1-1-2011

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

Q Huang
South China Agricultural University

G Yang
National University of Singapore

D S Wong
City University of Hong Kong

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Recommended Citation

Huang, Q; Yang, G; Wong, D S; and Susilo, Willy: Efficient strong designated verifier signature schemes without random oracle or with non-delegatability 2011, 373-385.
https://ro.uow.edu.au/infopapers/1754

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

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 tellingwhether 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 standardmodel,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 ofwhich 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

era2015, Efficient, strong, designated, verifier, signature, schemes, without, random, oracle, non, delegatability

Disciplines

Physical Sciences and Mathematics

Publication Details

Huang, Q., Yang, G., Wong, D. S. & Susilo, W. (2011). Efficient strong designated verifier signature schemes without random oracle or with non-delegatability. International Journal of Information Security, 10 (6), 373-385.

Efficient Strong Designated Verifier Signature Schemes without Random Oracles or Delegatability

Qiong Huang* Guomin Yang† Duncan S. Wong* Willy Susilo‡

Abstract

Designated verifier signature (DVS) is a cryptographic primitive that allows a signer to convince a verifier the validity of a statement in a way that the verifier is unable to transfer the conviction to a third party. In DVS, signatures are publicly verifiable. The validity of a signature ensures that it is from either the signer or the verifier. Strong DVS (SDVS) enhances the privacy of the signer so that anyone except the designated verifier cannot verify the signer's signatures.

In this paper we propose a highly efficient SDVS scheme based on pseudorandom functions, which is proved to be secure in the standard model. Compared with the most efficient SDVS scheme secure in the random oracle model, our scheme has almost the same complexity in terms of both the computational cost of generating a signature and signature size. A signature of our scheme is simply the output of a pseudorandom function. The security of the scheme is tightly reduced to the hardness of DDH problem and the security of the pseudorandom function.

Since our scheme is vulnerable to delegatability attacks, the study of which was initiated by Lipmaa, Wang and Bao in ICALP 2005, we then propose another construction of SDVS, which is the *first* one immune to delegatability attacks. The scheme is also very efficient, and has the same signature size with that of Lipmaa-Wang-Bao non-delegatable DVS scheme. We show that it is secure based on discrete logarithm assumption and gap Diffie-Hellman assumption in the random oracle model.

Keywords: strong designated verifier signature, non-delegatability, non-transferability, random oracles, standard model, signature scheme

*Department of Computer Science, City University of Hong Kong, Hong Kong S.A.R., China P.R. {csqhuang@student.cityu.edu.hk, duncan@cityu.edu.hk}.

†Temasek Laboratories, National University of Singapore, Singapore. {tslyg@nus.edu.sg}

‡School of Computer Science and Software Engineering, University of Wollongong, Australia. {wsusilo@uow.edu.au}.

Contents

1 Introduction

In undeniable signatures [7], 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. However, Alice can only decide when to prove, but not who to verify. So the conviction can be transferred to one or many verifiers. To solve this problem, Jakobsson, Sako and Impagliazzo [14] introduced the notion of *designated verifier proofs* (DVP) in Eurocrypt 1996, which allows Alice to prove the validity of a statement to Bob so that Bob is convinced of the validity of the statement, but unlike ordinary signature schemes, he could not 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 transcripts. Designated Verifier Signatures (DVS) are the non-interactive version of DVP. They are publicly verifiable so that if a signature is valid, any third party is convinced that it was produced by either Alice or Bob. DVS has applications in voting, i.e. a voting center should not able to convince others that a voter has voted, undeniable signatures, i.e. use DVS to non-interactively confirm/disavow a signature, and other applications where only certain people can be convinced of something.

In some cases it is required that if Cindy does not know Bob's secret key, she cannot tell if a signature for Bob is from Alice or other signers. Jakobsson et al. then introduced in the same work [14] a variant of DVS to enhance the signer's privacy, called *strong designated verifier signature* (SDVS), which requires that any two signer's signatures for the same verifier are indistinguishable to anyone other than the designated verifier. This property had not been formally studied and defined until 2004. Laguillaumie and Vergnaud [17] for the first time formally defined the property '*privacy of signer's identity*' to capture the 'strongness' of SDVS.

Jakobsson et al. [14] suggested a generic approach for constructing SDVS. Each user in the system holds two key pairs, one for a DVS scheme and the other for a public key encryption scheme. To sign a message M, Alice first uses its signing key to produce a DVS signature on M for Bob and then encrypts the signature under Bob's encryption key. The ciphertext is sent as the SDVS signature to Bob, who first decrypts the ciphertext using his decryption key to obtain Alice's DVS signature, and then checks the validity of the signature using verification keys of both Alice and Bob. This approach is conceptually simple, however, the resulting scheme is not efficient enough, as it requires the encryption of a DVS signature which usually is comprised of several group elements.

