s, \mathtt{pk}_s, \mathtt{pk}_v, M)$:
parse $\mathtt{sk}_s$ as x_s
select at random $r_s, z_v, c_v, w \leftarrow_{\$} \mathbb{Z}_q$
set $R_s \leftarrow g^{r_s}$, $R_v \leftarrow g^{z_v}\mathtt{pk}_v^{-c_v}$
set $W \leftarrow g^w$, $U \leftarrow \mathtt{pk}_v^w$
set $c \leftarrow \mathtt{H}(M, \mathtt{pk}_v^{x_s}, R_s, R_v, U)$
set $c_s \leftarrow c - c_v$, $z_s \leftarrow r_s + c_s x_s$
return $\sigma := (c_s, z_s, c_v, z_v, W)$

Fig. 2 Our non-delegatable SDVS scheme, SDVS_2

Verification Queries Given a message M and an alleged signature $\sigma := (c_s, z_s, c_v, z_v, W)$, $\mathcal{A}$ computes $R_s \leftarrow g^{z_s}\mathtt{pk}_s^{-c_s}$ and $R_v \leftarrow g^{z_v}\mathtt{pk}_v^{-c_v}$ and checks whether $c_s + c_v = \mathtt{H}(M, \mathtt{pk}_v^{\bar{x}}, R_s, R_v)$. If so, $\mathcal{A}$ returns 1; otherwise, it returns 0.

Finally, $\mathcal{F}$ outputs its forgery, i.e. (M^*, σ^*) where $\sigma^* = (c_s^*, z_s^*, c_v^*, z_v^*)$. If σ^* is not a valid signature of S on M^* for V, $\mathcal{A}$ aborts. Otherwise, $\mathcal{A}$ computes

$$R_s^* \leftarrow g^{z_s^*}\mathtt{pk}_s^{-c_s^*} \quad \text{and} \quad R_v^* \leftarrow g^{z_v^*}\mathtt{pk}_v^{-c_v^*}.$$

Let c^* be the hash value of $(M^*, \mathtt{pk}_v^{x_s}, R_s^*, R_v^*)$. $\mathcal{A}$ then rewinds $\mathcal{F}$ to the status of asking for the hash value of $(M^*, \mathtt{pk}_v^{x_s}, R_s^*, R_v^*)$ and feeds $\mathcal{F}$ with $\bar{c}^* \neq c^*$. The subsequent queries issued by $\mathcal{F}$ are answered as above. Again, we assume that $\mathcal{F}$ outputs another valid forgery on the same message, i.e. $(M^*, \bar{\sigma}^*)$ where $\bar{\sigma}^* = (\bar{c}_s^*, \bar{z}_s^*, \bar{c}_v^*, \bar{z}_v^*)$ such that $\bar{c}^* = \bar{c}_s^* + \bar{c}_v^* \neq c_s^* + c_v^* = c^*$ but $g^{\bar{z}_s^*}\mathtt{pk}_s^{-\bar{c}_s^*} = R_s^*$ and $g^{\bar{z}_v^*}\mathtt{pk}_v^{-\bar{c}_v^*} = R_v^*$. (Otherwise $\mathcal{A}$ aborts.) If $c_v^* \neq \bar{c}_v^*$, $\mathcal{A}$ outputs the discrete log of y as $x = \log_g y = (z_s^* - \bar{z}_s^*)/(c_s^* - \bar{c}_s^*)$. Otherwise, it aborts.

Probability Analysis The oracles are simulated by $\mathcal{A}$ perfectly using its knowledge of $\bar{x}$ (which is the secret key of either $\mathtt{pk}_s$ or $\mathtt{pk}_v$, depending on the outcome of the coin toss at the onset of the simulation). Therefore, $\mathcal{F}$ outputs its first valid forgery with probability at least ϵ_{uf}. Then, by an analysis similar to that in [7,24], we have that with probability at least $\epsilon_{\mathrm{uf}}^2/16q_{\mathtt{H}}\mathcal{F}$ outputs its valid forgery in the second run again which satisfies the aforementioned conditions. Since setting y as the public key of S or V (at the onset of the simulation by $\mathcal{A}$) is completely random and the whole simulation is perfect, we then have that with probability at least one-half it holds that $c_v^* \neq \bar{c}_v^*$. Thus, from the two valid forgeries, $\mathcal{A}$ can successfully compute the discrete log of y with probability at least

$$\epsilon_{\mathrm{dl}} \geq \frac{\epsilon_{\mathrm{uf}}^2}{16q_{\mathtt{H}}} \cdot \frac{1}{2} = \frac{\epsilon_{\mathrm{uf}}^2}{32q_{\mathtt{H}}}.$$

