Short Designated Verifier Signature Scheme and Its Identity-based Variant

Xinyi Huang[1], Willy Susilo[1], Yi Mu[1], and Futai Zhang[2]
(Corresponding author: Xinyi Huang)

Centre for Computer and Information Security Research
School of Computer Science & Software Engineering(SCSSE), University of Wollongong
Northfields Avenu, Wollongong NSW 2522, Australia[1] (Email: xh068@uow.edu.au)
School of Mathematics and Computer Science, Nanjing Normal University, P.R. China[2]

(Received Feb. 2, 2006; revised and accepted Apr. 29, 2006)

Abstract

The notion of strong designated verifier signature was put forth by Jakobsson, Sako and Impagliazzo in 1996, but the formal definition was defined recently by Saeednia, Kremer and Markowitch in 2003 and revisited by Laguillaumie and Vergnaud in 2004. In this paper, we firstly propose the notion of *short* strong designated verifier signature scheme, and extend it to the *short* identity-based strong designated verifier scheme. Then, we propose the *first* construction of short strong designated verifier signature scheme. We also extend our scheme to construct a short identity-based strong designated verifier signature scheme. The size of the signature of our schemes is *the shortest* compared to any existing schemes reported in the literature. We provide formal security proofs for our schemes based on the random oracle model. Finally, we also discuss an extension of our scheme to construct a short strong designated verifier signature *without random oracle.*

Keywords: Designated verifier signature, identity based, random oracle model, short signature, strong designated verifier signature scheme

1 Introduction

The concept of undeniable signature was proposed by Chaum and van Antwerpen [3] to enable signers to have complete control over their signatures. In this scheme, the verification of the signer's signature requires the participation of the signer in an interactive protocol. The signer can reject invalid signatures, but she must not be able to deny valid signatures. Since the introduction of undeniable signature schemes, there have been a wide range of research covering a variety of different schemes for undeniable signatures. For example, the recent work in [13] studies the security of the FDH variant of undeniable signature schemes. Undeniable signatures have various applications in cryptography, such as in licensing softwares, auctions and electronic voting. However, these types of signature schemes do not always achieve their goal, because the signer does not know to whom he is proving the validity of a signature [4, 5].

To overcome this problem, Jakobsson, Sako and Impagliazzo proposed *designated-verifier signature* (DVS) schemes in [8]. This signature scheme is the first non-interactive undeniable signature scheme that transforms Chaum's scheme [2] into a non-interactive verification using a designated verifier proof. In a DVS scheme, the signature provides authentication of a message *without* providing a non-repudiation property of traditional signatures. A designated verifier scheme can be used to convince a single third party, i.e. the designated verifier, and only the designated verifier who can be convinced about its validity or invalidity of the signatures, due to the fact that the designated verifier can always construct a signature intended for himself that is indistinguishable from an original signature. There is no interaction with the presumed signer required in this type of signature schemes. More recently, following this idea, Galbraith and Mao proposed a non-interactive undeniable signature scheme based on RSA [6] in the multi-user setting to have *anonymity* and *invisibility.* Libert and Quisquater [11] proposed an identity-based undeniable signature scheme that can be regarded as the identity-based version of Galbraith-Mao's scheme using pairings.

In [16], Steinfeld *et al.* extended the notion of DVS scheme to a Universal Designated Verifier Signature (UDVS) scheme, that allows any signature holder to convert it into a DVS specified to any designated verifier of his choice. They also showed that bilinear maps allow an elegant construction of a UDVS scheme. Based on their ideas, Steinfeld *et al.* [17] proposed how to extend the classical Schnorr or RSA signature schemes into UDVS schemes and Laguillaumie and Vergnaud [9] proposed a generic construction of DVS scheme from *any* bilinear maps.

In [8], Jakobsson *et al.* also briefly discussed a stronger notion called a *strong designated verifier signature* (SDVS) scheme. The *strongness* property required in this notion refers to the requirement of the designated verifier to use his secret key to verify the validity or invalidity of a signature. Therefore, only given the public keys of the potential signers and verifier, nobody (except the designated verifier) can determine who is the actual signer. This notion was formally defined and strengthened by Saeednia *et al.* in [15] and strengthen by Laguillaumie and Vergnaud [9]. Given an SDVS signature and two potential signing public keys, it is *computationally infeasible* for an eavesdropper to determine under which of the two secret keys the signature was performed. Following Saeednia *et al.*'s work, Susilo *et al.* proposed an identity-based (or ID-based, for short) SDVS scheme based on pairings in [18].

The definition of the ring signatures was formalized by Rivest, Shamir and Tauman in [14]. They also showed how to achieve a designated verifier signature scheme where two participants in a ring signature collaborate and generate a signature. We should note that the construction does *not* satisfy the strongness property of SDVS scheme, since the secret key of the verifier is *not* required to verify the authenticity of the signature. Following this idea, multi-designated verifier signature scheme was proposed in [10].

Throughout this paper, let $log_2(n)$ denote the length of the binary representation of n. $p|q$ means p divides q. The ring of integers modulo a number p is denoted by $\mathbb{Z}_p$, and its multiplicative group, which contains only the integers relatively prime to p, by $\mathbb{Z}_p^*$. The notation $||$ means concatenation.

In this paper, we propose the *first* construction of short strong designated verifier signature scheme. Compared to the existing schemes [8, 9, 15], our scheme is *very efficient* in terms of *signature generation* and the *signature length*. In particular, the signature length of our scheme is *only* $log_2(q)$, which is the *shortest* compared to the existing schemes. At the same time, we *do not require* any pairing operation, in contrast to the construction proposed in [9]. We note that it is arguably that *shortness* of signature schemes is *not* a notion rather than a property of signature schemes.

We also extend our scheme to construct a short identity-based strong designated verifier signature scheme. In contrast to the previous construction in [18], our scheme produces a significantly shorter signature length, and it requires less computational operations. Our schemes are simple and efficient, but as we shall show in this paper, our schemes achieve all the required properties of (identity based) strong designated verifier signature schemes. We provide security proofs for our schemes based on the random oracle model. We also provide a variant of our scheme that produces signature length *less than* $log_2(q)$. Finally, we show a modification of our scheme to construct an SDVS scheme *without* random oracle.

The rest of this paper is organized as follows. In the next section, we will provide some preliminaries and background required throughout the paper. In Section 3, we review the definition of SDVS schemes, and their ID-based variant. In Section 4, we introduce the new notion of short strong designated verifier signature schemes and their identity-based variant. In Section 5, we provide our concrete short SDVS scheme, together with its security proof. In Section 6, we present our concrete short ID-based SDVS scheme and its security proof. We compare the performance of our schemes to other existing schemes in Section 7. We will also discuss variants of our scheme that produce *shorter* signature length and a scheme that is secure under the standard model. Finally, Section 8 concludes the paper.

2 Preliminaries

2.1 Basic Concepts on Bilinear Pairings

Let $\mathbb{G}_1$ be a cyclic additive groups generated by P, whose order is a prime q. Let $\mathbb{G}_T$ be a cyclic multiplicative group with the same order q. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ be a bilinear mapping with the following properties:

1) *Bilinearity*: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1, a, b, \in \mathbb{Z}_q$.

2) *Non-degeneracy*: There exists $P, Q \in \mathbb{G}_1$ such that $\hat{e}(P, Q) \neq 1_{\mathbb{G}_T}$.

3) *Computability*: There exists an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

Bilinear pairing instance generator is defined as a probabilistic polynomial time algorithm $\mathcal{IG}$ that takes as input a security parameter ℓ and returns a uniformly random tuple $param = (q, \mathbb{G}_1, \mathbb{G}_T, \hat{e}, P)$ of bilinear parameters, including a prime number q of size greater than ℓ ($q \geq 2^\ell$), a cyclic additive group $\mathbb{G}_1$ of order q, a multiplicative group $\mathbb{G}_T$ of order q, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ and a generator P of $\mathbb{G}_1$. For a group $\mathbb{G}$ of prime order, we denote the set $\mathbb{G}^* = \mathbb{G} \setminus \{\mathcal{O}\}$ where $\mathcal{O}$ is the identity element of the group.

The related complexity assumptions are as follows.