1.1 Related Work

Ring Signature. DVS focuses on the setting in which there are only two parties who can produce indistinguishable signatures. Rivest, Shamir and Tauman [23] extended the notion to the setting in which there are a number of signers, and introduced the notion of *ring signatures*. The signers form a group, and any signer in the group can produce indistinguishable signatures, but anyone outside of it cannot. Anyone is ensured that a valid signature is from the group, but cannot tell the identity of the real signer. Rivest et al. proposed the first ring signature scheme based on trapdoor permutations and symmetric encryption in the random oracle model [3]. The first efficient ring signature scheme secure without random oracles was proposed by Shacham and Waters [25], which is based on Waters signature [30]. Since DVS is the two-user version of ring signature, it turns out that the two-user version of Shacham-Waters scheme is a DVS scheme secure without random oracles.

UDVS. Steinfeld, Bull, Wang and Pieprzyk [26] studied another variant of DVS, named *universal designated verifier signature* (UDVS), in which a (standard) signature holder can designate any third party as the verifier and transform the signature into a DVS signature for that party. They proposed a UDVS scheme based on Boneh-Lynn-Shacham (BLS) short signature scheme [6], thus only secure in the random oracle model. They also showed that UDVS is equivalent to identity-based key encapsulation mechanism. Zhang, Furukawa and Imai [31] proposed another UDVS scheme based on Boneh-Boyen short signature [4] in the standard model. Laguillaumie, Libert and Quisquater [16] and Huang, Susilo, Mu and Wu [12] came up with the same UDVS scheme based on Waters signature, which is also secure without random oracles. Vergnaud presented in [29] another two UDVS schemes, one based on Boneh-Boyen signature and secure in the standard model but requiring a very strong assumption named *knowledge-of-exponent assumption* [8], and the other based on BLS signature and thus secure in the random oracle model.

MDVS. The notion of *multi-designated verifiers signature* (MDVS) was first studied by Jakobsson et al. [14] as well. In MDVS, the signer can designate many parties as verifiers for checking the validity of a signature. Jakobsson et al. proposed a concrete scheme of MDVS based on their DVS scheme by aggregating the public keys and secret keys of all the designated verifiers so that only if all of them cooperate together can they simulate the signatures. Laguillaumie and Vergnaud [18] were the first to give a formal definition of MDVS. They gave a construction of MDVS based on ring signatures, which however, fails to provide privacy of signer's identity [17]. They also gave another construction of bi-designated verifier signature scheme based on bilinear maps, protecting the anonymity of signers.

IBDVS. The above studies of (variants of) DVS are in the public key infrastructure setting. Another interesting topic is identity-based designated verifier signature (IBDVS). Susilo, Zhang and Mu [28] proposed the first identity-based strong designated verifier signature (IBSDVS) scheme with random oracles based on Bilinear Diffie-Hellman (BDH) assumption. Huang, Susilo, Mu and Zhang [13] proposed a SDVS scheme based on Diffie-Hellman key exchange. Their scheme is very efficient, and has the shortest signature, which simply is the hash of the message and the common key shared between Alice and Bob. The security is based on Gap Diffie-Hellman (GDH) assumption in the random oracle model. They also showed how to deploy their technique to the identity-based setting to obtain an IBSDVS scheme.

Non-Delegatable DVS. In ICALP 2005 Lipmaa, Wang and Bao [19] initiated the study of *delegatability attacks* in DVS. Roughly speaking, a DVS scheme is non-delegatable if one produces a valid signature, it must '*know*' the secret key of either Alice or Bob. In other words, there exists an extractor which given oracle access to the forger algorithm (and the message), outputs the secret key of either the signer or the verifier. Though delegatability is desirable in some settings, e.g. proxy DVS, it is undesirable in many other settings [19].

Many schemes were shown to be vulnerable to delegatability attacks, such as [24, 26, 27, 17]. Lipmaa et al. then proposed the first non-delegatable DVS scheme with tight security reduction using the technique of Katz and Wang [15]. The signer's public key consists of g_1^x, g_2^x, and a signature is a non-interactive zero-knowledge proof of the equal discrete logarithm of the two components in the public key. Unforgeability of their scheme tightly reduces to the Decisional Diffie-Hellman (DDH) assumption, and the non-delegatability is proved by rewinding the forger in the random oracle model. Huang, Susilo, Mu and Wu [11] proposed a UDVS scheme based on BLS signature, which is also immune to delegatability attacks. Their signature is a proof of knowledge of either a BLS signature or the verifier's secret key. Very recently, Huang, Susilo and Wong [10] proposed the first IBDVS scheme supporting non-delegatability, which is based on Gentry-Silverberg hierarchical identity-based encryption scheme [9]. The signature also contains a proof of knowledge showing either Alice or Bob's secret key is involved in the generation.

To the best of our knowledge, all the non-delegatable DVS schemes (and variants) use Fiat-Shamir heuristic to implement the proof of knowledge, and therefore their security relies on the randomness of the underlying hash function. So far there is no DVS scheme in the standard model that supports non-delegatability. The dilemma is due to the lack of an efficient non-interactive proof of knowledge system which allows the extraction of the witness in the position of exponents, since in almost all the signature schemes the secret key is the exponent in the public key.

1.2 Our Contribution

In the line of research on DVS and its variants, there are some problems that have been open since their introduction, for example:

1. construction of non-delegatable DVS and its variants without random oracles;
2. construction of non-delegatable SDVS in the random oracle model;
3. construction of SDVS without random oracles which is much more efficient than the generic construction [14].