This completes the proof. □

Theorem 5 SDVS_2 *is perfectly non-transferable.*

To simulate the signer's signature on message M, the designated verifier first selects at random $r_v, z_s, c_s \leftarrow_{\$} \mathbb{Z}_q$ and computes $R_v \leftarrow g^{r_v}$, $R_s \leftarrow g^{z_s}\mathtt{pk}_s^{-c_s}$. It then sets $c \leftarrow \mathtt{H}(M, \mathtt{pk}_s^{x_v}, R_s, R_v)$ and $c_v \leftarrow c - c_s, z_v \leftarrow r_v + c_v x_v$. It outputs $\sigma := (c_s, z_s, c_v, z_v)$ as the simulated signature on M. Then, the theorem follows from the fact that both S and V can produce identically distributed proofs using their respective secret key, since they can compute the same key K and (the interactive version of) the proof of knowledge is perfect special honest verifier zero-knowledge.

Theorem 6 *If there is an algorithm $\mathcal{F}$ which for some message M can produce valid signatures in polynomial time and with probability ϵ after issuing at most $q_{\mathtt{H}}$ queries to the random oracle, then SDVS_2 is non-delegatable with knowledge error $1/q$ in the random oracle model.*

Proof Assume that $\epsilon > \kappa = 1/q$, where $1/q$ is the probability that the adversary guesses the value of $\mathtt{H}(M, g^{x_s x_v}, R_s, R_v)$ without asking the random oracle.

Let $\mathcal{F}_M$ be a forger with input message M. Consider two runs of $\mathcal{F}_M$ by $\mathcal{K}$ on the same random tape. In both runs, $\mathcal{K}$ executes $\mathcal{F}_M$ step-by-step, except that $\mathcal{K}$ returns different random values (c versus $\bar{c}$) as the answer to the hash query $\mathtt{H}(M, K, R_s, R_v)$ where $K = g^{x_s x_v}$. Since (K, R_s, R_v) are in the hash input, their values must be unchanged in the two runs. Let $R_s = g^{r_s}$ and $R_v = g^{r_v}$ for some $r_s, r_v \in \mathbb{Z}_q$. Suppose that both signatures are valid, it must be that

$$c_s + c_v = \mathtt{H}(M, K, g^{z_s}\mathtt{pk}_s^{-c_s}, g^{z_v}\mathtt{pk}_v^{-c_v}), \text{ and}$$
$$\bar{c}_s + \bar{c}_v = \mathtt{H}(M, K, g^{\bar{z}_s}\mathtt{pk}_s^{-\bar{c}_s}, g^{\bar{z}_v}\mathtt{pk}_v^{-\bar{c}_v}).$$

Then we have

$$z_s = r_s + x_s c_s, \tag{11}$$
$$z_v = r_v + x_v c_v, \tag{12}$$
$$\bar{z}_s = r_s + x_s \bar{c}_s, \tag{13}$$
$$\bar{z}_v = r_v + x_v \bar{c}_v. \tag{14}$$

If $c_s \neq \bar{c}_s$, then from Eqs. (11) and (13) we have that $x_s = (z_s - \bar{z}_s)/(c_s - \bar{c}_s)$. Similarly, if $c_v \neq \bar{c}_v$, from Eqs. (12) and (14), we have that $x_v = (z_v - \bar{z}_v)/(c_v - \bar{c}_v)$. Since $c_s + c_v \neq \bar{c}_s + \bar{c}_v$, it must be that either $c_s \neq \bar{c}_s$ or $c_v \neq \bar{c}_v$. Then, we can extract the secret key of either S or V as above.