1) **Bilinear Diffie-Hellman (BDH) Problem.** Given a randomly chosen $P \in \mathbb{G}_1$, as well as aP, bP and cP (for unknown randomly chosen $a, b, c \in \mathbb{Z}_q^*$), compute $\hat{e}(P, P)^{abc}$. For the BDH problem to be hard, $\mathbb{G}_1$ and $\mathbb{G}_T$ must be chosen so that there is no known algorithm for efficiently solving the Diffie-Hellman problem in either $\mathbb{G}_1$ or $\mathbb{G}_M$. We note that if the BDH problem is hard for a pairing $\hat{e}$, then it follows that $\hat{e}$ is non-degenerate.

2) **Decisional Bilinear Diffie-Hellman (DBDH) Problem.** Given a randomly chosen $P \in \mathbb{G}_1$, as well as aP, bP, cP (for unknown randomly chosen $a, b, c \in \mathbb{Z}_q^*$) and $h \in \mathbb{G}_M$, decide whether $h = \hat{e}(P, P)^{abc}$.

3) **Gap Bilinear Diffie-Hellman (Gap BDH) Problem.** Given a randomly chosen $P \in \mathbb{G}_1$, as well as aP, bP and cP (for unknown randomly chosen $a, b, c \in \mathbb{Z}_q^*$), compute $\hat{e}(P, P)^{abc}$ with the help of the DBDH oracle.

4) **Computational Diffie-Hellman (CDH) Problem.** Given a randomly chosen $P \in \mathbb{G}_1$, as well as aP, bP, for unknown $a, b \in \mathbb{Z}_q^*$, compute abP.

5) **Decisional Diffie-Hellman (DDH) Problem.** Given a randomly chosen $P \in \mathbb{G}_1$, aP, bP and $h \in \mathbb{G}_1$, decide whether $h \overset{?}{=} abP$. It is known that DDH in $\mathbb{G}_1$ is easy and can be solved in polynomial time.

6) **Gap Diffie-Hellman (GDH) Problem.**
A prime order group $\mathbb{G}_1$ is a GDH group if there exists an efficient polynomial-time algorithm that solves the DDH problem in $\mathbb{G}_1$ and there is no probabilistic polynomial-time algorithm that solves the CDH problem with non-negligible probability. The Diffie-Hellman problem on such a group is called *Gap Diffie-Hellman Problem*, that states given a randomly chosen generator g, and g^a, g^b for unknown $a, b \in \mathbb{Z}_q^*$, compute g^{ab} with the help of the DDH oracle.

2.2 Collision-resistant Hashing

Let H be a hash function and an algorithm A has advantage ϵ in finding the collision of H if $\Pr[A = (m_0, m_1) : m_0 \neq m_1, H(m_0) = H(m_1)] = \epsilon$. We say H is a collision-resistant hashing if the probability ϵ is negligible for any polynomially bounded algorithm $\mathcal{A}$.

3 Review of (ID-based) Strong Designated Verifier Signatures Schemes

The notion of SDVS schemes was briefly mentioned in [8]. They suggested that "in order to make protocols *strong* designated verifier, transcripts can be probabilistically encrypted using the public key of the intended verifier". This notion has recently relaxed in [9] that proved that using an additional `IND-CCA2` public key encryption layer is actually sufficient to create any DVS scheme strong. This "generic" construction implies that the notion of SDVS schemes relies on the existence of an additional `IND-CCA2` public key encryption scheme.

In [15], the notion of SDVS schemes was formalized by requiring anyone to produce identically distributed transcripts that are indistinguishable from the original transcript that was produced by the original signer. For completeness, we review their definition as follows. We will provide the formal model of SDVS schemes in Section 4.

Definition 1. Strong Designated Verifier Signatures (SDVS) *[15]*
Let $P(A, B)$ be a protocol for Alice to prove the correctness of statement Θ to a designated verifier, Bob. $P(A, B)$ is called to be a strong designated verifier protocol if anyone *other than Bob can produce identically distributed transcripts that are indistinguishable from those of $P(A, B)$ for everyone, except for Bob.*

To achieve SDVS schemes, the general assumption is the verifier has constructed his public key and made it available publicly (with a certificate from the trusted authority to make sure the authenticity of the public key). That means, before a signature can be designated to the designated verifier, he has to setup his public key accordingly. Without this setup, a signer cannot designate her signature to the designated verifier.

Recently, this notion has been extended to an ID-based SDVS scheme in [18]. The new notion was introduced to reduce the restriction of having the designated verifier's public key setup before the signature can be designated by the signer. Hence, the signer can always designate her signature to anyone in the system without the need of any interaction with the receiver beforehand. The only requirement is to have the ID of the receiver published. The notion of ID-based SDVS schemes from [18] is as follows.

Definition 2. ID-based Strong Designated Verifier Signatures (ID-based SDVS) *[18]*
Let $P(A, B)$ be a protocol for Alice to prove the truth of the statement Θ to Bob. We require that Alice can just obtain Bob's ID from the system and use this information to show the correctness of Θ to Bob. $P(A, B)$ is said to be an ID-based strong designated verifier protocol if anyone can produce identically distributed transcripts that are indistinguishable from those of $P(A, B)$ for everyone, except for Bob.

Although they provide a generic construction of ID-based SDVS schemes in [18], their main scheme is essentially an ID-based version of Saeednia *et al.*'s scheme [15].

4 Short (ID-based) Strong Designated Verifier Signatures Schemes

In this section, we present our new notion of short SDVS schemes (SSDVS) (as noted earlier, we stress that it is arguably that *shortness* of signature schemes is *not* a notion rather than a property of signature schemes). We also extend our notion to its ID-based variant to create the short ID-based SDVS scheme (SIDSDVS).

4.1 Short Strong Designated Verifier Signatures

There exist two participants in the system, namely Alice and Bob, who act as the sender and the receiver (or the designated verifier), respectively. We assume that both

Alice and Bob have setup their public key setting, and we denote $\mathcal{SK}_i, \mathcal{PK}_i$ as a pair of secret key and public key for user i, where $i \in \{A, B\}$. An SSDVS scheme consists of three essential algorithms as follows.

- `Signature Generation`: A deterministic algorithm that uses the signer's secret key, the designated verifier's public key and a message m to generate a signature σ. That is, $\sigma \leftarrow \mathsf{Sign}(\mathcal{SK}_A, \mathcal{PK}_B, m)$.
- `Signature Verification`: A deterministic algorithm that accepts a message m, a signature σ, the signer's public key $\mathcal{PK}_A$ and a secret key $\mathcal{SK}_B$ and returns True if the signature is correct, or $\perp$ otherwise. That is, $\{\mathsf{True}, \perp\} \leftarrow \mathsf{Verify}(\mathcal{PK}_A, \mathcal{SK}_B, m, \sigma)$.
- `Transcript Simulation`: An algorithm that is run by the verifier to produce identically distributed transcripts that are *indistinguishable* from the original protocol.

In addition to the above main algorithms, we also require the following.

- **Correctness.** All signatures that are generated correctly by Sign algorithm, will always pass the verification algorithm. That is, $\Pr(True \leftarrow \mathsf{Verify}(\mathcal{PK}_A, \mathcal{SK}_B, m, \mathsf{Sign}(\mathcal{SK}_A, \mathcal{PK}_B, m))) = 1$.
- **Transcript Simulation Generation.** We require that the verifier, who holds the secret key $\mathcal{SK}_B$ can always produce identically distributed transcripts that are indistinguishable from the original protocol via the `Transcript Simulation` algorithm.

Unforgeability of SSDVS:
We provide a formal definition of existential forgeability of an SSDVS scheme under a chosen message attack (*EF-CMA*). It is defined using the following games between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- Setup: Input the security parameter ℓ, $\mathcal{C}$ runs the algorithm to obtain the secret key and public key pair $(\mathcal{SK}_A, \mathcal{PK}_A)$, $(\mathcal{SK}_B, \mathcal{PK}_B)$ to represent the signer, A, and the receiver, B, respectively.
- Sign Queries: $\mathcal{A}$ can request a signature on a message m for the signer A, and the designated verifier B. In respond, $\mathcal{C}$ outputs a signature σ for a message m.
- Verify Queries: $\mathcal{A}$ can request a signature verification on a pair (m, σ) for signer A, and the designated verifier B. In respond, $\mathcal{C}$ outputs True if it is correct, or $\perp$ otherwise.
- Output: Finally, $\mathcal{A}$ outputs a new pair (m^*, σ^*), where m^* has never been queried during the Sign Queries and σ^* is a valid signature.

