2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering

Constructing Strong Designated Verifier Signatures from Key Encapsulation Mechanisms

Borui Gong*, ✉ Man Ho Au* and Haiyang Xue†*
* *Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China*
csbgong@comp.polyu.edu.hk, csallen@comp.polyu.edu.hk
† *State Key Laboratory of Information Security, IIE, CAS, Beijing, China*
haiyangxc@gmail.com

***Abstract*—A designated verifier signature (DVS) allows a signer to convince a verifier that a message has been endorsed in a way that the conviction cannot be transferred to any third party. This is achieved by the property that the signature can be generated by one of them. Since DVS is publicly verifiable, a valid DVS implies that the signature must be created by either the signer or the verifier. To enhance privacy of signers' identity, a strong DVS (SDVS) disallows public verification.**

In this paper, we investigate various aspects of SDVS with making two contributions. Firstly, we consider SDVS in the multi-user setting and propose two strengthened models, namely, multi-user and multi-user^{+}. To illustrate the significance of our models, we show that it is possible to forge an SDVS when the attacker is given signatures from an honest signer to multiple dishonest verifiers. Secondly, we give a generic construction of SDVS from Key Encapsulation Mechanism (KEM) and Pseudorandom Function (PRF) in the standard model. Our generic construction is secure in the multi-user setting if the underlying KEM and PRF are secure. We also give instantiations based on DDH and LWE assumptions respectively.

***Keywords*-Signature, SDVS, Standard Model, KEM, Post-Quantum**

I. Introduction

The concept of undeniable signature was first proposed by Chaum et al. [6]. It consists of a signer named Alice and a verifier named Bob. When Bob wants to verify the signature created by Alice, he must interact with Alice through an interactive verification protocol. This means that the verifier cannot check the validity of signature by himself. In other words, the signer has complete control of the signature in order to avoid other undesirable verifiers from getting convinced of its validity. However, because of blackmailing [8] and mafia [7] attacks, undeniable signatures may not always achieve their goals.

Motivated by the need to give signer control over who could verify his/her signatures, Jakobsson et al. [14] proposed a designated verifier signature (DVS) scheme with briefly discussing the concept of strong designated verifier signature (SDVS). Their DVS scheme is the first non-interactive undeniable signature scheme by using designated verifier proof. In their scheme, only designated verifier can be convinced by the signature's validity or invalidity without requiring any interaction with the presumed signer. This scheme follows a very simple approach: each user holds two key pairs, one for generating signatures while the other for encrypting signatures. When Alice (signer) wants to generate a signature to Bob (verifier), she first uses her signing key to generate a signature, followed by encrypting it under Bob's encryption key. Once Bob receives the signature, he decrypts it first and verifies its validity. This simple approach requires an encryption followed by a verification, which is therefore less efficient than desired.

The notion of SDVS was first formalized by Saeednia et al. [21]. The concept of privacy of signer's identity (PSI) was then formalized by Laguilaumie et at. [15]. It means that no third party can distinguish which signer generates the signature without verifier's secret key, which actually captures property in strong designated verifier signature.

Since its introduction, many SDVS schemes have been proposed. Huang et al. [13] proposed the first short designated verifier signature scheme with its identity-based variant. Huang et al. [11] proposed the first SDVS scheme in standard model, based on DDH problem. Subsequently, new schemes under various assumptions (e.g. DBDH, CDH, GDH, $\mathcal{R}$-SIS) have been proposed [1], [5], [23]. However, they are under specific hardness assumptions. Their security are analyzed in single-user setting. That is, the attacker is given the public keys of the target signer and verifier, and may issue queries with respect to these two entities.

In this paper, we initiate the study of SDVS in the multi-user setting. We observe that existing models may not capture attacks in practice. Specifically, adversary may have access to signatures generated for different designated verifiers; Furthermore, adversary may obtain useful information from signatures generated by other signers on the target verifier. In other words, the adversary may produce valid signatures collaborating with dishonest signers. However, these attacks are not captured by the existing models where adversary is restricted to issue queries with respect to the target signer and verifier. Indeed, we give an example below to show that a secure SDVS scheme in the existing models

✉ corresponding author

2324-9013/19/$31.00 ©2019 IEEE
DOI 10.1109/TrustCom/BigDataSE.2019.00084

can be broken in the first case.

Consider the following SDVS scheme. Let G, G_T be cyclic groups of the same order and $\hat{e} : G \times G \rightarrow G_T$ be a bilinear pairing between these groups. Let g be a generator of G. Further assume $H : \{0,1\}^* \rightarrow G$ be a hash function from $\{0,1\}^*$ to G. The public-secret key pair of our example scheme is set as $(pk, sk) = (g^x, x)$.