In this work we give complete and affirmative answers to the last two problems. Specifically, we construct an SDVS scheme which is provably secure without resorting to the random oracle heuristic. The scheme is surprisingly simple and much more efficient than the aforementioned generic construction. The signer S shares a common key with each designated verifier. To sign a message M for verifier V, S uses the key shared with V as the key to select a pseudorandom function PRF. Its signature on M is simply the output of PRF on input M, and thus is very short. The security of our scheme, e.g. unforgeability and privacy of signer's identity, tightly reduces to that of the pseudorandom function and the hardness of DDH problem. Our scheme also enjoys the perfect non-transferability, as S and V can produce the same signature on any message.

Regarding the security proofs, we use a slightly different approach here. Usually in the proof one gradually changes the game in the setup of public keys and the way that the adversary's queries are handled, and finally comes to a game in which the adversary wins only with the desired advantage. However, the condition on winning the game remains untouched. While in our proofs, besides the aforementioned modifications to the game, we also change step-by-step the condition that the adversary wins the game. In this way, we are able to tightly reduce the security of this extremely simple scheme to DDH assumption and the security of the pseudorandom function.

Regarding the delegatability attacks, it is not hard to see that the scheme is vulnerable to such attacks, since the key for the pseudorandom function is derived from the common key shared between S and V. If one knows the shared key, certainly it can represent S to sign any message for V. We then propose another efficient construction of SDVS, which to the best of our knowledge, is the *first* non-delegatable SDVS scheme. The signature is a proof of knowledge of the secret key of either S or V, and thus the non-delegatability is achievable. The price of supporting this extra property is that the security of the scheme is only provable in the random oracle model. Unforgeability of the scheme is based on Discrete Logarithm (DL) assumption, and the privacy of signer's identity relies on the intractability of Gap Diffie-Hellman (GDH) problem.

1.3 Outline

In the next section we give the formal definition of SDVS, as well as its security model. We then review in Sec. 3 the definition of pseudorandom functions and the underlying assumptions used in our construction. The construction of SDVS scheme without random oracles is presented in Sec. 4, followed by its security analysis. The non-delegatable SDVS scheme is then proposed in Sec. 5, along with its security proofs in the random oracle model. We compare our schemes with some existing (S)DVS schemes in the literature in Sec. 6. Finally, the paper is concluded in Sec. 7.

2 Strong Designated Verifier Signature

Roughly speaking, strong designated verifier signature (SDVS) allows the signer to sign documents so that only the intended verifier can verify the validity of its signatures and in the meanwhile prevents the verifier to transfer the conviction. Below we give the formal definition of this notion.

Definition 2.1 (SDVS)**.** *A* Strong Designated Verifier Signature (SDVS) *scheme in the public-key infrastructure setting consists of the following four (probabilistic) polynomial-time algorithms.*

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

$$\begin{aligned} &\Pr[\mathsf{Ver}(\mathtt{sk}_v, \mathtt{pk}_s, \mathtt{pk}_v, M, \mathsf{Sign}(\mathtt{sk}_s, \mathtt{pk}_s, \mathtt{pk}_v, M)) = 1] &= 1, \\ \text{and} \quad &\Pr[\mathsf{Ver}(\mathtt{sk}_v, \mathtt{pk}_s, \mathtt{pk}_v, M, \mathsf{Sim}(\mathtt{sk}_v, \mathtt{pk}_s, \mathtt{pk}_v, M)) = 1] &= 1 \end{aligned}$$

Besides the correctness, a secure SDVS should also satisfy unforgeability, non-transferability and privacy of signer's identity. If an SDVS scheme is said to be non-delegatable, it should also support non-delegatability. We define these properties in the following subsections respectively.

2.1 Unforgeability

Unforgeability requires that any third party other than the signer S and the designated verifier V, cannot forge a signature on behalf of S with non-negligible probability. Formally, it is defined by the following game played between a game challenger C and a probabilistic polynomial-time adversary $\mathcal{A}$:

1. C prepares the key pairs of 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. In this phase, $\mathcal{A}$ adaptively issues queries to the following oracles for polynomially many times:
   - $\mathcal{O}_{\mathsf{Sign}}$: Given a message M, the oracle returns a signature σ on M to $\mathcal{A}$, which is valid with respect to $\mathtt{pk}_s$ and $\mathtt{pk}_v$.
   - $\mathcal{O}_{\mathsf{Sim}}$: Given a message M, the oracle returns a signature σ on M to $\mathcal{A}$, which is valid with respect to $\mathtt{pk}_s$ and $\mathtt{pk}_v$.
   - $\mathcal{O}_{\mathsf{Ver}}$: 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 with respect to $\mathtt{pk}_s$ and $\mathtt{pk}_v$, and 0 otherwise.
3. Finally, $\mathcal{A}$ outputs its forgery, (M^*, σ^*). It wins the game if
   (a) $1 \leftarrow \mathsf{Ver}(\mathtt{sk}_v, \mathtt{pk}_s, \mathtt{pk}_v, M^*, \sigma^*)$, and
   (b) $\mathcal{A}$ did not query $\mathcal{O}_{\mathsf{Sign}}$ and $\mathcal{O}_{\mathsf{Sim}}$ on input M^*.