Note that in the analysis above, the value of $K = g^{x_s x_v}$ is unknown to the extractor $\mathcal{K}$. Since the forger $\mathcal{F}_M$ may make many hash queries of the form $(M, \cdot, R_s, R_v)$, in order to rewind $\mathcal{F}_M$ to the proper status, i.e. the status that $\mathcal{F}_M$ made the hash query $(M, g^{x_s x_v}, R_s, R_v)$, $\mathcal{K}$ needs to guess

at random the hash query in which the second element is the correct value of K (and the last two elements are R_s and R_v). This brings a factor $1/q_{\mathrm{H}}$ to the success probability of the extractor. Conditioned on that the random guess is correct, by a similar analysis to that in [7,24], we have that with probability at least $(\epsilon - 1/q)/16$, $\mathcal{F}_M$ produces another valid signature σ' on the same message M in the second run, which together with the valid signature in the first run, enables $\mathcal{K}$ to extract either $\mathtt{sk}_s$ or $\mathtt{sk}_v$. Hence, the probability that $\mathcal{K}$ succeeds in extracting either $\mathtt{sk}_s$ or $\mathtt{sk}_v$ is at least $(\epsilon - 1/q)/16q_{\mathrm{H}}$. □

Theorem 7 *If GDH assumption* $(t_{\mathrm{gdh}}, \epsilon_{\mathrm{gdh}})$*-holds in* $\mathbb{G}$, SDVS_2 *is then* $(t, q_{\mathsf{Sign}}, q_{\mathsf{Sim}}, q_{\mathsf{Ver}}, \epsilon)$*-PSI-secure in the random oracle model, where* $t \approx t_{\mathrm{gdh}}$ *and* $\epsilon \leq \epsilon_{\mathrm{gdh}} + 2/q$.

Proof Assume that $\mathcal{D}$ is an adversary against the privacy of signer's identity, which runs in time t, issues at most q_{Sign} signing queries, q_{Sim} simulation queries and q_{Ver} verification queries and wins with probability $1/2 + \epsilon$. Using $\mathcal{D}$ as a subroutine, we build an algorithm $\mathcal{E}$ for solving the GDH problem.

Given a random instance of GDH problem, say, $(\mathbb{G}, g, q, g^u, g^w)$ and a DDH oracle, $\mathcal{E}$ randomly selects $u' \in \mathbb{Z}_q$ and sets $\mathtt{pk}_{s_0} = g^u$, $\mathtt{pk}_{s_1} = g^u g^{u'}$ and $\mathtt{pk}_v = g^w$. It invokes $\mathcal{D}$ on input $(\mathtt{pk}_{s_0}, \mathtt{pk}_{s_1}, \mathtt{pk}_v)$ and then simulates oracles for it. $\mathcal{E}$ maintains two tables, HT and ST, which are initially empty. For simplicity, we assume that $\mathcal{D}$ does not duplicate queries, and we do not consider queries to $\mathcal{O}_{\mathsf{Sim}}$ in the proof.

Hash Queries Given an input (M, K, R_s, R_v), if there is a tuple $((M, K, R_s, R_v), c)$ in table HT, $\mathcal{E}$ returns c. Otherwise, it submits $(g, \mathtt{pk}_{s_0}, \mathtt{pk}_v, K)$ and $(g, \mathtt{pk}_{s_1}, \mathtt{pk}_v, K)$ to the DDH oracle and obtains two bits, b_0 and b_1. If $b_0 = 1$, $\mathcal{E}$ outputs K and halts; if $b_1 = 1$, it outputs $K/(g^w)^{u'}$ and halts; otherwise ($b_0 = b_1 = 0$), it selects at random a fresh $c \in \mathbb{Z}_q$, stores $((M, K, R_s, R_v), c)$ in HT and returns c.

Signing Queries Given a bit d and a message M, $\mathcal{E}$ randomly chooses $c_s, c_v, z_s, z_v \in \mathbb{Z}_q$ and computes $R_s = g^{z_s}\mathtt{pk}_{s_d}^{-c_s}$, $R_v = g^{z_v}\mathtt{pk}_v^{-c_v}$ and $c = c_s + c_v$. If c was ever used, $\mathcal{E}$ repeats the process above. It then stores $((M, \perp, R_s, R_v), c)$ in HT and returns $\sigma := (c_s, z_s, c_v, z_v)$. Meanwhile, it stores (d, M, σ) into signature table ST. Note that though $\mathcal{E}$ does not know $g^{x_{s_d} x_v}$, the simulated signature is still identically distributed as real ones.

Verification Queries Given a bit d, a message M and an alleged signature $\sigma = (c_s, z_s, c_v, z_v)$, if there is a tuple (d, M, σ) in table ST, $\mathcal{E}$ returns 1; otherwise, it returns 0.