When the signer (with public/secret key (pk_s, sk_s)) generates a signature σ for the designated verifier (with public/secret key (pk_v, sk_v)) on message m, it computes signature $\sigma = \hat{e}(pk_v^{sk_s}, H(m))$. This signature's validity can be verified by the verifier through checking if $\sigma \stackrel{?}{=} \hat{e}(pk_s^{sk_v}, H(m))$. It can be proven easily that the scheme is unforgeable in the single-user setting under the BDH hardness assumption, where H is a random oracle. In addition, it enjoys PSI since identifying the actual signer implies solving the DBDH problem.

In practice, however, the signer may generate signatures for different designated verifiers. If the attacker is given signatures for verifiers whose keys are under its control, our example scheme is not secure any more. To be more specific, assume the attacker would like to forge on message m for verifier pk_v. The attacker may first creates a "rouge key" of the form $pk_v' = (pk_v)^k$ for some randomly chosen k. He may request a signature from the signer on message m with respect to pk_v'. The signature is of the form $\sigma = \hat{e}(pk_v^{sk_s}, H(m))$. The attacker can compute $\sigma' = \sigma^{\frac{1}{k}}$ as a valid forgery on m under pk_v. In other words, an SDVS scheme secure in the existing model may not be sufficient when the scheme is used in a more complicated situation, with involving multiple users in practice.

To address this issue, we strengthen existing models and propose two models, namely, multi-user and multi-user$^+$. In our first model (multi-user), adversary can issue queries from given lists of signers and verifiers, and also corrupt them; In our second model (multi-user$^+$), adversary can obtain signatures from the signer on any verifiers of its choice (i.e. the verifier's public keys are created by the adversary).

We also propose a generic construction of SDVS from *KEM* and *PRF*. We prove that our generic construction gives secure SDVS in the multi-user (resp. multi-user$^+$) model provided that the underlying *KEM* is IND-CPA (resp. IND-CCA) secure[1] and that *PRF* is pseudorandom. It means that any progress made in *KEM* implies advances in SDVS based on our generic construction. We would like to remark that our generic construction does not rely on the random oracle model so we can base on *KEM* schemes with different features. We also give two instantiations based on DDH assumption and two lattice-based versions.

[1]Looking ahead, we also require that the encapsulation process of the underlying *KEM* can be separated into two phases in which the first phase is independent from the receiver's public key. We observe that many existing *KEM* fulfil this requirement.

A. Our Contributions

We proposed strengthened security models of SDVS in the multi-user setting. We also proposed a generic approach to construct SDVS in the enhanced models. Specifically, we made the following contributions.

- We proposed two security models of SDVS to model security requirements in the multi-user setting, namely, multi-user and multi-user$^+$.
- We proposed a generic construction of SDVS from *KEM* and *PRF*. We proved that our generic construction is secure in the standard model assuming the security of the underlying *KEM* and *PRF*.

Table I summarizes differences between the existing SDVS security models and our two models. Let S and V denote lists of signers and verifiers' public keys chosen by the challenger. We can see from table I that in the existing models, the adversary can only issue queries with respect to the specific challenge signer and verifier. In our enhanced models, the adversary can make additional queries beyond the challenge identities or can issue queries on the verifier chosen by himself adaptively. In other words, the adversary can access more information in our models than in the original model. Our models can also corrupt queries, meaning that the adversary can request for privacy keys of any public keys in S and V except the challenge public keys (also chosen by the adversary).

B. Related Work

Motivated by undeniable signature [6] which was proposed in 1989, Jakobsson et al. [14] proposed the notion of *designated verifier signature* (DVS) and the concept of *strong designated verifier signature* (SDVS).

In the study of SDVS, Shahrokh et al. [21] first presented its formal definition in 2003. Susilo et al. [22] introduced a variant in the identity-based setting. It was further enhanced in [10], [13] with additional features. Huang et al. [12], [13] proposed its efficient variants respectively by using short signatures. Huang et al. [11] also proposed an efficient variant in 2011. Hou et al. [9] proposed a designed designated verifier transitive signature. Additional features like non-delegatability have been considered in [1], [17], [24]. Geotae [20] introduced the first lattice-based construction in the standard model in 2016.

C. Outline

This paper is organized as followed. In the next section, we relate the underlying *KEM* and *PRF* schemes with their security requirements and the definition of DVS. In section III, we give definitions on SDVS with our extended models. We then give our generic construction in section IV, followed by the security proofs in section V. In section VI, we give four instantiations based on our construction. We compare our instantiations with the existing SDVS schemes. We give our conclusion in the end of this paper.