Definition 2.2 (Unforgeability)**.** *An SDVS scheme is said to be $(t, q_{\mathsf{Sign}}, q_{\mathsf{Sim}}, q_{\mathsf{Ver}}, \epsilon)$*-unforgeable *if there is no adversary $\mathcal{A}$ which runs in time at most t, issues at most q_{Sign} queries to $\mathcal{O}_{\mathsf{Sign}}$, at most q_{Sim} queries to $\mathcal{O}_{\mathsf{Sim}}$, and at most q_{Ver} queries to $\mathcal{O}_{\mathsf{Ver}}$, and wins the game with probability at least ϵ.*

2.2 Non-transferability

Non-transferability says that given a message-signature pair (M, σ) which is accepted by the designated verifier, it is infeasible for any probabilistic polynomial-time distinguisher to tell whether the message was signed by the signer or the designated verifier. Formally, we consider the following definition.

Definition 2.3 (Non-Transferability). *An SDVS scheme is* non-transferable *if the signature output by the signer is* computationally indistinguishable *from that output by the designated verifier, i.e.*

$$\{\mathsf{Sign}(\mathtt{sk}_s, \mathtt{pk}_s, \mathtt{pk}_v, M)\} \approx \{\mathsf{Sim}(\mathtt{sk}_v, \mathtt{pk}_s, \mathtt{pk}_v, M)\}$$

That is, for any probabilistic polynomial-time distinguisher $\mathcal{D}$, for any $(\mathtt{pk}_s, \mathtt{sk}_s) \leftarrow \mathsf{Kg}(1^k)$, $(\mathtt{pk}_v, \mathtt{sk}_v) \leftarrow \mathsf{Kg}(1^k)$, any message $M \in \{0,1\}^*$, it holds that

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

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 two distributions are identical, we say that the SDVS scheme is* perfectly non-transferable*.*

2.3 Privacy of Signer's Identity

Privacy of signer's identity (PSI), first defined by Laguillaumie and Vergnaud [17], formalizes of the motivation for the introduction of SDVS. Basically, it requires that one cannot tell signatures generated by signer S_0 for V apart from those by S_1, if it does not know V's secret key. Below is the formal definition of PSI, modeled by a game played between the challenger C and a distinguisher $\mathcal{D}$.

1. C generates the 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 for polynomially many times as in the unforgeability game, except that all the oracles take an additional input $d \in \{0,1\}$ indicating the signer. That is, the oracles generate/verify signatures with respect to $\mathtt{pk}_{s_d}$ and $\mathtt{pk}_v$.
3. $\mathcal{D}$ outputs a message M^*. C then 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 except that it could not query $\mathcal{O}_{\mathsf{Ver}}$ on input (d, M^*, σ^*) for any $d \in \{0,1\}$. Finally it outputs a bit b', and wins the game if $b' = b$.

Definition 2.4 (Privacy of Signer's Identity). *An SDVS scheme is said to be $(t, q_{\mathsf{Sign}}, q_{\mathsf{Sim}}, q_{\mathsf{Ver}}, \epsilon)$-*PSI-secure *if there is no adversary $\mathcal{D}$ which runs in time at most t, issues at most q_{Sign} queries to $\mathcal{O}_{\mathsf{Sign}}$, at most q_{Sim} queries to $\mathcal{O}_{\mathsf{Sim}}$, and at most q_{Ver} queries to $\mathcal{O}_{\mathsf{Ver}}$, and wins the game above with probability that deviates from one-half by more than ϵ.*

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

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

2.4 Non-Delegatability

Intuitively, non-delegatability requires that if one produces a valid signature on a message, it must 'know' the secret key of S or V. Formally, we consider the following definition, which is modified from [19].

Definition 2.5 (Non-delegatability)**.** *Let $\kappa \in [0,1]$ be the knowledge error, and $\mathcal{F}$ be 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 probabilistic polynomial-time 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 : Readers may notice that our definition of non-delegatability is different from the original one given in [19]. However, like the two equivalent definitions of proofs of knowledge [1], it can be shown that Def. 2.5 is equivalent to Lipmaa et al.'s definition.

3 Tools and Assumptions

Pseudorandom Function Ensemble: Roughly speaking, pseudorandom functions are functions whose output distribution is computationally indistinguishable from that of truly random functions, if the seeds are random. The following definition is from [20].

Definition 3.1. *Let $\{A_k, B_k\}_{k\in\mathbb{N}}$ be a sequence of domains and let $\mathbf{F} = \{\mathrm{PRF}_k\}_{k\in\mathbb{N}}$ be a function ensemble such that the random variable PRF_k assumes values in the set of $A_k \to B_k$ functions. Then* $\mathbf{F}$ *is called an* efficiently computable pseudorandom function ensemble *if the following conditions hold:*

1. *(*efficient computation*) There exist probabilistic polynomial-time algorithms $\mathcal{I}$ nd $\mathcal{V}$, 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 probabilistic polynomial-time 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 corresponding 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 and $q|p-1$. Let $\mathbb{G}$ be a multiplicative group of order q and g be its generator. For example, $\mathbb{G}$ is an order-q subgroup of $\mathbb{Z}_p^*$.