The success probability of an adversary to win the game is defined by

$$Succ^{EF-CMA}_{SSDVS,\mathcal{A}}(\ell).$$

Definition 3. *We say that an SSDVS scheme is EF-CMA secure under a chosen message attack if the probability of success of any polynomially bounded adversary in the above game is negligible. In other words,*

$$Succ^{EF-CMA}_{SSDVS,\mathcal{A}}(\ell) \leq \epsilon.$$

Privacy of Signer's Identity:
We give a formal definition of the privacy of signer's identity (or the *strongness* property of the DVS scheme) under a chosen message attack (*PSI-CMA*) defined using the following games between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- Setup: Input a security parameter ℓ, $\mathcal{C}$ runs the algorithm to obtain the secret key and public key pair $(\mathcal{SK}_{A_0}, \mathcal{PK}_{A_0})$, $(\mathcal{SK}_{A_1}, \mathcal{PK}_{A_1})$, $(\mathcal{SK}_B, \mathcal{PK}_B)$ to represent the two signers, A_0, A_1, and the receiver, B, respectively.
- Sign Queries: $\mathcal{A}$ can request a signature on a message m for the signer A_r ($r \in \{0, 1\}$), and the designated verifier B. In respond, $\mathcal{C}$ outputs a signature σ for a message m.
- Verify Queries: $\mathcal{A}$ can request a signature verification on a pair (m, σ) for signer A_r, $r \in \{0, 1\}$, and the designated verifier B. In respond, $\mathcal{C}$ outputs True if it is correct, or $\perp$ otherwise.
- Output: Finally, $\mathcal{A}$ outputs a target message $m^* \in \{0, 1\}^*$ and provides it to $\mathcal{C}$. $\mathcal{C}$ performs a coin tossing to select a random $r \in \{0, 1\}$ and computes $\sigma^* \leftarrow \mathsf{Sign}(\mathcal{SK}_r, \mathcal{PK}_B, m)$ and returns it to $\mathcal{A}$. $\mathcal{A}$ must perform an educated guess to obtain the correct value of r. During this time, $\mathcal{A}$ can request Sign and Verify queries to $\mathcal{C}$, for any $m \neq m^*$ and $\sigma \neq \sigma^*$. After all the queries, $\mathcal{A}$ outputs a bit r'.

The Advantage of an *PSI-CMA* adversary $\mathcal{A}$ to win the game is defined by

$$Adv^{PSI-CMA}_{SSDVS,\mathcal{A}}(\ell) = |2\Pr[r' = r] - 1|.$$

Definition 4. *We say that an SSDVS scheme is a strong DVS scheme under a chosen message attack if the advantage of any polynomially bounded adversary in the above game is negligible. In other words,*

$$Adv^{PSI-CMA}_{SSDVS,\mathcal{A}}(\ell) \leq \epsilon.$$

4.2 Short ID-based Strong Designated Verifier Schemes

The notion of short ID-based SDVS scheme is similar to the definition of the short SDVS scheme defined in section 4.1. There exist two participants in the system, namely Alice and Bob, who act as the sender and the receiver (or the designated verifier), respectively. All participants have their identity, ID_i, $i \in \{A, B\}$, published. There exists a Private Key Generator (PKG) in the system that will provide a secret key to a participant during an Extract algorithm. Each participant's secret key is denoted by $\mathcal{SK}_{\mathsf{ID}_i}$, which is derived from the Extract algorithm. Without losing generality, we assume that A is the signer and B is the designated verifier. A short SIDSDVS scheme (SIDSDVS) consists of the following algorithms.

- `Signature Generation.` A deterministic algorithm that uses the signer's secret key, the ID of the designated verifier and a message m to generate a signature σ. That is, $\sigma \leftarrow \mathsf{IDSign}(\mathsf{ID}_B, \mathcal{SK}_{\mathsf{ID}_A}, m)$.

- `Signature Verification.` A deterministic algorithm that receives a message m, a signature σ, a verifier's secret key $\mathcal{SK}_{\mathsf{ID}_B}$ and a sender's identity ID_A, that returns `True` if the signature is correct, or $\perp$ otherwise. That is, $\{True, \perp\} \leftarrow \mathsf{IDVerify}(\mathsf{ID}_A, \mathcal{SK}_{\mathsf{ID}_B}, m, \sigma)$.

- `Transcript Simulation.` An algorithm that is run by the verifier to produce identically distributed transcripts that are *indistinguishable* from the original protocol.

In addition to the above main algorithms, we also require the following.

- **Correctness.** All signatures that are generated correctly by IDSign algorithm, will always pass the verification algorithm. That is, $\Pr(True \leftarrow \mathsf{IDVerify}(\mathsf{ID}_A, \mathcal{SK}_{\mathsf{ID}_B}, m, \mathsf{IDSign}(\mathsf{ID}_B, \mathcal{SK}_{\mathsf{ID}_A}, m))) = 1$.

- **Transcript Simulation Generation.** We require that the verifier, who holds the secret key $\mathcal{SK}_{\mathsf{ID}_B}$ can always produce identically distributed transcripts that are indistinguishable from the original protocol via the `Transcript Simulation` algorithm.

Unforgeability of SIDSDVS:
We provide a formal definition of existential unforgeability of a short SIDSDVS scheme (SIDSDVS) under a chosen message attack. It is defined using the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- Setup: $\mathcal{C}$ runs this algorithm to generate system's parameter.

- Key Extraction Queries: $\mathcal{A}$ can request the secret key of the user with the identity ID_i. $\mathcal{C}$ will return $\mathcal{SK}_{\mathsf{ID}_i}$ as the answer.

- IDSign Queries: $\mathcal{A}$ can request a signature on a message m for a signer ID_i, and the designated verifier ID_j. In respond, $\mathcal{C}$ outputs a signature σ for a message m.

- IDVerify Queries: $\mathcal{A}$ can request a signature verification on a pair (m, σ) for signer ID_i, and the designated verifier ID_j. In respond, $\mathcal{C}$ outputs True if it is correct, or $\perp$ otherwise.

- Output: Finally, $\mathcal{A}$ outputs a new pair (m^*, σ^*) with the signer ID_{i^*} and the designated verifier ID_{j^*} such that:

  1) $\mathcal{A}$ didn't submit ID_{i^*} or ID_{j^*} during the Key Extraction Queries.
  2) m^* has never been queried during the IDSign Queries with the signer ID_{i^*} and the designated verifier ID_{j^*}
  3) σ^* is a valid signature of the message m^* with the signer ID_{i^*} and the designated verifier ID_{j^*}.

The success probability of an adversary to win the game is defined by

$$Succ^{EF-CMA}_{SIDSDVS,\mathcal{A}}(\ell).$$

Definition 5. *We say that a short SIDSDVS scheme (SIDSDVS) is existentially unforgeable under a chosen message attack if the probability of success of any polynomially bounded adversary in the above game is negligible. In other words,*

$$Succ^{EF-CMA}_{SIDSDVS,\mathcal{A}}(\ell) \leq \epsilon.$$

Privacy of Signer's Identity
We give a formal definition of the privacy of signer's identity (or the *strongness* property of the Identity-based DVS scheme) under a chosen message attack (*PSI-CMA*) defined using the following games between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- Setup: $\mathcal{C}$ runs this algorithm to generate system's parameter.

- Key Extraction Queries: $\mathcal{A}$ can request the secret key of the user with the identity ID_i. $\mathcal{C}$ will return $\mathcal{SK}_{\mathsf{ID}_i}$ as the answer.

- IDSign Queries: $\mathcal{A}$ can request a signature on a message m for a signer ID_i, and the designated verifier ID_j. In respond, $\mathcal{C}$ outputs a signature σ for a message m.

- IDVerify Queries: $\mathcal{A}$ can request a signature verification on a pair (m, σ) for signer ID_i, and the designated verifier ID_j. In respond, $\mathcal{C}$ outputs True if it is correct, or $\perp$ otherwise.