Table I
DIFFERENCES BETWEEN EXISTING MODELS AND OUR MODELS

	Challenge Public Keys (pk_s, pk_v)	Signature Queries (pk_s, pk_v)
Existing Model	$pk_s \in S, pk_v \in V$, $\lvert S\rvert = 1, \lvert V\rvert = 1$	$pk_s \in S, pk_v \in V$, $\lvert S\rvert = 1, \lvert V\rvert = 1$
Multi-user	$pk_s \in S, pk_v \in V$	$pk_s \in S, pk_v \in V$
Multi-user$^+$	$pk_s \in S, pk_v \in V$	$pk_s \in S$, no restriction on pk_v

S and *V* indicate signers and verifiers' public key lists chosen by the challenger
pk_s and pk_v indicate signer and verifier's public keys respectively

II. PRELIMINARIES

Definition 1 (KEM): A standard *key encapsulation mechanism* (*KEM*) consists of the following three PPT algorithms.

- **KeyGen:** The randomized key generation algorithm returns public/secret key pair (pk, sk) with input 1^k, where k is a security parameter. This algorithm can be expressed as, $KeyGen(1^k) \to (pk, sk)$.
- **Encap:** The encapsulation algorithm takes public key as input, returning key K with its encapsulation C. It can be written as, $Encap(pk) \to (K, C) \in K_{pk} \times C_{pk}$.
- **Decap:** The decapsulation algorithm takes secret key sk and encapsulation C as input. It returns corresponding key K or outputs $\perp$ to indicate invalid encapsulation. It can be written as, $Decap(C, sk) = K$ or $\perp$.

Definition 2 (2-Phase KEM): We call a *KEM* scheme as a 2-phase *KEM* if its *Encap* algorithm can be divided into the following two phases.

- ***Encap***^{1} : It will first choose a random value $w \xleftarrow{\$} Q$ and output C, it can be written as, $Encap^1(w) \to C$.
- ***Encap***^{2} : In the second phase, it takes C, public key pk and w as input. It finally returns K, whose encapsulation is C. It can be written as, $Encap^2(C, pk, w) \to K$.

Definition 3 (Security of KEM): We call a *KEM* scheme is (t, ϵ_{cpa})-CPA (resp. (t, q_d, ϵ_{cca})-CCA) secure if there does not exist such a PPT adversary who can win the following game in time t with at least ϵ_{cpa} (resp. ϵ_{cca}) advantage (resp. after making q_d decryption queries). The game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is as follows.

1) **Setup**: By inputing security parameter k, challenger $\mathcal{C}$ generates a pair of keys $(pk, sk) \leftarrow KeyGen(1^k)$ and gives pk to the adversary $\mathcal{A}$.
2) **Phase 1** (Only in CCA game): In this phase, adversary $\mathcal{A}$ submits a string C_i to decapsulation oracle $\mathcal{O}_{dec}$. The oracle will return decapsulation result $dec_{sk}(C_i)$.
3) **Challenge**: In the challenge phase, $\mathcal{A}$ issues encapsulation queries to $\mathcal{C}$. Encapsulation oracle $\mathcal{O}_{enc}$ randomly selects $b \in \{0, 1\}$ and computes $(C^*, K^*) \leftarrow Encap(pk)$. Challenger $\mathcal{C}$ will return (C^*, K^*) if $b = 0$; otherwise, it will return (C^*, K') where $K' \xleftarrow{\$} \{0, 1\}^{|K^*|}$. ($C^*$ is called target ciphertext)
4) **Phase 2** (Only in CCA game): Phase 2 is the same as Phase 1 with the restriction that submitted encapsulation query C_i should not be identical to C^*.
5) **Guess**: $\mathcal{A}$ outputs a guess b' of b and wins the game if $b' = b$. The advantage of $\mathcal{A}$ in winning this game is defined as
$$\epsilon_{cpa}(resp.\ \epsilon_{cca}) = 2(\Pr[b' = b] - \frac{1}{2}).$$
The scheme is secure if ϵ_{cpa} (resp. ϵ_{cca}) is negligible.

Definition 4 (PRF): Assuming that the inputs of the *pseudorandom function* (*PRF*) we considered here can be arbitrary. Let $\{0, 1\}^l$ be its output. Let $\mathbf{F} = \{PRF_k\}_{k \in N}$ be a function set such that any variable PRF_k assumes values in the set of $\{0, 1\}^* \to \{0, 1\}^l$. $\mathbf{F}$ is called an efficiently computable pseudorandom function ensemble if

1) (*efficient computation*) I and V are PPT algorithms and there is a mapping function ϕ, mapping from strings to functions, such that $\phi(I(1^k))$ and PRF_k are identically distributed and $V(i, x) = (\phi(i))(x)$.
2) ((t, ϵ_{prf})-*pseudorandomness*) For any PPT distinguisher $\mathcal{D}$, he can not distinguish a *PRF* function to a real random function with negligible probability.
$$\left|\Pr[\mathcal{D}^{PRF_k}(1^k) = 1] - \Pr[\mathcal{D}^{RF_k}(1^k) = 1]\right| < \epsilon_{prf}$$
where $\mathbf{R} = \{RF_k\}_{k \in N}$ is the set involving RF_k. RF_k is uniformly distributed over $\{0, 1\}^* \to \{0, 1\}^l$.