Definition 3.2 (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 satisfies the following condition:*

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

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

Definition 3.3 (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 satisfies the following condition:*

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

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

Definition 3.4 (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* $\mathcal{O}_{\rm ddh}$ *which on input* $(g_1, g_2, g_1^u, g_2^v) \in \mathbb{G}^4$ *tells if the given tuple is a DDH tuple (i.e.* $u \stackrel{?}{=} v$*), runs in time at most* t*, and satisfies the following condition:*

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

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 Oracles

Let $\mathbb{G}$ be a cyclic multiplicative group of large prime order q, and g be its generator. Let $\lambda(\cdot)$ and $\ell(\cdot)$ be two positive polynomials in the security parameter k. Let $\mathbf{F} = \{\mathrm{PRF}_K\}_{K \in \mathcal{K}_{\rm PRF}}$ be a family of pseudorandom functions $\mathrm{PRF}_K : \{0,1\}^\lambda \rightarrow \{0,1\}^\ell$ with key space $\mathcal{K}_{\rm PRF}$. We assume that the message space is $\{0,1\}^\lambda$.[2] Let $\mathtt{H} : \mathbb{G} \rightarrow \mathcal{K}_{\rm PRF}$ be a collision-resistant hash function. Our SDVS scheme is depicted in Fig. 1, where for simplicity, we omit the usage of function H, and simply use the shared key (i.e. $K = g^{x_s x_v}$) as the key for the pseudorandom function.

```
$\mathsf{Kg}(1^k)$:
    $x \xleftarrow{\$} \mathbb{Z}_q$
    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 $\sigma \stackrel{?}{=} \mathrm{PRF}_K(M)$

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

Figure 1: Our SDVS Scheme in the Standard Model, SDVS_1

Below we analyze the security of SDVS_1. As we shall see later (Theorem 4.2), 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 queries to oracle $\mathcal{O}_{\mathsf{Sim}}$, as these queries can be perfectly handled using oracle $\mathcal{O}_{\mathsf{Sign}}$.

Security Analysis. It is known that pseudorandom function is a (deterministic) message authentication code that is strongly unforgeable under chosen message attacks [2]. After obtaining many signatures on messages at its choices, the adversary still could not forge a new one. On the other hand, 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 privacy of signer's identity cannot distinguish which signer produced the challenge signature. Formally, we have the following theorems.

[2] We can also assume the message space to be $\{0,1\}^*$ and then use a collision-resistant hash function to map the messages to $\{0,1\}^\lambda$.

Theorem 4.1. *Suppose that $\mathcal{A}$ is an adversary that $(t, q_{\mathsf{Sign}}, q_{\mathsf{Ver}}, \epsilon)$ breaks the unforgeability of* SDVS_1*. There exist 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* $\mathbf{F}$ *such that*

$$t_1, t_2 \approx t, \quad \text{and} \quad \epsilon_{\mathrm{ddh}} + \epsilon_{\mathrm{prf}} > \epsilon - \frac{q_{\mathsf{Sign}} + q_{\mathsf{Ver}}}{2^\ell}.$$

Proof. We'll prove the theorem by a series of games. Let $\mathcal{A}$ be an adversary against the unforgeability of our SDVS scheme. 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 $a, b \overset{\$}{\leftarrow} \mathbb{Z}_q$, 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 as a random element of $\mathbb{G}$, i.e. $K \overset{\$}{\leftarrow} \mathbb{G}$. Besides, the validity of the adversary's forgery is now also with respect to this random key K. 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 an instance of DDH problem, 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 $\mathbb{G}, g, p$. 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 $\sigma \overset{?}{=} \mathrm{PRF}_Z(M)$. Finally, $\mathcal{A}$ outputs its forgery, (M^*, σ^*). The adversary $\mathcal{A}_1$ then tests if $\sigma^* = \mathrm{PRF}_Z(M^*)$. If so, it outputs 1, meaning that $Z = g^{ab}$; otherwise it outputs 0, meaning that Z is randomly chosen from $\mathbb{G}$. Obviously, 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 of $\mathbb{G}$, 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

$$\left|\Pr\left[b = 1 | Z = g^{ab}\right] - \Pr\left[b = 1 | Z \overset{\$}{\leftarrow} \mathbb{G}\right]\right| \quad = \quad |\Pr[X_0] - \Pr[X_1]|$$

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

G_2: Notice that 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 obtain that

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