- Output: Finally, $\mathcal{A}$ outputs a target message $m^* \in \{0,1\}^*$ with the possible signer $\mathsf{ID}_{A_0}, \mathsf{ID}_{A_1}$ and designated verifier ID_B to $\mathcal{C}$. The requirements are that

  1) $\mathcal{A}$ didn't submit $\mathsf{ID}_{A_0}, \mathsf{ID}_{A_1}$ or ID_B during the Key Extraction Queries.
  2) m^* has never been queried during the IDSign Queries with the signer $\mathsf{ID}_{A_r}, r \in \{0,1\}$ and the designated verifier ID_B.

  $\mathcal{C}$ performs a coin tossing to select a random r and computes $\sigma^* \leftarrow \mathsf{IDSign}(\mathsf{ID}_B, \mathcal{SK}_{\mathsf{ID}_{A_r}}, m^*)$ and returns it to $\mathcal{A}$. $\mathcal{A}$ must perform an educated guess to the correct value of r. During this time, $\mathcal{A}$ can request IDSign and IDVerify queries to $\mathcal{C}$ with the requirements that

1) m^* can't be queried during the IDSign Queries with the signer $\mathsf{ID}_{A_r}, r \in \{0,1\}$ and the designated verifier ID_B.
2) (m^*, σ^*) can't be queried during the IDVerify Queries with the signer $\mathsf{ID}_{A_r}, r \in \{0,1\}$ and the designated verifier ID_B.

The Advantage of an *PSI-CMA* adversary $\mathcal{A}$ to win the game in SIDSDVS is defined by

$$Adv^{PSI-CMA}_{SIDSDVS,\mathcal{A}}(\ell) = |2\Pr[r' = r] - 1|.$$

Definition 6. *We say that a short SIDSDVS scheme (SIDSDVS) is a strong IDDVS scheme under a chosen message attack if the advantage of any polynomially bounded adversary in the above game is negligible. In other words,*

$$Adv^{PSI-CMA}_{SIDSDVS,\mathcal{A}}(\ell) \le \epsilon.$$

5 A Concrete Short SDVS Scheme

In this section, we present our concrete construction of a short SDVS scheme. The description of our scheme is as follows.

- Setup: Let p be a large prime and q a prime divisor of $p-1$. Let g be an element in $\mathbb{Z}_p^*$ of order q. Let $p, q \ge 2^\ell$, where ℓ is the security parameter. Let $H : \{0,1\}^\infty \to \mathbb{Z}_q^*$ be a cryptographic collision-resistant hash function. The signer, Alice, randomly selects her secret key $x_A \in \mathbb{Z}_q^*$ and computes her public key as $y_A = g^{x_A} \pmod p$. Here, $\mathcal{PK}_A := y_A$. The designated verifier, Bob, also selects his secret key $x_B \in \mathbb{Z}_q^*$ and sets his public key as $y_B = g^{x_B} \pmod p$. Here $\mathcal{PK}_B := y_B$. Let $\mathcal{SK}_A := x_A$ and $\mathcal{SK}_B := x_B$.

- Sign: To sign a message $m \in \{0,1\}^\infty$ for Bob, Alice performs the following.

  – Compute $k = y_B^{x_A} \pmod p$.

  – Compute $\sigma = H(m||k)$.

  The signature on a message m is σ.

- Verify: To verify the validity of a signature σ on a message m, Bob computes $k = y_A^{x_B} \pmod p$ and tests whether

  $$H(m||k) \overset{?}{=} \sigma$$

  holds with equality. If it holds, then output True. Otherwise, output $\perp$.

- Transcript Simulation: Bob can produce the signature $\hat{\sigma}$ intended for himself, by performing the following.

  – Compute $k = y_A^{x_B} \pmod p$.

  – Compute $\hat{\sigma} = H(m||k)$.

  Note that the signature is indistinguishable from the original signature created by Alice.

Correctness:
The correctness of the verification algorithm is due to the following.

$$\sigma = H(m||y_A^{x_B}) = H(m||g^{ab}) = H(m||y_B^{x_A}).$$

We omit the modulus operation when it is clear from the context. □

Remarks:
We note that at a glance, the proposed scheme is similar to the notion of keying hash function as introduced in [1]. In a keying hash function, the key is a random and secure value known only to the parties. In our scheme, each party can use her/his secret key to obtain the value of k. One can also find that if the value k is published by the signer A, anyone can obtain a valid signature and B will *not* believe that the signature is sent by A. However, as explained earlier, the purpose of our scheme is that the signer A uses the designated verifier signature to convince B, and only B will believe with the validity of the signature. In this sense, A will not publish the value k and the attack model proposed in [12] (which is related to the *delegation* of the signature) is not applicable to this model. We also need to point out that the scheme proposed in [12] is *not* a strong DVS scheme rather than a DVS scheme, since it does not provide the signer's privacy.

5.1 Security Analysis

Theorem 1. *Our SSDVS scheme is a designated verifier signature scheme.*

Proof. We note that the verification algorithm requires $y_A^{x_B}$, where x_B is the secret key of the designated verifier B. Hence, B can always "simulate" a valid signature by producing a valid signature himself. This is achieved by constructing a signature σ, where $\sigma = H(m, y_A^{x_B})$. Note that the signature produced by B is indistinguishable from the one that was produced by A. Hence, no third party can be convinced with the validity or invalidity of this signature other than the designated verifier himself. If the designated verifier has not generated such a signature, then he will believe that the signature was indeed generated by the signer A. □

Theorem 2. (Unforgeability) *Let $\mathcal{A}$ be an* EF-CMA-adversary *against our SSDVS scheme with success probability $Succ^{EF-CMA}_{SSDVS,\mathcal{A}}$. In time t he can make q_H queries to the $H : \{0,1\}^* \to \mathbb{Z}_q^*$ ($q \ge 2^\ell$, ℓ is the system's security parameter), q_S queries to the signing algorithm and q_V to the verifying algorithm, then there exists an algorithm $\mathcal{B}$*

who can use $\mathcal{A}$ to solve an instance of the GDH problem with the probability:

$$Succ_{\mathcal{B}}^{GDH} \geq Succ_{SSDVS,\,\mathcal{A}}^{EF-CMA} - \frac{q_V}{2^\ell - q_H - q_S}.$$

Proof. See Appendix. □

Theorem 3. (Privacy of Signer's Identity) *Let $\mathcal{A}$ be an* PSI-CMA-adversary *against our SSDVS scheme with advantage $Adv_{SSDVS,\,\mathcal{A}}^{PSI-CMA}$. In time t he can make q_H queries to the $H : \{0,1\}^* \to \mathbb{Z}_q^*$ ($q \geq 2^\ell$, ℓ is the system's security number) , q_S queries to the signing algorithm and q_V to the verifying algorithm, then there exists an algorithm $\mathcal{B}$ who can use $\mathcal{A}$ to solve an instance of the GDH problem with the probability:*

$$Succ_{\mathcal{B}}^{GDH} \geq \frac{1}{2}\,Adv_{SSDVS,\,\mathcal{A}}^{PSI-CMA} - \frac{q_V}{2^\ell - q_H - q_S}.$$

Proof. See Appendix. □

6 A Concrete Short SIDSDVS Scheme

In this section, we present our concrete construction of short SIDSDVS scheme (SIDSDVS). The SIDSDVS scheme consists of the following algorithms.

- Setup: PKG generates two groups $(\mathbb{G}_1, +)$ and $(\mathbb{G}_T, \cdot)$ of prime order q ($q \geq 2^\ell$, ℓ is the system's security number) and a bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, together with an arbitrary generator $P \in \mathbb{G}_1$. He also selects his secret key (or master key) $s \in \mathbb{Z}_q^*$ and set $P_{pub} = sP$. Finally, two cryptographically collision-resistant hash functions are selected, $H_0 : \{0,1\}^\infty \to \mathbb{G}_1$ and $H_1 : \{0,1\}^\infty \to \mathbb{Z}_q^*$. The system parameters and their descriptions are $(\mathbb{G}_1, \mathbb{G}_T, q, \hat{e}, P, P_{pub}, H_0, H_1)$. Each user has his/her identity, ID_i, published. In this scenario, Alice has published her identity, ID_A, and Bob has published his identity, ID_B. Let $\mathsf{Q}_{\mathsf{ID}} = H_0(\mathsf{ID})$ and the user's secret key is computed as $\mathcal{S}_{ID} = s\mathsf{Q}_{\mathsf{ID}}$.
- IDSign: To sign a message $m \in \{0,1\}^\infty$ for Bob, Alice performs the following.
  - Compute $k = \hat{e}(\mathsf{Q}_{ID_B}, \mathcal{S}_{ID_A})$.
  - Compute $\sigma = H_1(m||k)$.