The intuition behind the simulation of verification oracle is that if a signature was ever returned by $\mathcal{E}$ to the adversary, it must be valid. Otherwise, if the signature is invalid, certainly the oracle should return 0; if it is valid, then the adversary forged a valid signature. In this case, with probability at least $1 - 1/q$ it holds that $\mathcal{D}$ made a hash query (M, K, R_s, R_v), where $K = \mathtt{pk}_{s_d}^{x_v}$, $R_s = g^{z_s}\mathtt{pk}_{s_d}^{-c_s}$ and $R_v = g^{z_v}\mathtt{pk}_v^{-c_v}$. According to the simulation of the hash oracle, $\mathcal{E}$ obtained g^{uw} and halted. Thus, the only case left is that the signature is valid but the adversary did not ask the random oracle for the hash of (M, K, R_s, R_v). Therefore, the simulation of the verification oracle would cause at most $1/q$ difference to the the adversary's success probability.

When it is ready, $\mathcal{D}$ submits a challenge message M^*. $\mathcal{E}$ then flips a coin b, randomly chooses $c_s^*, c_v^*, z_s^*, z_v^* \in \mathbb{Z}_q$ and computes $R_s^* = g^{z_s^*}\mathtt{pk}_{s_b}^{-c_s^*}$, $R_v^* = g^{z_v^*}\mathtt{pk}_v^{-c_v^*}$ and $c^* = c_s^* + c_v^*$. Again, if c^* was used, $\mathcal{E}$ repeats the process above. It then returns $\sigma^* := (c_s^*, z_s^*, c_v^*, z_v^*)$ and stores $((M^*, \perp, R_s^*, R_v^*), c^*)$ in HT and (b, M^*, σ^*) in ST, and $\mathcal{E}$ continues to simulate oracles for $\mathcal{D}$ and monitor the hash queries as above. Finally, $\mathcal{D}$ outputs a bit b'. If until now $\mathcal{E}$ has not halted, it aborts and fails to find g^{ab}.

Probability Analysis From above, we see that all the queries are perfectly handled, except there is a $1/q$ difference in the simulation of verification oracle. Notice that in order to win the PSI game, $\mathcal{D}$ has to make hash queries with (M, K, R_s, R_v) where $K = g^{x_{s_0} x_v}$ or $K = g^{x_{s_1} x_v}$. Otherwise, since the outputs of oracle $\mathtt{H}$ are completely random, $\mathcal{D}$ would have no information about c^* at all if it did not make a hash query on (M^*, K, R_s^*, R_v^*), except that it guesses the hash value at random, which is correct with probability at most $1/q$. Therefore, we have that

$$\begin{aligned}\epsilon_{\mathrm{gdh}} &\geq \left(1 - \frac{1}{q}\right)\left(\left(\frac{1}{2} + \epsilon\right) - \left(\frac{1}{2} + \frac{1}{q}\right)\right) \\ &= \left(1 - \frac{1}{q}\right)\left(\epsilon - \frac{1}{q}\right) > \epsilon - \frac{2}{q}.\end{aligned}$$

This completes the proof. □

Remark 3 The security reduction for unforgeability is not tight, since we need to rewind the adversary in order to extract the discrete logarithm from the signatures. We could also try to apply Katz–Wang technique [17] to obtain a tight reduction. However, the unforgeability would then rely on DDH assumption, which is stronger than the one we are currently using. Besides, the proof of PSI may still need the GDH assumption. In a consequence, the security of the new scheme relies on two mutually conflictive assumptions, e.g. DDH and GDH.

Remark 4 From the proof above, we can see that the underlying assumption of PSI can be further weakened to Strong Diffie–Hellman (Strong DH) assumption [1][2], since each query

[2] The Strong DH assumption states that it is intractable to compute g^{ab} given g, g^a, g^b for unknown $a, b \in \mathbb{Z}_p$ and access to a DDH oracle which takes as input (X, Z) outputs 1 if $Z = X^a$ and 0 otherwise. Notice that this is different from the one introduced by Boneh and Boyen [6] in constructing identity-based encryption schemes.

to the DDH oracle is of the form $(*, g^w, K)$ where the g^w is fixed in all the queries. The same holds for the short SDVS scheme in [5,15] as well.