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 the 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 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. Obviously, if F is chosen from $\mathbf{F}$, $\mathcal{A}_2$ perfectly simulated Game G_1; if F is a truly random function, $\mathcal{A}_2$ perfectly simulated Game G_2. Therefore, we obtain equation (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 and q_{Ver} verification queries, the probability that the forgery output by the adversary is valid in Game G_2 is upper bounded by

$$\Pr[X_2] \leq \frac{1}{2^\ell - q_{\mathsf{Sign}} - q_{\mathsf{Ver}}} < \frac{q_{\mathsf{Sign}} + q_{\mathsf{Ver}}}{2^\ell} \tag{4}$$

Combining equations (1)–(4), we get that

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

This completes the proof. □

Theorem 4.2. SDVS_1 *is perfectly non-transferable.*

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

Theorem 4.3. *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 $(t_1, \epsilon_{\text{ddh}})$-breaks DDH assumption and an adversary $\mathcal{A}_2$ that $(t_2, \epsilon_{\text{prf}})$-breaks the pseudorandomness of* $\mathbf{F}$ *with*

$$t_1, t_2 \approx t, \quad \textit{and} \quad \epsilon_1 + \epsilon_2 \geq \frac{\epsilon}{2}$$

Proof. Let $\mathcal{D}$ be the distinguisher against privacy of signer's identity. We consider the following games, and we 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 $K_1 = g^{a_1 c}$. Essentially, 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 that

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

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

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

To see the equation above, we use $\mathcal{D}$ to build another algorithm $\mathcal{D}_1$ to $(t_1, \epsilon_{\text{ddh}})$-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 $\sigma \leftarrow \text{PRF}_{K_d}(M)$. Given a verification query (d, M, σ), it computes and returns $\sigma \overset{?}{=} \text{PRF}_{K_d}(M)$. When $\mathcal{D}$ submits its challenge message M^*, $\mathcal{D}_1$ chooses a random bit b, and returns $\sigma^* \leftarrow \text{PRF}_{K_b}(M^*)$. 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

$$\left|\Pr\left[\delta = 1 | Z = g^{a_0 c}\right] - \Pr\left[\delta = 1 | Z \overset{\$}{\leftarrow} \mathbb{G}\right]\right| \quad = \quad |\Pr[X_0] - \Pr[X_1]|$$

Therefore, we obtain the equation (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

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

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 $\text{PRF}_{K_0}(M)$. We have

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

To prove the equation above, we construct an algorithm $\mathcal{D}_2$ to $(t_2, \epsilon_{\text{prf}})$-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 and returns $\sigma \leftarrow \text{PRF}_{K_1}(M)$. 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 $\sigma' \overset{?}{=} \sigma$; if $d = 1$, it computes and returns $\sigma \overset{?}{=} \text{PRF}_{K_1}(M)$. 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 $\sigma^* \leftarrow \text{PRF}_{K_1}(M^*)$. $\mathcal{D}_2$ 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 equation (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 equations (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. □

5 SDVS without Delegatability

It is clear that our SDVS scheme based on pseudorandom functions is *delegatable*. The signer S or the verifier V can simply release $K = g^{x_s x_v}$ to any third party so that it can sign any message on behalf of S for V. Below in Fig. 2 we give another construction of SDVS, called SDVS_2, which does not suffer from the delegatability issue but at the price of provable security in the random oracle model only, where $\mathbb{G}$ is a multiplicative group of large prime order q, g is its generator, and $\mathtt{H} : \{0,1\}^* \times \mathbb{G}^3 \to \mathbb{Z}_q$ is a collision-resistant hash function which will be modeled as a random oracle in the security proofs.

The idea behind the construction above is that in order to sign a message M, the signer S produces a signature of knowledge showing that it knows the secret key of either S or V, i.e.

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

So either $\mathtt{sk}_s$ or $\mathtt{sk}_v$ must be involved in the generation of σ. To make the scheme a 'strong' DVS, we propose 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

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

Security Analysis. For the security of SDVS_2, we have the following theorems.

Theorem 5.1. *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*

$$t_{\mathrm{uf}} \approx t_{\mathrm{dl}}, \quad \textit{and} \quad \epsilon_{\mathrm{uf}} \leq \sqrt{32 q_{\mathtt{H}} \epsilon_{\mathrm{dl}}}$$

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

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

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