Definition 5 (DVS): A *designated verifier signature* (DVS) consists of the following three PPT algorithms.

- **KG:** The key generation algorithm takes 1^k as input where k is security parameter, followed by returning a public/secret key pair (*pk, sk*). This algorithm can be written as, $KG(1^k) \to (pk, sk)$.
- **Sign:** The signing algorithm takes message m, signer's public and secret keys $(pk_s,\ sk_s)$ and designated verifier's public key pk_v as input. It will return signature σ of message m, which can be written as $Sign(sk_s, pk_s, pk_v, m) \to \sigma$.
- **Ver:** The verification algorithm takes signature σ, corresponding message m, verifier's public and secret keys (sk_v, pk_v) and signer's public key pk_s as input. It will output 1 if it is a valid signature, otherwise it will output 0. It can be written as $Ver(sk_v, pk_v, pk_s, m, \sigma) \to b$ (b is 1 if the signature is valid, otherwise b is 0).

Correctness: The correctness of DVS requires that for any $KG(1^k) \to (pk_s, sk_s)$, $KG(1^k) \to (pk_v, sk_v)$ and any message $m \in \{0, 1\}^*$, we have the following,
$$\Pr[Ver(sk_v, pk_v, pk_s, m, Sign(sk_s, pk_s, pk_v, m)) = 1] = 1.$$

III. Our Strengthened models

In this section, we present our strengthened models which allow the attacker to issue queries with respect to multiple verifiers, some of which may be corrupted or with keys chosen adversarially. The difference is summarized in Table I. Formally, our strengthened models are defined as followed.

Definition 6 (Unforgeability): An SDVS scheme is *unforgeable* in multi-user (resp. multi-user$^+$) setting if no PPT adversary can forge a valid signature on a message of its choice without knowing the signer and verifier's secret key.

The following game between challenger $\mathcal{C}$ and PPT adversary $\mathcal{A}$ formally defines unforgeability.

1) **Setup:** On input security parameter k, $\mathcal{C}$ runs *KG* algorithm to obtain multiple signers and the verifiers' public-secret key pairs. Let $S = \{pk_{s_1}, pk_{s_2}..., pk_{s_m}\}$ and $V = \{pk_{v_1}, pk_{v_2}..., pk_{v_n}\}$ be signers and verifiers' public keys respectively. $\mathcal{A}$ is given *S* and *V*.
2) **Queries:** $\mathcal{A}$ can issue queries to the following oracles. Note that $\mathcal{A}$ can also issue a corrupt query to obtain the secret keys of signer and verifier in the lists (except the challenge public keys, i.e. pk_s^* and pk_v^*).
   - $\mathcal{O}_{sign}$: $\mathcal{A}$ can issue signing queries between signer $pk_s \in S$ and verifier pk_v.
   - $\mathcal{O}_{sim}$: $\mathcal{A}$ can request verifier pk_v to simulate signature on message m between signer $pk_s \in S$.
   - $\mathcal{O}_{ver}$: $\mathcal{A}$ can request verification queries on the pair (m, σ) on the signer $pk_s \in S$ and verifier pk_v.
   - *Restrictions:* In the multi-user setting, an additional restriction applies, namely, $pk_v \in V$ for queries to $\mathcal{O}_{sign}$, $\mathcal{O}_{sim}$, $\mathcal{O}_{ver}$. In the multi-user$^+$ setting, pk_v can be any value chosen by $\mathcal{A}$.
3) **Forgery:** Finally, $\mathcal{A}$ outputs a forgery (m^*, σ^*) on signer and verifier from lists and wins the game if,
   - $Ver(sk_v^*, pk_v^*, pk_s^*, m^*, \sigma^*) = 1$, and
   - $\mathcal{A}$ has not issued $\mathcal{O}_{sign}$ and $\mathcal{O}_{sim}$ on input m^* on signer pk_s^* and verifier pk_v^* before.

   The probability of forging a valid signature is denoted by $\Pr[Forge]$. An SDVS scheme is unforgeable if

$$\Pr[Forge] < \epsilon(k),$$

where $\epsilon(k)$ is negligible.

Definition 7 (Non-Transferability): An SDVS scheme is non-transferable if there exists a PPT simulation algorithm *Sim* which takes sk_v, pk_v, pk_s and message m as input. It outputs a simulated signature that is indistinguishable from the real signature generated by the signer on the same m.