  The signature on a message m is σ.
- IDVerify: To verify the validity of a signature σ on a message m, Bob tests whether

  $$H_1(m||\hat{e}(\mathsf{Q}_{ID_A}, \mathcal{S}_{ID_B})) \stackrel{?}{=} \sigma$$

  holds with equality. If it does, then output True. Otherwise, output $\perp$.
- Transcript Simulation: Bob can produce the signature $\hat{\sigma}$ intended for himself for a message of his choice. This is done as follows.
  - Compute $k = \hat{e}(\mathsf{Q}_{ID_A}, \mathcal{S}_{ID_B})$.
  - Compute $\sigma = H_1(m||k)$.

  We note that the resulting signature is indistinguishable from the original one that was produced by Alice.

Correctness:
The correctness of the verification algorithm is due to the following.

$$\begin{aligned}\sigma &= H_1(m||\hat{e}(\mathsf{Q}_{ID_B}, \mathcal{S}_{ID_A}))\\ &= H_1(m||\hat{e}(\mathsf{Q}_{ID_B}, s\mathsf{Q}_{ID_A}))\\ &= H_1(m||\hat{e}(s\mathsf{Q}_{ID_B}, \mathsf{Q}_{ID_A}))\\ &= H_1(m||\hat{e}(\mathcal{S}_{ID_B}, \mathsf{Q}_{ID_A}))\\ &= H_1(m||\hat{e}(\mathsf{Q}_{ID_A}, \mathcal{S}_{ID_B})).\end{aligned}$$

We note that $\hat{e}(\mathsf{Q}_{ID_B}, \mathcal{S}_{ID_A}) = \hat{e}(\mathcal{S}_{ID_B}, \mathsf{Q}_{ID_A})$ because of the properties of bilinear pairings. □

6.1 Security Analysis

Using the same technique as we used in Section 5.1, we obtain the following theorems.

Theorem 4. *Our SIDSDVS is a designated verifier signature scheme.*

Theorem 5. (Unforgeability) *Let $\mathcal{A}$ be an* EF-CMA-adversary *against our SIDSDVS scheme with success probability $Succ_{SIDSDVS,\,\mathcal{A}}^{EF-CMA}$. In time t he can make q_H queries to the $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$ ($q \geq 2^\ell$, ℓ is the system's security parameter), q_S queries to the signing algorithm and q_V to the verifying algorithm, then there exists $\mathcal{B}$ who can use $\mathcal{A}$ to solve an instance of the Gap BDH problem with probability:*

$$Succ_{\mathcal{B}}^{Gap\ BDH} \geq Succ_{SIDSDVS,\,\mathcal{A}}^{EF-CMA} - \frac{q_V}{2^\ell - q_H - q_S}.$$

Theorem 6. (Privacy of Signer's Identity) *Let $\mathcal{A}$ be an* PSI-CMA-adversary *against our SIDSDVS scheme with advantage $Adv_{SIDSDVS,\,\mathcal{A}}^{PSI-CMA}$. In time t he can make q_H queries to the $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$ ($q \geq 2^\ell$, ℓ is the system's security number) , q_S queries to the signing algorithm and q_V to the verifying algorithm, then there exists $\mathcal{B}$ who can use $\mathcal{A}$ to solve an instance of the Gap BDH problem with probability:*

$$Succ_{\mathcal{B}}^{Gap\ BDH} \geq \frac{1}{2}\,Adv_{SIDSDVS,\,\mathcal{A}}^{PSI-CMA} - \frac{q_V}{2^\ell - q_H - q_S}.$$

7 Efficiency Comparison

In this section, we compare efficiency of the proposed schemes with the existing schemes. We will compare the efficiency of the schemes based on the length of the signature and the number of operations required.

In Table 1, we compare our scheme against the most efficient strong DVS scheme proposed in [9] and [15]. In this comparison, we set $q = 160$ bits for Saeednia *et al.*'s scheme [15], Laguillaumie and Vergnaud's scheme [9] and our SSDVS scheme.

As shown in Table 1, our signature length is the shortest (only 160 bits). In terms of the number of operations required, our scheme only requires 2 exponentiations in $\mathbb{Z}_q^*$ and 2 hash operations, and there is *no* pairing operation required. It is clear that our scheme outperforms Saeednia *et al.*'s and Laguillaumie and Vergnaud's scheme [9]. In particular, Laguillaumie and Vergnaud's scheme requires bilinear pairings operations which are computationally costly.

In Table 2, we compare our SIDSDVS scheme against the ID-based DVS scheme proposed in [18], which is an ID-based version of [15]. We assume that the bit length of the element in $\mathbb{G}_1$ is 1024 bits, and $q = 160$ bits.

It is clear from Table 2 that our scheme outperforms Susilo *et al.*'s scheme [18]. Our scheme only requires 2 pairing operations and 2 hash operations, while the scheme in [18] requires 3 pairings, 1 addition, 2 multiplication in $\mathbb{G}_1$, 3 exponentiation in $\mathbb{G}_2$ and 2 hash operations.

7.1 Further Discussions

More Efficient Schemes:
It is easy to see that the signature length of our scheme is based on the hash function employed, H_1. Hence, if we replace the hash function definition of H_1 with the following

$$H_1 : \{0,1\}^\infty \rightarrow \mathbb{Z}_{2^\ell}^*$$

where ℓ is the security parameter, then our signature length will be 2^ℓ. Assuming such a hash function exists and secure, then our signature scheme becomes more efficient.

A Scheme *without* Random Oracle:
We shall point out that a variant of our scheme can be used to construct an SSDVS (SIDSDVS) scheme *without* random oracle. Without losing generality, we illustrate the modification in a non-ID-based setting. The signature and verification algorithm are modified as follows.

1) Sign: To sign a message $m \in \{0,1\}^\infty$ for Bob, Alice performs the following.
   - Compute $k = y_B^{x_A} \pmod p$.
   - Compute $\sigma = \mathcal{E}_k(m)$, where $\mathcal{E}$ denotes a symmetric key encryption, such as AES.

   The signature on a message m is σ.

2) Verify: To verify the validity of a signature σ on a message m, Bob computes $k = y_A^{x_B}$ and tests whether

$$\sigma \stackrel{?}{=} \mathcal{E}_k(m)$$

   holds with equality. If it holds, then output True. Otherwise, output $\perp$.

One can view this scheme as a standard MAC scheme that uses a shared key k. The security of this scheme is based on the CDH assumption and the security of the underlying symmetric key encryption scheme $\mathcal{E}_k$.

8 Conclusion

In this paper, we presented the *first* short strong designated verifier signature scheme and its ID-based variant. Our schemes outperform the existing schemes known in the literature. Our construction has opened a new area of research, namely short strong designated verifier signature schemes, which have never been investigated before. Unlike the previous constructions, our schemes only produce 160 bits (if $q = 160$) signature which is very short compared to the existing schemes. We provided security proofs for our schemes based on the random oracle model. We also discussed variants of our scheme which is more efficient and provably secure under the standard assumption.

References

[1] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Advanced in Cryptology (Crypto'96)*, LNCS 1109, pp. 1-15, Springer-Verlag, 1996.

[2] D. Chaum, "Zero-knowledge undeniable signatures," in *Advances in Cryptology (Eurocrypt'90)*, LNCS 473, pp. 458-464, Springer-Verlag, 1990.

[3] D. Chaum and H. van Antwerpen, "Undeniable signatures," in *Advances in Cryptology (Crypto'89)*, LNCS 435, pp. 212-216, Springer-Verlag, 1990.

[4] Y. Desmedt, C. Goutier, and S. Bengio, "Special uses and abuses of the Fiat-Shamir passport protocol," in *Advances in Cryptology (Crypto'87)*, LNCS 293, pp. 21-39, Springer-Verlag, 1998.

[5] Y. Desmedt and M. Yung, "Weaknesses of undeniable signature schemes," *Advances in Cryptology (Eurocrypt'91)*, LNCS 547, pp. 205-220, Springer-Verlag, 1991.