```
$\mathsf{Sim}(\mathtt{sk}_v, \mathtt{pk}_s, \mathtt{pk}_v, M)$:
    parse $\mathtt{sk}_v$ as $x_v$
    select at random $r_v, z_s, c_s \xleftarrow{\$} \mathbb{Z}_q$
    set $R_v \leftarrow g^{r_v}$, $R_s \leftarrow g^{z_s}\mathtt{pk}_s^{-c_s}$, $K \leftarrow \mathtt{pk}_s^{x_v}$
    set $c \leftarrow \mathtt{H}(M, K, R_s, R_v)$
    set $c_v \leftarrow c - c_s$, $z_v \leftarrow r_v + c_v x_v$
    return $\sigma := (c_s, z_s, c_v, z_v)$
```

Figure 2: Our Non-delegatable SDVS Scheme, SDVS_2

Again, as shown in Theorem 5.2, SDVS_2 is perfectly non-transferable, thus we don't need to handle the adversary's queries to oracle $\mathcal{O}_{\mathsf{Sim}}$, it is sufficient for us to merely consider how to deal with signing queries, verification queries and hash queries in the proof, which is deferred to Appendix A.

Remark 3 : Unlike DVS-KW in [19], the unforgeability of DVS-KW is proved in the *non-programmable random oracle model* [21], while the proof above is in the *programmable* random oracle model. Besides, the security reduction above is not tight, since we need to rewind the adversary in order to extract the discrete logarithm from the signatures. However, these losses are expectable, because the unforgeability of our scheme relies on the hardness of discrete logarithm problem, which is believed to be much harder than DDH problem, the underlying problem of the unforgeability of DVS-KW.

Note that in SDVS_2, both S and V can produce identically distributed proofs using their respective secret key, because (the interactive version of) the proof of knowledge is perfect special honest-verifier zero-knowledge and both S and V can compute the same key K. Therefore, we have the following theorem immediately.

Theorem 5.2. SDVS_2 *is perfectly non-transferable.*

Theorem 5.3. *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, one must have that $c_s + c_v = \mathtt{H}(M, K, g^{z_s}\mathtt{pk}_s^{-c_s}, g^{z_v}\mathtt{pk}_v^{-c_v})$ and $\bar{c}_s + \bar{c}_v = \mathtt{H}(M, K, g^{\bar{z}_s}\mathtt{pk}_s^{-\bar{c}_s}, g^{\bar{z}_v}\mathtt{pk}_v^{-\bar{c}_v})$, from which have that

$$\begin{aligned} z_s &= r_s + x_s c_s \quad (11) & \qquad \bar{z}_s &= r_s + x_s \bar{c}_s \quad (13) \\ z_v &= r_v + x_v c_v \quad (12) & \qquad \bar{z}_v &= r_v + x_v \bar{c}_v \quad (14) \end{aligned}$$

If $c_s \neq \bar{c}_s$, then from equations (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 equations (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 $1/q_{\mathtt{H}}$ to the success probability of the extractor. Conditioned on that the random guess is correct, by a similar analysis to that in [22, 5], 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 $(\epsilon - 1/q)/16q_{\mathtt{H}}$. □

Theorem 5.4. *If GDH assumption $(t_{\text{gdh}}, \epsilon_{\text{gdh}})$-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*

$$t \approx t_{\text{gdh}} \quad \textit{and} \quad \epsilon \leq \epsilon_{\text{gdh}} + \frac{2}{q}$$

We defer the proof to Appendix A.

6 Comparison

In Table 1 we compare our schemes with some existing (S)DVS schemes in the literature, where we denote the size of a signature and the size of a public key by '|sig|' and '|pk|', respectively; by 'strong' we mean if the scheme supports privacy of signer's privacy; by 'non-dele' we mean if the scheme supports non-delegatability; and by 'model' we mean that the security of the scheme is in the random oracle model (rom) or the standard model (std).

scheme	**|sig|**	**|pk|**	**strong**	**non-dele**	**model**
[14]	$2\|\mathbb{Z}_p\| + 3\|\mathbb{Z}_q\|$	$1\|\mathbb{Z}_p\|$	×	×	rom
[18]	$n_r + 1\|\mathbb{G}_T\|$	$1\|\mathbb{G}\|$	√	×	rom
[19]	$4\|\mathbb{Z}_q\|$	$2\|\mathbb{Z}_p\|$	×	√	rom
[13]	$\|\mathtt{H}(\cdot)\|$	$1\|\mathbb{Z}_p\|$	√	×	rom
SDVS_1	$\|\mathrm{PRF}_K(\cdot)\|$	$1\|\mathbb{Z}_p\|$	√	×	**std**
SDVS_2	$4\|\mathbb{Z}_q\|$	$1\|\mathbb{Z}_p\|$	√	√	rom

Table 1: Comparison with other schemes.

From the comparison one can see that in terms of signature size, SDVS_1 which is secure in the standard model, outperforms schemes in [14, 18, 19] which are secure in the random oracle model, and is comparable with the scheme in [13] which so far is the most efficient SDVS scheme in the random oracle model. Besides, SDVS_1 has almost the same computational efficiency as [13]. On the other hand, to the best of our knowledge, SDVS_2 is the first SDVS scheme that is non-delegatable, which compared with Lipmaa-Wang-Bao DVS scheme [19], has the same signature size and shorter public key.

7 Conclusions and Future Work

In this paper we proposed the a pretty efficient SDVS scheme that has quite short signatures, and is provably secure in the standard model. Since it is vulnerable to delegatability attacks, we then proposed the first non-delegatable SDVS scheme which is as efficient as Lipmaa-Wang-Bao DVS scheme, at the price of provable security in the random oracle model.

Many problems about DVS and its variants are still left unsolved, i.e. the construction of non-delegatable DVS and variants in the standard model. It seems that non-delegatable DVS without random oracles is much harder to achieve. The user secret key in most of the signature schemes in the literature is comprised of element(s) of the group $\mathbb{Z}_q$, and usually is put on the exponent position of the public key. As far as we know there is no method for extracting the exponent from element(s) in the underlying group $\mathbb{G}$ without resorting to the random oracle heuristic. We leave this problem as our future work.

References

[1] M. Bellare and O. Goldreich. On defining proofs of knowledge. In *CRYPTO92*, volume 740 of *LNCS*, pages 390–420. Springer, 1992.