That is, for any PPT distinguisher $\mathcal{D}$, any $(pk_s, sk_s) \leftarrow KG(1^k)$, $(pk_v, sk_v) \leftarrow KG(1^k)$ and any message $m \in \{0,1\}^*$, it holds that

$$\left| \Pr\left[\begin{array}{c} \sigma_0 \leftarrow \mathsf{Sign}(sk_s, pk_s, m), \\ \sigma_1 \leftarrow \mathsf{Sim}(sk_v, pk_v, m), \\ b \xleftarrow{\$} \{0,1\}, \\ b' \leftarrow D(pk_s, sk_s, pk_v, sk_v, \sigma_b) \end{array} : b' = b \right] - \frac{1}{2} \right| < \epsilon(k)$$

where $\epsilon(k)$ is a negligible function with security parameter k. The random coins consumed by $\mathcal{D}$ and the probability takes over the randomness used in *KG*, *Sign* and *Sim*. If the probability is equal to $\frac{1}{2}$, we say that the SDVS scheme is *perfectly non-transferable*.

Definition 8 (Privacy of Signer's Identity (PSI)): We call a scheme that satisfies *privacy of signer's identity* (PSI) in the multi-user (resp. multi-user$^+$) setting if a third party cannot tell whether the signature generated by signer S_0 or by signer S_1 correctly without knowing signer's and verifier's secret key.

The game below, which is played by a challenger $\mathcal{C}$ and a distinguisher $\mathcal{D}$, formally defines privacy of signer's identity in the multi-user setting. Let *S* and *V* denote the lists of signers and verifiers' public keys generated by $\mathcal{C}$, same as the unforgeability game.

1) **Setup:** $\mathcal{C}$ generates public and secret keys for signers and verifiers. The corresponding public key lists, namely, *S* and *V*, are given to distinguisher $\mathcal{D}$.
2) **Queries:** $\mathcal{D}$ can adaptively issue $\mathcal{O}_{sign}$, $\mathcal{O}_{sim}$ and $\mathcal{O}_{ver}$ queries on signer pk_{s_i} and verifier pk_{v_i}, same as in the unforgeability game. $\mathcal{D}$ can also corrupt secret keys on signer and verifier from the lists.
3) **Challenge:** $\mathcal{D}$ chooses two signers, e.g. S_0^* and S_1^*, from *S* and one verifier pk_v^* from *V* to be the challenge identities. $\mathcal{D}$ submits a message m^* and $\mathcal{C}$ tosses a coin $b \in \{0,1\}$ and computes challenge signature $\sigma^* \leftarrow \mathrm{Sign}(sk_{s_b}^*, pk_{s_b}^*, pk_v^*, m^*)$. $\mathcal{C}$ then returns σ^* to $\mathcal{D}$.
4) **Queries:** $\mathcal{D}$ continues to issue queries as in step 2 with the restriction that no verification queries on $(m^*, \sigma^*, pk_{s_i}^*)$ for any $pk_{s_i}^* \in \{S_0^*, S_1^*\}$. Note that $\mathcal{D}$ cannot corrupt challenge identities' secret keys.
5) **Guess:** Finally, $\mathcal{D}$ outputs a guess b' of b and wins the game if $b' = b$. The probability of $\mathcal{D}$ in winning this game is defined as $\Pr[PSI]$. An SDVS scheme possesses *PSI* if

$$\left| \Pr[PSI] - \frac{1}{2} \right| < \epsilon(k),$$

where $\epsilon(k)$ is negligible.

Definition 9 (SDVS): An SDVS scheme is secure in the multi-user (resp. multi-user$^+$) setting if it possesses unforgeability, non-transferability and privacy of signer's identity.

IV. Our Construction

A. Overview of Our Construction

Our generic construction of SDVS relies on *KEM* and *PRF*, where *KEM* must be 2-phase as discussed in Definition 2. This is not too restrictive since most *KEM* schemes can satisfy this requirement. Our generic construction is secure in the multi-user setting (resp. multi-user$^+$ setting) if *PRF* is secure and the underlying *KEM* is IND-CPA secure (resp. IND-CCA secure). Below we give a high-level description of our generic construction.

In our construction, we use *KeyGen* in *KEM* to generate signer's keys, i.e. $(pk_s, sk_s) \leftarrow \textit{KeyGen}(1^k)$. We use the first phase in *KEM*'s *Encap*, $C \leftarrow \textit{Encap}^1(w)$, to generate verifier's keys, i.e. $pk_v \leftarrow C$, $sk_v \leftarrow w$ (w is the randomness used in *Encap*^{1}). To sign a message, the signer uses his secret key to decapsulate C to obtain the session key, i.e. $K_{sv} \leftarrow \textit{Decap}(C, sk_s)$. He then uses this key in *PRF* to sign message m, i.e. $\sigma \leftarrow \textit{PRF}_{K_{sv}}(m)$. For verification, verifier executes the second phase in *Encap* to acquire the same session key, i.e. $K_{sv} \leftarrow \textit{Encap}^2(C, pk_s, sk_v := w)$, followed by checking $\sigma \stackrel{?}{=} \textit{PRF}_{K_{sv}}(m)$.