6 Comparison

In Table 1, we compare our two SDVS schemes with some existing (S)DVS schemes. The third and the fourth rows of the table consider the size of a public key and that of a signature, respectively. The fifth and sixth rows count the numbers of exponentiations and pairings in signature generation and verification, respectively. The seventh row considers whether the scheme supports PSI. The eighth row considers whether the scheme has non-delegatability. The ninth row considers whether the security could be proved without random oracles. The last row shows the underlying number-theoretic assumptions for the security.

In terms of signature size, SDVS_1 that is secure in the standard model outperforms schemes in [16,20,22], which are secure in the random oracle model, and is comparable with [5,15] which by far is the most efficient SDVS in the random oracle model. SDVS_1 also has comparable computational complexity as [5,15]. On the other side, to the best of our knowledge, SDVS_2 is the first *non-delegatable* SDVS scheme, which compared with Lipmaa et al.'s DVS scheme [22], and has the same signature size and shorter public key, but looser reduction for unforgeability.

7 Conclusion and future work

In this paper, we proposed an efficient SDVS scheme based on pseudorandom functions. It has short signatures and is provably secure in the standard model. We then proposed another scheme, which is the first SDVS scheme with non-delegatability. The price we need to pay is the provable security in the random oracle model. Many problems about DVS are still open, for example, the construction of non-delegatable (S)DVS without random oracles. We leave it as our future work.

Acknowledgments We would like to thank the anonymous reviewers for their invaluable comments. Q. Huang and D.S. Wong were supported by a grant from CityU (Project No. 7002711). W. Susilo was supported by the Australian Research Council Future Fellowship FT0991397.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: CT-RSA, vol. 2020 of Lecture Notes in Computer Science, pp. 143–158. Springer (2001)
2. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Proceedings of Advances in Cryptology—CRYPTO 1992, vol. 740 of Lecture Notes in Computer Science, pp. 390–420. Springer (1992)
3. Bellare, M., Namprempre, C.: Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In: Proceedings of Advances in Cryptology—ASIACRYPT 2000, vol. 1976 of Lecture Notes in Computer Science, pp. 531–545. Springer (2000)
4. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73. ACM (1993)
5. Bhaskar, R., Herranz, J., Laguillaumie, F.: Aggregate designated verifier signatures and application to secure routing. Int. J. Secur. Netw. **2**(3/4), 192–201 (2007)
6. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Proceedings of Advances in Cryptology—EUROCRYPT 2004, vol. 3027 of Lecture Notes in Computer Science, pp. 56–73. Springer (2004a)
7. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Proceedings of Advances in Cryptology—CRYPTO 2004, vol. 3152 of Lecture Notes in Computer Science, pp. 41–55. Springer (2004b)
8. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. J. Cryptol. **17**(4), 297–319 (2004) A preliminary version appeared in Asiacrypt (2001)
9. Chaum, D., van Antwerpen, H.: Undeniable signatures. In: Proceedings of Advances in Cryptology—CRYPTO 1989, vol. 435 of Lecture Notes in Computer Science, pp. 212–216. Springer (1989)
10. Gentry, C., Silverberg, A.: Hierarchical id-based cryptography. In: Proceedings of Advances in Cryptology—ASIACRYPT 2002, vol. 2501 of Lecture Notes in Computer Science, pp. 548–566. Springer (2002)
11. Huang, Q., Susilo, W., Wong, D.S. (2009) Non-delegatable identity-based designated verifier signature. Cryptology ePrint Archive, Report 2009/367
12. Huang, X., Mu, Y., Susilo, W., Wu, W.: Provably secure pairing-based convertible undeniable signature with short signature length. In: Proceedings of 1st International Conference on Pairing-Based Cryptography, Pairing 2007, vol. 4575 of Lecture Notes in Computer Science, pp. 367–391. Springer (2007)
13. Huang, X., Susilo, W., Mu, Y., Wu, W.: Universal designated verifier signature without delegatability. In: Proceedings of 8th International Conference on Information and Communications Security, ICICS 2006, vol. 4307 of Lecture Notes in Computer Science, pp. 479–498. Springer (2006)
14. Huang, X., Susilo, W., Mu, Y., Wu, W.: Secure universal designated verifier signature without random oracles. Int. J. Inf. Secur. **7**(3), 171–183 (2007)
15. Huang, X., Susilo, W., Mu, Y., Zhang, F.: Short designated verifier signature scheme and its identity-based variant. Int. J. Netw. Secur. **6**(1), 82–93 (2008)
16. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Proceedings of Advances in Cryptology—EUROCRYPT 1996, vol. 1070 of Lecture Notes in Computer Science, pp. 143 – 154. Springer (1996)
17. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: ACM Conference on Computer and Communications Security, pp. 155–164. ACM (2003)
18. Laguillaumie, F., Libert, B., Quisquater, J.-J.: Universal designated verifier signatures without random oracles or non-black box assumptions. In: Proceedings of 5th International Conference on Security and Cryptography for Networks, SCN 2006, vol. 4116 of Lecture Notes in Computer Science, pp. 63–77. Springer (2006)
19. Laguillaumie, F., Vergnaud, D.: Designated verifier signatures: anonymity and efficient construction from any bilinear map.