[2] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT00*, volume 1976 of *LNCS*, pages 531–545. Springer, 2000.

[3] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS*, pages 62–73. ACM, 1993.

[4] D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT04*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.

[5] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO04*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[6] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004. A preliminary version appeared in Asiacrypt 2001.

[7] D. Chaum and H. van Antwerpen. Undeniable signatures. In *CRYPTO89*, volume 435 of *LNCS*, pages 212–216. Springer, 1989.

[8] I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO91*, volume 576 of *LNCS*, pages 445–456. Springer, 1991.

[9] C. Gentry and A. Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT02*, volume 2501 of *LNCS*, pages 548–566. Springer, 2002.

[10] Q. Huang, W. Susilo, and D. S. Wong. Non-delegatable identity-based designated verifier signature. Cryptology ePrint Archive, Report 2009/367, 2009. `http://eprint.iacr.org/`.

[11] X. Huang, W. Susilo, Y. Mu, and W. Wu. Universal designated verifier signature without delegatability. In *ICICS06*, volume 4307 of *LNCS*, pages 479–498. Springer, 2006.

[12] X. Huang, W. Susilo, Y. Mu, and W. Wu. Secure universal designated verifier signature without random oracles. *International Journal of Information Security*, 7(3):171–183, 2007.

[13] X. Huang, W. Susilo, Y. Mu, and F. Zhang. Short designated verifier signature scheme and its identity-based variant. *International Journal of Network Security*, 6(1):82–93, 2008.

[14] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT96*, volume 1070 of *LNCS*, pages 143 – 154. Springer, 1996.

[15] J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. In *CCS*, pages 155–164. ACM, 2003.

[16] F. Laguillaumie, B. Libert, and J.-J. Quisquater. Universal designated verifier signatures without random oracles or non-black box assumptions. In *SCN06*, volume 4116 of *LNCS*, pages 63–77. Springer, 2006.

[17] F. Laguillaumie and D. Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In *SCN04*, volume 3352 of *LNCS*, pages 105–119. Springer, 2004.

[18] F. Laguillaumie and D. Vergnaud. Multi-designated verifiers signatures. In *ICICS04*, volume 3269 of *LNCS*, pages 495–507. Springer, 2004.

[19] H. Lipmaa, G. Wang, and F. Bao. Designated verifier signature schemes: Attacks, new security notions and a new construction. In *ICALP05*, volume 3580 of *LNCS*, pages 459–471. Springer, 2005.

[20] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004.

[21] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO02*, volume 2442 of *LNCS*, pages 111–126. Springer, 2002.

[22] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000.

[23] R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In C. Boyd, editor, *ASIACRYPT01*, volume 2248 of *LNCS*, pages 552–565. Springer, 2001.

[24] S. Saeednia, S. Kremer, and O. Markowitch. An efficient strong designated verifier signature scheme. In *ICISC03*, volume 2971 of *LNCS*, pages 40–54. Springer, 2003.

[25] H. Shacham and B. Waters. Efficient ring signatures without random oracles. In T. Okamoto and X. Wang, editors, *PKC07*, volume 4450 of *LNCS*, pages 166–180. Springer, 2007.

[26] R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk. Universal designated-verifier signatures. In *ASIACRYPT03*, volume 2894 of *LNCS*, pages 523–542. Springer, 2003.

[27] R. Steinfeld, H. Wang, and J. Pieprzyk. Efficient extension of standard Schnorr/RSA signatures into universal designated-verifier signatures. In *PKC04*, volume 2947 of *LNCS*, pages 86–100. Springer, 2004.

[28] W. Susilo, F. Zhang, and Y. Mu. Identity-based strong designated verifier signature schemes. In *ACISP04*, volume 3108 of *LNCS*, pages 313–324. Springer, 2004.

[29] D. Vergnaud. New extensions of pairing-based signatures into universal designated verifier signatures. In *ICALP06*, volume 4052 of *LNCS*, pages 58–69. Springer, 2006.

[30] B. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT05*, volume 3494 of *LNCS*, pages 114–127. Springer, 2005.

[31] R. Zhang, J. Furukawa, and H. Imai. Short signature and universal designated verifier signature without random oracles. In *ACNS05*, volume 3531 of *LNCS*, pages 483–498. Springer, 2005.

A Security Proofs of SDVS_2

(Proof of Theorem 5.1). Let $\mathcal{F}$ be a forger algorithm 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 then 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 $r_s, z_v, c_v \in \mathbb{Z}_q$, and computes $R_s \leftarrow g^{r_s}$, $R_v \leftarrow g^{z_v}\mathtt{pk}_v^{-c_v}$. It sets $c \leftarrow \mathtt{H}(M, \mathtt{pk}_v^{\bar{x}}, R_s, R_v)$ and $c_s \leftarrow c - c_v$, $z_s \leftarrow r_s + c_s\bar{x}$. The signature $\sigma := (c_s, z_s, c_v, z_v)$ is then returned to $\mathcal{F}$.

Verification Queries: Given a message M and an alleged signature $\sigma := (c_s, z_s, c_v, z_v)$, $\mathcal{A}$ 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 if $c_s + c_v = \mathtt{H}(M, \mathtt{pk}_v^{\bar{x}}, R_s, R_v)$. 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

$$x = \log_g y = \frac{z_v^* - \bar{z}_v^*}{c_s^* - \bar{c}_s^*}$$

Otherwise, it aborts.

(**Probability Analysis**): It is readily seen that 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 a similar analysis with that in [22, 5] 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 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. □

(Proof of Theorem 5.4). 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 is at most $1/q$ different from the real one.

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

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

This completes the proof. □