B. Details of Our Generic Construction

Given a *KEM* scheme K = (K.*KeyGen*, K.*Encap*, K.*Decap*), which is a 2-phase encapsulation mechanism, and a *PRF* function, we can construct a secure SDVS scheme D = (D.*KG*, D.*Sign*, D.*Ver*). The construction is as follows.

1) ***D.KG:*** The key generation algorithm takes 1^k as input where k is the security parameter. It invokes *KeyGen* in *KEM* to obtain signer's keys, namely, $K.\textit{KeyGen}(1^k) \rightarrow (D.pk_s, D.sk_s)$. It also invokes the first phase in *Encap* to obtain verifier's keys, namely, $K.\textit{Encap}^1(w) \rightarrow C$, $(C, w) \rightarrow (D.pk_v, D.sk_v)$.
2) ***D.Sign:*** The signing algorithm takes the signer's keys, the verifier's public key and the message as input, namely, ($D.sk_s$, $D.pk_s$, $D.pk_v$, m). It first runs the decapsulation algorithm in *KEM* to obtain the key, i.e. $K.\textit{Decap}(D.pk_v, D.sk_s) \rightarrow K_{sv}$. It then takes key K_{sv} and message m into *PRF* algorithm with returning signature σ, $\textit{PRF}_{K_{sv}}(m) \rightarrow \sigma$. The signing algorithm can be written as, $D.\textit{Sign}(D.sk_s, D.pk_s, D.pk_v, m) \rightarrow \sigma$.
3) ***D.Ver:*** The verification algorithm takes the verifier's keys, public key of signer, the message and signature as input, namely, ($D.sk_v$, $D.pk_v$, $D.pk_s$, m, σ). It first runs the second phase of encapsulation algorithm in *KEM* to compute key K_{sv}, namely, $K.\textit{Encap}^2(D.pk_v, D.pk_s, D.sk_v) \rightarrow K_{sv}$. It will then invoke K_{sv} and message m into *PRF* to obtain its signature σ'. If $\sigma' = \sigma$, it returns 1; otherwise, it returns 0. The whole verification algorithm can be written as, $D.\textit{Ver}(D.sk_v, D.pk_v, D.pk_s, m, \sigma) \rightarrow b$.

V. Security of Our Construction

We prove that our SDVS scheme is secure if the underlying *KEM* and *PRF* schemes are secure.

Theorem 1: If the underlying *KEM* scheme is IND-CPA (resp. IND-CCA) secure and *PRF* function achieves pseudoramdomness, then we can construct a secure SDVS scheme in the multi-user setting (resp. multi-user$^+$ setting).

The proof of Theorem 1 is divided into the proof of the following three lemmas, which stated that our generic construction possesses unforgeability, non-transferability and privacy of signer's identity.

Lemma 1: If the underlying *KEM* scheme is IND-CPA (resp. IND-CCA) secure and *PRF* is a pseudorandom function, then our constructed scheme D achieves the property of unforgeability (in the multi-user setting) (resp. multi-user$^+$ setting). That is, $\Pr^{SDVS}_{\mathcal{A},D}[\textit{Forge}]$ is negligible.

Lemma 2: If the underlying *KEM* is IND-CPA (resp. IND-CCA) secure and *PRF* achieves pseudorandomness, our constructed scheme D is perfectly non-transferable.

Lemma 3: If the underlying *KEM* scheme is IND-CPA (resp. IND-CCA) secure and *PRF* is a pseudorandom function, then our constructed scheme D (with multi-user setting) (resp. multi-user$^+$ setting) achieves the property of privacy of signer's identity (PSI). That is, $\Pr^{SDVS}_{\mathcal{A},D}[\textit{PSI}]$ is no larger than $\frac{1}{2}$, indicating that the adversary has no advantage compared with randomly guessing the bit.

As we will see in the proof of Lemma 2, the scheme is perfectly non-transferable so the queries to $\mathcal{O}_{\textit{sim}}$ can be perfectly handled by $\mathcal{O}_{\textit{sign}}$ in the game of unforgeability and privacy of signer's identity. Hence, we only consider signing and verification queries in these two games.

Proof: (of Lemma 1) For any PPT forger $\mathcal{A}$, $\Pr[\textit{Forge}]$ in the multi-user (resp. multi-user$^+$) setting is negligible assuming *KEM* scheme is IND-CPA (resp. IND-CCA) secure and *PRF* achieves pseudorandomness.

We prove this lemma by using a sequence of games played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. Let G_i denote the i-th game and X_i imply the event that $\mathcal{A}$ outputs a valid forgery in game G_i. Let *S* and *V* denote two lists for signers and verifiers, with m and n entities respectively.