[6] S. Galbraith and W. Mao, "Invisibility and anynimity of undeniable and confirmer signatures," in *the Cryptographers' Track at the RSA Conference 2003(CT-RSA 2003)*, LNCS 2612, pp. 80-97, Springer-Verlag, 2003.

[7] X. Huang, Y. Mu, W. Susilo, and F. Zhang, "Short designated verifier proxy signature from pairings," in *The First International Workshop on Security in*

Table 1: Comparison between Saeednia *et al.*'s scheme, Laguillaumie and Vergnaud's scheme and our scheme.

Scheme	**Signature Length**	$+$ in $\mathbb{Z}_q$	$*$ in $\mathbb{Z}_q^*$	exp. in $\mathbb{Z}_q^*$	$*$ in $(\mathbb{G},+)$	Pairing	Hash
Saeednia et al.*'s scheme*	480 bits	1	3	4	0	0	2
Laguillaumie and Vergnaud's scheme	271 bits	0	0	0	2	2	4
Our Scheme	160 bits	0	0	2	0	0	2

Table 2: Comparison between Susilo *et al.*'s scheme and our scheme.

Scheme	**Signature Length**	**Pairings**	$+$ in $\mathbb{G}_1$	* in $\mathbb{G}_1$	exp. in $\mathbb{G}_2$	Hash operations
Susilo et al.*'s scheme*	1344 bits	3	1	2	3	2
Our SIDSDVSscheme	160 bits	2	0	0	0	2

Ubiquitous Computing Systems (SecUbiq'05), LNCS 3823, pp. 835-844, Springer Verlag, 2005.

[8] M. Jakobsson, K. Sako, and R. Impagliazzo, "Designated verifier proofs and their applications," in *Advances in Cryptology (Eurocrypt'96)*, LNCS 1070, pp. 143-154, Springer-Verlag, 1996.

[9] F. Laguillaumie and D. Vergnaud, "Designated verifiers Signature: anonymity and efficient construction from *any* bilinear map," in *Fourth Conference on Security in Communication Networks'04 (SCN'04)*, LNCS 3352, pp. 107-121, Springer-Verlag, 2004.

[10] F. Laguillaumie and D. Vergnaud, "Multi-designated verifiers signatures," in *Information and Communications Security (ICICS/04)*, LNCS 3269, pp. 495-507, Springer-Verlag, 2004.

[11] B. Libert and J. J. Quisquater, "Identity based undeniable signatures," *Topics in Cryptology, CT-RSA 2004*, LNCS 2964, pp. 112-125, Springer-Verlag, 2004.

[12] H. Lipmaa, G. Wang and F.Bao, "Designated Verifier Signature Schemes: Attacks, New Security Notions and A New Construction," in *The 32nd International Colloquium on Automata, Languages and Programming, ICALP 2005*, LNCS 3580, pp. 459-471, Springer-Verlag, 2004.

[13] W. Ogata, K. Kurosawa, and S.-H. Heng, "The security of the FDH variant of Chaums undeniable signature scheme," in *Public Key Cryptography (PKC'05)*, LNCS 3385, pp. 328-345, Springer-Verlag, 2005.

[14] R. L. Rivest, A. Shamir, Y. Tauman, "How to leak a secret," in *Proceedings of Asiacrypt'01*, LNCS 2248, pp. 552-565, Springer-Verlag, 2001.

[15] S. Saeednia, S. Kramer, and O. Markovitch, "An efficient strong designated verifier signature scheme," in *The 6th International Conference on Information Security and Cryptology (ICISC'03)*, pp. 40-54, Springer-Verlag, 2003.

[16] R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk, "Universal designated-verifier signatures," in *Proceedings of Asiacrypt'03*, LNCS 2894, pp. 523-543, Springer-Verlag, 2003.

[17] R. Steinfeld, H. Wang, and J. Pieprzyk, "Efficient extension of standard schnorr/RSA signatures into universal designated-verifier signatures," in *Proceedings of PKC'04*, LNCS 2947, pp. 86-100, Springer-Verlag, 2004.

[18] W. Susilo, F. Zhang, and Y. Mu, "Identity-based strong designated verifier signature schemes," in *Proceedings of the 9th Australasian Conference on Information Security and Privacy (ACISP'04)*, LNCS 3108, pp. 313-324, Springer-Verlag, 2004.

Appendix

Proof of Theorem 2 Given a random instance (g, g^a, g^b) of the Gap Diffie-Hellman (GDH) problem, we will show how $\mathcal{B}$ can use $\mathcal{A}$ to obtain the value of g^{ab} with the help of Decisional Diffie-Hellman (DDH) Oracle. In the proof, we regard the hash function as the random oracle $\mathcal{H}$.

$\mathcal{B}$ will simulate all the oracles in the proof to answer $\mathcal{A}'s$ queries. In the simulation, $\mathcal{B}$ will maintain a list which is called *$\mathcal{H}$-List* to record the hash queries and the corresponding values. This *$\mathcal{H}$-List* consists of the items $(m, r, \sigma, coin)$, where (m, r) is the input of the hash, σ is the output of the hash, $coin = 1$ if $r = g^{ab}$ and $coin = 0$ if $r \neq g^{ab}$ (This is determined by the DDH oracle). We assume that $\mathcal{A}$ is well-behaved in the sense that $\mathcal{A}$ will never repeat the same queries in our simulation.

- Game$_0$ We consider an *EF-CMA* adversary $\mathcal{A}$ with the success probability $Succ^{EF-CMA}_{SSDVS,\mathcal{A}}$. The signers, Alice , selects her secret key $\mathcal{SK}_A = x_A \in \mathbb{Z}_q^*$, computes $y_A = g^{x_A}$ and sets her public key $\mathcal{PK}_A = y_A$. The designated verifier, Bob, also selects his secret key $\mathcal{SK}_B = x_B \in \mathbb{Z}_q^*$, computes $y_B = g^{x_B}$ sets his public key $\mathcal{PK}_B = y_B$.

  The adversary $\mathcal{A}$, with y_A and y_B, can query the hash oracle $\mathcal{H}$, the signing oracle S and verify oracle V. $\mathcal{A}$ outputs (m^*, σ^*), such that $\mathsf{Verify}(m^*, \sigma^*, \mathcal{SK}_B, \mathcal{PK}_A) = 1$.

Let $q_{\mathcal{H}}, q_S, q_V$ denote the number of queries to the $\mathcal{H}$, signing algorithm and verifying algorithm. The requirement is that m^* cannot be queried to the signing algorithm.

In any $\mathsf{Game_i}$, we denote by $\mathsf{Forge_i}$ the event $\mathsf{Verify}(m^*, \sigma^*, \mathcal{SK}_B, \mathcal{PK}_A) = 1$. By definition, we have

$$\Pr[\mathsf{Forge_0}] = Succ^{EF-CMA}_{SSDVS,\,\mathcal{A}}.$$

- $\mathsf{Game_1}$ In this game, $\mathcal{B}$ set $y_A = g^a$ and $y_B = g^b$ where g^a, g^b is the instance of the GDH problem. Since a, b are randomly chosen, therefore

  $$\Pr[\mathsf{Forge_1}] = \Pr[\mathsf{Forge_0}].$$

- $\mathsf{Game_2}$ In this game, $\mathcal{B}$ simulates the random oracle $\mathcal{H}$. For any query (m_i, r_i) to the oracle $\mathcal{H}$, $\mathcal{B}$ submits (g^a, g^b, r_i) to the DDH oracle and DDH oracle will tell $\mathcal{B}$ whether $r_i = g^{ab}$.

  1) $r_i = g^{ab}$, $\mathcal{B}$ checks the $\mathcal{H}$-*List*

     a. If there exists an item $(m_i, \perp, \sigma_i, 1)$ in the $\mathcal{H}$-*List* (item of this form can be added into the $\mathcal{H}$-*List* during the signing queries), $\mathcal{B}$ returns σ_i as the answer.

     b. Otherwise, $\mathcal{B}$ chooses $\sigma_i \in_R \mathbb{Z}_q^*$ such that there is no item $(\cdot, \cdot, \sigma_i, \cdot)$ in the $\mathcal{H}$-*List*. $\mathcal{B}$ then adds $(m_i, r_i, \sigma_i, 1)$ into the $\mathcal{H}$-*List* and returns σ_i as the answer.