In: Proceedings of 4th International Conference on Security in Communication Networks, SCN 2004, vol. 3352 of Lecture Notes in Computer Science, pp. 105–119. Springer (2004a)
20. Laguillaumie, F., Vergnaud, D.: Multi-designated verifiers signatures. In: Proceedings of 6th International Conference on Information and Communications Security, ICICS 2004, vol. 3269 of Lecture Notes in Computer Science, pp. 495–507. Springer (2004b)
21. Li, Y., Lipmaa, H., Pei, D.: On delegatability of four designated verifier signatures. In: Proceedings of 7th International Conference on Information and Communications Security, ICICS 2005, vol.e 3783 of Lecture Notes in Computer Science, pp. 61–71. Springer (2005)
22. Lipmaa, H., Wang, G., Bao, F.: Designated verifier signature schemes: Attacks, new security notions and a new construction. In: Proceedings of 32th International Colloquium on Automata, Languages and Programming, ICALP 2005, vol. 3580 of Lecture Notes in Computer Science, pp. 459–471. Springer (2005)
23. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. J. ACM **51**(2), 231–262 (2004)
24. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptol. **13**(3), 361–396 (2000)
25. Rivest, R., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd C. (ed.) Proceedings of Advances in Cryptology—ASIACRYPT 2001, vol. 2248 of Lecture Notes in Computer Science, pp. 552–565. Springer (2001)
26. Saeednia, S., Kremer, S., Markowitch, O.: An efficient strong designated verifier signature scheme. In: Proceedings of 6th International Conference on Information Security and Cryptology, ICISC 2003, vol. 2971 of Lecture Notes in Computer Science, pp. 40–54. Springer (2003)
27. Shacham, H., Waters, B.: Efficient ring signatures without random oracles. In: Okamoto, T., Wang, X. (eds.) Proceedings of Public Key Cryptography 2007, vol. 4450 of Lecture Notes in Computer Science, pp. 166–180. Springer (2007)
28. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J.: Universal designated-verifier signatures. In: Proceedings of Advances in Cryptology—ASIACRYPT 2003 vol. 2894 of Lecture Notes in Computer Science, pp. 523–542. Springer (2003)
29. Steinfeld, R., Wang, H., Pieprzyk, J.: Efficient extension of standard Schnorr/RSA signatures into universal designated-verifier signatures. In: Proceedings of Public Key Cryptography 2004, vol. 2947 of Lecture Notes in Computer Science, pp. 86–100. Springer (2004)
30. Susilo, W., Zhang, F., Mu, Y.: Identity-based strong designated verifier signature schemes. In: Proceedings of 9th Australasian Conference on Information Security and Privacy, ACISP 2004, vol. 3108 of Lecture Notes in Computer Science, pp. 313–324. Springer (2004)
31. Vergnaud, D.: New extensions of pairing-based signatures into universal designated verifier signatures. In: Proceedings of 33th International Colloquium on Automata, Languages and Programming, ICALP 2006, vol. 4052 of Lecture Notes in Computer Science, pp. 58–69. Springer (2006)
32. Wang, B., Song, Z.: A non-interactive deniable authentication scheme based on designated verifier proofs. Inf. Sci. **179**(6), 858–865 (2009)
33. Zhang, R., Furukawa, J., Imai, H.: Short signature and universal designated verifier signature without random oracles. In: Proceedings of 3rd International Conference on Applied Cryptography and Network Security, ACNS 2005, vol. 3531 of Lecture Notes in Computer Science, pp. 483–498. Springer (2005)