$\mathbf{G_0}$: This game is with multi-user setting (resp. multi-user$^+$). Challenger $\mathcal{C}$ invokes adversary with *S* and *V* lists. Adversary $\mathcal{A}$ can issue signing and verification queries on the signer and verifier from the lists (resp. no restrictions on the verifier in multi-user$^+$). We can have that,

$$\Pr^{SDVS}_{\mathcal{A},D}[\textit{Forge}(\textit{multi-user})](\textit{resp. multi-user}^+) = \Pr[X_0]. \tag{1}$$

$\mathbf{G_1}$: In this game, the key used in *PRF* between challenge identities pk_s^* and pk_v^* is randomly chosen, i.e. $K' \xleftarrow{\$} N$. When $\mathcal{A}$ issues signing and verification queries between them, $\mathcal{C}$ uses this key K' to response. The only difference between this game and game 0 is the key used in *PRF*. If the adversary can distinguish these two games, we can construct an adversary $\mathcal{A}_1$ to break the IND-CPA (resp. IND-CCA) game in *KEM*. Therefore, we have,

$$|\Pr[X_1] - \Pr[X_0]| \leq mn \cdot \epsilon_{\textit{cpa}}(\textit{resp. } \epsilon_{\textit{cca}}). \tag{2}$$

We construct adversary $\mathcal{A}_1$ in CPA (resp. CCA) game where $\mathcal{A}_1$ is given challenge ciphertext (C^*, K^*) and public key pk^*. $\mathcal{A}_1$ will simulate the game for the adversary in SDVS. He randomly guesses a signer-verifier pair from the two lists as challenge identities and sets, $pk_s^* = pk^*$, $pk_v^* = C^*$. The key used between these identities is K^*. Note that $\mathcal{A}_1$ will abort if he cannot guess the challenge identities correctly.

- In the multi-user$^+$ setting, $\mathcal{A}$ can issue queries on verifier pk_{v_i} beyond the V list. To response this query, $\mathcal{A}_1$ will make decapsulation queries on $C_i \leftarrow pk_{v_i}$ in CCA game (under pk_s^*) and get the corresponding key K_i. He then uses this key to response signing and verification queries.

If the SDVS's adversary successfully forges the signature, $\mathcal{A}_1$ outputs 0, indicating that K^* is the correct shared key; Otherwise $\mathcal{A}_1$ randomly outputs a bit b'. Hence his probability to win the game is $mn \cdot \epsilon_{cpa}$ (resp. $mn \cdot \epsilon_{cca}$). Therefore, the difference between game 0 and game 1 is equal to mn times the advantage that $\mathcal{A}_1$ can distinguish them in CPA (resp. CCA) game.

$\mathbf{G_2}$: In this game, we replace *PRF* with a truly random function. It means that the signature is randomly chosen from $\{0,1\}^l$ in this game. We have,

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

To obtain the above equation, we construct an adversary $\mathcal{A}_2$ to break the pseudorandomness of *PRF* with advantage ϵ_{prf}. Given an oracle function $F(\cdot)$ which is either a pseudorandom function chosen from $\mathbf{F}$ or a truly random function. Here, $\mathcal{A}_2$ maintains a table T, which is initially empty.

When responding to signing queries on m_i between pk_s^* and pk_v^*, $\mathcal{A}_2$ returns σ_i if (m_i, σ_i) exists in T; Otherwise, $\mathcal{A}_2$ submits message m_i to function F and returns σ_i, followed by storing it in table. When responding to verification queries on (m_i, σ_i), $\mathcal{A}_2$ will just return $\sigma_i' \stackrel{?}{=} \sigma_i$ if m_i exists in the table with σ_i'; Otherwise, $\mathcal{A}_2$ forwards message m_i to function F with obtaining a signature σ_i'. He then returns $\sigma_i' \stackrel{?}{=} \sigma_i$ to adversary $\mathcal{A}$ with storing (m_i, σ_i') in table T. The pseudorandom function is deterministic so that our simulation is perfect. Finally, $\mathcal{A}$ outputs a forgery (m^*, σ^*) on pk_s^* and pk_v^*. $\mathcal{A}_2$ submits m^* to function F with obtaining $\sigma^{*\prime}$ and outputs 1 if $\sigma^{*\prime} = \sigma^*$, indicating that function F is chosen from $\mathbf{F}$; Otherwise, he outputs 0.