  2) Else $r_i \neq g^{ab}$, $\mathcal{B}$ chooses $\sigma_i \in_R \mathbb{Z}_q^*$ such that there is no item $(\cdot, \cdot, \sigma_i, \cdot)$ in the $\mathcal{H}$-*List*. $\mathcal{B}$ then adds $(m_i, r_i, \sigma_i, 0)$ into the $\mathcal{H}$-*List* and returns σ_i as the answer.

  In the random oracle model, this game is clearly identical to the previous one. Hence

  $$\Pr[\mathsf{Forge_2}] = \Pr[\mathsf{Forge_1}]$$

- $\mathsf{Game_3}$ In this game, $\mathcal{B}$ simulates the signing algorithm. After receiving $\mathcal{A}$'s choice of the message m_i, $\mathcal{B}$ checks the $\mathcal{H}$-*List*:

  1) If there is an item $(m_i, r_i, \sigma_i, 1)$ in the $\mathcal{H}$-*List* (which means $r_i = g^{ab}$), $\mathcal{B}$ outputs σ_i as the signature.

  2) Else, $\mathcal{B}$ chooses $\sigma_i \in_R \mathbb{Z}_q^*$ such that there is no item $(\cdot, \cdot, \sigma_i, \cdot)$ in the $\mathcal{H}$-*List*. $\mathcal{B}$ then adds $(m_i, \perp, \sigma_i, 1)$ into the $\mathcal{H}$-*List* and returns σ_i as the signature.

  Then $\mathcal{A}$ gets the value σ_i as the signature of m_i. Of course, this oracle simulates the signature perfectly, so

  $$\Pr[\mathsf{Forge_3}] = \Pr[\mathsf{Forge_2}].$$

- $\mathsf{Game_4}$ In this game, $\mathcal{B}$ simulates the verifying algorithm. After receiving $\mathcal{A}$'s request of (m_i, σ_i), $\mathcal{B}$ performs the followings:

  1) If there is no item $(\cdot, \cdot, \sigma_i, \cdot)$ in the $\mathcal{H}$-*List*, $\mathcal{B}$ rejects (m_i, σ_i) as an invalid signature.

  2) Else, there is an item $(\cdot, \cdot, \sigma_i, \cdot)$ in the $\mathcal{H}$ -*List*:

     a. If this item has the form of $(m_i, \perp, \sigma_i, 1)$ or $(m_i, r_i, \sigma_i, 1)$, $\mathcal{B}$ will accept it as a valid signature.

     b. Otherwise, $\mathcal{B}$ rejects it as an invalid signature.

  This makes a difference only if (m_i, σ_i) is a valid signature, while σ_i is not queried from the $\mathcal{H}$. Since, $\mathcal{H}$ is uniformly distributed, this case happens with probability less than $\frac{1}{2^\ell - q_H - q_S}$. Summing up for all verifying queries, we get

  $$\Pr[\mathsf{Forge_3}] - \Pr[\mathsf{Forge_4}] \leq \frac{q_V}{2^\ell - q_H - q_S}.$$

After $\mathsf{Game_4}$ terminates, $\mathcal{A}$ outputs a valid signature (m^*, σ^*) such that $\mathsf{Verify}(m^*, \sigma^*, \mathcal{SK}_B, \mathcal{PK}_A) = 1$, that is there is an item $(\cdot, \cdot, \sigma^*, \cdot)$ in the $\mathcal{H}$-*List*. By the definition of the EF-CMA adversary model, m^* can not be queried in the sign oracle, so σ^* is returned as the hash value of $\mathcal{A}'s$ query (m^*, r^*). That is to say there is an item $(m^*, r^*, \sigma^*, 1)$ in the $\mathcal{H}$-*List* and $r^* = g^{ab}$. Therefore, $\mathcal{B}$ successfully solves an instance of the GDH problem with probability:

$$Succ^{GDH}_{\mathcal{B}} \geq Succ^{EF-CMA}_{SSDVS,\,\mathcal{A}} - \frac{q_V}{2^\ell - q_H - q_S}.$$

□

Proof of Theorem 3: Given a random instance (g, g^a, g^b) of the Gap Diffie-Hellman (GDH) problem , we will show how $\mathcal{B}$ can use $\mathcal{A}$ to obtain the value of g^{ab} with the help of Decisional Diffie-Hellman (DDH) Oracle. In the proof, we regard the hash function as the random oracle $\mathcal{H}$.

$\mathcal{B}$ will simulate all the oracles in the proof. In the simulation, $\mathcal{B}$ will maintain a list which is called $\mathcal{H}$-*List* to record the hash queries and the corresponding values. We assume that $\mathcal{A}$ is well-behaved in the sense that $\mathcal{A}$ will never repeat the same queries in our simulation.

- $\mathsf{Game_0}$ Consider an *PSI-CMA* adversary $\mathcal{A}$ with advantage $Adv^{PCI-CMA}_{SSDVS,\,\mathcal{A}}$. There are two possible signers A_0 and A_1 with the corresponding secret key $x_{A_0}, x_{A_1} \in \mathbb{Z}_q^*$ and public key as $y_{A_0} = g^{x_{A_0}}, y_{A_1} = g^{x_{A_1}} \pmod p$. The designated verifier, Bob, also selects his secret key $x_B \in \mathbb{Z}_q^*$ and sets his public key as $y_B = g^{x_B} \pmod p$.

  The adversary $\mathcal{A}$, with y_{A_0}, y_{A_1} and y_B, can query the random oracle $\mathcal{H}$, the signature algorithm and verify algorithm. $\mathcal{A}$ outputs a message m^*. A challenge signature is produced by flipping a coin $r \in \{0,1\}$ and computing $\sigma^* = Sign(m^*, x_{A_r}, y_B)$. Input σ^*, the adversary $\mathcal{A}$ outputs a bit r'.

Let $q_{\mathcal{H}}, q_S, q_V$ denote the number of queries to the $\mathcal{H}$, signing algorithm and verifying algorithm. The requirements are that m^* cannot be queried to the signing algorithm and (m^*, σ^*) cannot be queried to the verifying algorithm.

In any $\mathsf{Game_i}$, we denote by $\mathsf{Guess_i}$ the event $r = r'$. By definition, we have

$$|2\Pr[\mathsf{Guess_0}] - 1| = Adv^{PSI-CMA}_{SSDVS,\mathcal{A}}.$$

- $\mathsf{Game_1}$ In this game, $\mathcal{B}$ randomly chooses $a' \in_R \mathbb{Z}_q^*$ and compute $y_{A_0} = g^a$, $y_{A_1} = g^a \cdot g^{a'}$ and $y_B = g^b$, where g^a, g^b is the instance of GDH problem. Since a, b, a' are randomly chosen, therefore

$$\Pr[\mathsf{Guess_1}] = \Pr[\mathsf{Guess_0}].$$

- $\mathsf{Game_2}$ In this game, $\mathcal{B}$ simulates the random oracle $\mathcal{H}$. There is a table $\mathcal{H}$-*List* which maintains all the queries and answers and consists of the item $(m_i, r_i, \sigma_i, signer_i, coin_i)$. Here (m_i, r_i) is the input of the $\mathcal{H}$, σ_i is the output of $\mathcal{H}$, $signer_i \in \{A_0, A_1, \perp\}$ and $coin_i \in \{0, 1\}$. For any query (m_i, r_i) to the oracle $\mathcal{H}$, $\mathcal{B}$ submits (g^a, g^b, r_i) to the DDH oracle and DDH oracle will tell $\mathcal{B}$ whether $r_i = g^{ab}$.

  1) If $r_i = g^{ab}$, $\mathcal{B}$ checks whether the $\mathcal{H}$-*List* contains $(m_i, \perp, \sigma_i, A_0, 1)$ (item has this form will be added into the $\mathcal{H}$-*List* during the simulation of the sign algorithm). If it does,$\mathcal{B}$ outputs σ_i as the answer. Else $\mathcal{B}$ chooses $\sigma_i \in_R \mathbb{Z}_q^*$ such that there is no item $(\cdot, \cdot, \sigma_i, \cdot, \cdot)$ in the $\mathcal{H}$-*List*. Then $\mathcal{B}$ adds $(m_i, r_i, \sigma_i, A_0, 1)$ to the $\mathcal{H}$-*List* and outputs σ_i as the answer.
  2) Else $r_i \neq g^{ab}$, then $\mathcal{B}$ submits $(g^a \cdot g^{a'}, g^b, r_i)$ to the DDH oracle and DDH oracle will tell $\mathcal{B}$ whether $r_i = g^{(a+a')b}$.
     a. If $r_i = g^{(a+a')b}$, $\mathcal{B}$ checks whether the $\mathcal{H}$-*List* contains $(m_i, \perp, \sigma_i, A_1, 1)$ (item has this form will be added into the $\mathcal{H}$-*List* during the simulation of the sign algorithm). If it does, $\mathcal{B}$ outputs σ_i as the answer.
     b. Else $\mathcal{B}$ chooses $\sigma_i \in_R \mathbb{Z}_q^*$ such that there is no item $(\cdot, \cdot, \sigma_i, \cdot, \cdot)$ in the $\mathcal{H}$-*List*. Then $\mathcal{B}$ adds $(m_i, r_i, \sigma_i, A_1, 1)$ to the $\mathcal{H}$-*List* and outputs σ_i as the answer.
  3) Otherwise, $r_i \neq g^{ab}$ and $r_i \neq g^{(a+a')b}$. If this case happens, $\mathcal{B}$ chooses $\sigma_i \in_R \mathbb{Z}_q^*$ such that there is no item $(\cdot, \cdot, \sigma_i, \cdot, \cdot)$ in the $\mathcal{H}$-*List*. Then $\mathcal{B}$ adds $(m_i, r_i, \sigma_i, \perp, 0)$ to the $\mathcal{H}$-*List* and outputs σ_i as the answer.

In the random oracle model, this game is clearly identical to the previous one. Hence

$$\Pr[\mathsf{Guess_2}] = \Pr[\mathsf{Guess_1}].$$

- $\mathsf{Game_3}$ In this game, $\mathcal{B}$ simulates the signing algorithm. After receiving $\mathcal{A}$'s choice of the message m_i and the sender A_r $(r \in \{0, 1\})$, $\mathcal{B}$ performs:

  1) If there is an item $(m_i, r_i, \sigma_i, A_r, 1)$ in the $\mathcal{H}$-*List*, $\mathcal{B}$ outputs σ_i as the signature.
  2) Else $\mathcal{B}$ chooses $\sigma_i \in_R \mathbb{Z}_q^*$ such that there is no item $(\cdot, \cdot, \sigma_i, \cdot, \cdot)$ in the $\mathcal{H}$-*List*. Then $\mathcal{B}$ adds $(m_i, \perp, \sigma_i, A_r, 1)$ to the $\mathcal{H}$-*List* and outputs σ_i as the answer.

Of course, this oracle simulates the signature, so

$$\Pr[\mathsf{Guess_3}] = \Pr[\mathsf{Guess_2}].$$

- $\mathsf{Game_4}$ In this game, $\mathcal{B}$ simulates the verifying algorithm. After receiving $\mathcal{A}$'s request of (m_i, σ_i, A_r) $(r \in \{0, 1\})$, $\mathcal{B}$ does:

  1) If there is no item $(\cdot, \cdot, \sigma_i, \cdot, \cdot)$ in the $\mathcal{H}$-*List*, $\mathcal{B}$ rejects (m_i, σ_i) as an invalid signature.
  2) Else, there is an item $(\cdot, \cdot, \sigma_j, \cdot, \cdot)$ in the $\mathcal{H}$-*List* such that $\sigma_j = \sigma_i$:
     a. If this item has the form $(m_i, \perp, \sigma_i, A_r, 1)$ or $(m_i, r_i, \sigma_i, A_r, 1)$, then $\mathcal{B}$ accepts (m_i, σ_i) as a valid signature.
     b. Otherwise, $\mathcal{B}$ rejects it as an invalid signature.

This makes a difference only if (m_i, σ_i, A_r) is a valid signature, while σ_i is not queried from the $\mathcal{H}$. Since, $\mathcal{H}$ is uniformly distributed, this case happens with probability less than $\frac{1}{2^\ell - q_H - q_S}$. Summing up for all verifying queries, we get

$$\Pr[\mathsf{Guess_3}] - \Pr[\mathsf{Guess_4}] \leq \frac{q_V}{2^\ell - q_H - q_S}.$$

- $\mathsf{Game_5}$ In this game, $\mathcal{A}$ outputs a message m^* such that m^* is not queried to the singing algorithm. we choose $r \in_R \{0, 1\}$ and $\sigma^* \in_R \mathbb{Z}_q^*$ such that there is no item $(\cdot, \cdot, \sigma^*, \cdot, \cdot)$ in the $\mathcal{H}$-*List*. Then we return (m^*, σ^*) to the adversary $\mathcal{A}$. $\mathcal{A}$ now still can query to the signing and verifying algorithm. Since the challenge signature is randomly chosen it gives $\mathcal{A}$ no information about r, in an information theoretic sense, we have

$$\Pr[\mathsf{Guess_5}] = 1/2.$$

The final game is indistinguishable from the previous one unless (m^*, σ^*) is queried to the verifying algorithm, m^* is queried to the signing oracle, (m^*, g^{ab}) or $(m^*, g^{(a'+a)b})$ is queried to the $\mathcal{H}$ by the adversary $\mathcal{A}$. By the definition of the PSI-CMA attack model, m^* can not be queried to the signing oracle and (m^*, σ^*) cannot be queried to the verifying algorithm. So the last case must happen, that is to say there is an item $(m^*, r^*, \sigma^*, A_0, 1)$ or $(m^*, r^*, \sigma^*, A_1, 1)$ in the $\mathcal{H}$-*List*:

1) If $(m^*, r^*, \sigma^*, A_0, 1)$ is in the $\mathcal{H}$-*List*, $\mathcal{B}$ outputs r^* as the solution of the GDH problem since $r^* = g^{ab}$.
2) Else $(m^*, r^*, \sigma^*, A_1, 1)$ is in the $\mathcal{H}$-*List*, $\mathcal{B}$ outputs $r^* \cdot (g^b)^{-a'}$ as the solution of the GDH problem since $r^* \cdot (g^b)^{-a'} = g^{(a+a')b} \cdot (g^b)^{-a'} = g^{ab}$.

Therefore, $\mathcal{B}$ can obtain the solution of GDH problem. Therefore

$$|\Pr[\mathsf{Guess}_5] - \Pr[\mathsf{Guess}_4]| = Succ_{\mathcal{B}}^{GDH}.$$

That is

$$Succ_{\mathcal{B}}^{GDH} \geq \frac{1}{2}\, Adv_{SSDVS,\, \mathcal{A}}^{PSI-CMA} - \frac{q_V}{2^\ell - q_H - q_S}.$$

□

Xinyi Huang received his M.S degree in operational research and cybernetics from Nanjing Normal University (China) in 2005. Currently, he is a Ph. D student of School of Information Technology and Computer Science in the University of Wollongong, Australia. His research mainly focuses on cryptography and its applications.

Willy Susilo is an Associate Professor at the School of Information Technology and Computer Science of the University of Wollongong. He is the coordinator of Network Security Research Laboratory at the University of Wollongong. His research interests include cryptography, information security, computer security and network security. His main contribution is in the area of digital signature schemes, in particular fail-stop signature schemes and short signature schemes. He has served as a program committee member in a number of international conferences. He is a member of the IACR. He has published numerous publications in the area of digital signature schemes and encryption schemes.

Yi Mu received his PhD from the Australian National University in 1994. He was previously with the School of Computing and IT at the University of Western Sydney as a lecturer and the Department of Computing at Macquarie University as a senior lecturer. He has been with the University of Wollongong since 2003. His current research interests include network security, electronic commerce security, wireless security, access control, computer security, and cryptography. He also previously worked at quantum cryptography, quantum computers, atomic computations, and quantum optics. He has published 120 fully refereed papers in international journals and conferences and 4 edited books. Yi Mu has served in program committees of 42 international conferences and editorial boards of 4 international Journals. He is a senior member of the IEEE and a member of the IACR.

Futai Zhang is a professor in the school of Mathematics and Computer Science at Nanjing Normal University, China.He received his Ph.D in Cryptography from Xidian University, China in 2001.His research interests are public key cryptography and the security of multicast and Ad Hoc networks.