Note that if F is chosen from $\mathbf{F}$, this is actually game 1; If F is a truly random function, this is game 2. Therefore, the difference between these two games is whether function F is chosen from $\mathbf{F}$. Thus we can obtain equation (3). In game 2, the signature between pk_s^* and pk_v^* is a truly random string. After querying q_{sign}, q_{sim} and q_{ver} queries, the probability that $\mathcal{A}$ outputs a valid forgery is up to

$$\begin{aligned} \Pr[X_2] &\leq (2^l - q_{sign} - q_{sim} - q_{ver})^{-1} \\ &< (q_{sign} + q_{sim} + q_{ver})2^{-l}, \end{aligned} \tag{4}$$

which is negligible. Combing equations (2) to (4), we have,

$$\begin{aligned} &\Pr_{\mathcal{A},D}^{SDVS}[\mathit{Forge}(\mathit{multi\text{-}user})](\mathit{resp.\ multi\text{-}user}^+) \\ &= \Pr[X_0] \leq \sum_{i=1}^{2} |\Pr[X_i] - \Pr[X_{i-1}]| + \Pr[X_2] \\ &< mn \cdot \epsilon_{cpa}(\mathit{resp.}\ \epsilon_{cca}) + \epsilon_{prf} + (q_{sign} + q_{sim} + q_{ver})2^{-l}. \end{aligned} \tag{5}$$

We can see from equation (5) that the probability of breaking the unforgeability in multi-user setting (resp. multi-user$^+$ setting) is negligible, which completes our proof.

■

Proof: (of Lemma 2) To simulate the signer's signature on message m, the designated verifier does the following, namely, $K.Encap^2(pk_v, pk_s, sk_v) \rightarrow K_{sv}$, $PRF_{K_{sv}}(m) \rightarrow \sigma$. The verifier can simulate the signature by running the second phase in *Encap* algorithm with inputting pk_v, pk_s and sk_v. The key K_{sv} that he can obtain is the same as the key that the signer uses to generate signatures. Since both the signer and the verifier can compute the same key, they can generate the same signature on message m, i.e. $tag = PRF_{K_{sv}}(m)$. Therefore, our constructed D scheme is perfectly non-transferability.

■

Proof: (of Lemma 3) For any PPT distinguisher $\mathcal{A}$ in SDVS's PSI game, $\Pr[\mathit{PSI}]$ in the multi-user (resp. multi-user$^+$) setting is negligibly close to $1/2$ assuming *KEM* scheme is IND-CPA (resp. IND-CCA) secure and *PRF* achieves property of pseudorandomness.

Let $\mathcal{A}$ be the distinguisher and $\mathcal{C}$ be the challenger against privacy of signer's identity game. Let K_0 denote the shared key between signer $pk_{s_0}^*$ and verifier pk_v^*, K_1 denote shared key between $pk_{s_0}^*$ and verifier pk_v^*. We consider the following games played between $\mathcal{A}$ and $\mathcal{C}$. Let X_i denote the event that $\mathcal{A}$ outputs the correct guess bit in game G_i.

$\mathbf{G_0}$: This game is the PSI game with multi-user (resp. multi-user$^+$) setting. We can have,

$$\Pr_{\mathcal{A},D}^{SDVS}[\mathit{PSI}(\mathit{multi\text{-}user})](\mathit{resp.\ multi\text{-}user}^+) = \Pr[X_0]. \tag{6}$$

$\mathbf{G_1}$: In this game, the key shared between $pk_{s_0}^*$ and pk_v^* used in *PRF* is randomly chosen, i.e. $K_0' \stackrel{\$}{\leftarrow} N$. When $\mathcal{A}$ issues signing and verification queries on these identities, $\mathcal{C}$ uses K_0' to response. The only difference between this game and game 0 is the key used in *PRF* when responding oracles between S_0 and V. Thus we can have,

$$|\Pr[X_1] - \Pr[X_0]| \leq mn \cdot \epsilon_{cpa}(\mathit{resp.}\ \epsilon_{cca}). \tag{7}$$

To obtain the above equation, we can construct an adversary $\mathcal{A}_1$ to break the IND-CPA (resp. IND-CCA)'s game and the analysis is identical to the game 1 of unforgeability and we omit the details here.

$\mathbf{G_2}$: In this game, we replace key K_1 between $pk_{s_1}^*$ and pk_v^* used in *PRF* to a random string, i.e. $K_1' \xleftarrow{\$} N$. Similar to game 1, we can have that,

$$\begin{aligned}|\Pr[X_2] - \Pr[X_1]| &\leq (m-1)n \cdot \epsilon_{cpa}(\textit{resp. } \epsilon_{cca}) \\ &< mn \cdot \epsilon_{cpa}(\textit{resp. } \epsilon_{cca}).\end{aligned} \tag{8}$$

This is the same as the transition from $\mathbf{G}_0$ to $\mathbf{G}_1$.

$\mathbf{G_3}$: In game 3, we replace PRF_{K_0} function used between $pk_{s_0}^*$ and pk_v^* to a truly random function . For every signing query on message m_i with $pk_{s_0}^*$, the signer's signature is chosen at random from $(0,1)^l$ instead of computing $PRF_{K_0}(m_i)$. We can have the following equation,

$$|\Pr[X_3] - \Pr[X_2]| \leq \epsilon_{prf}. \tag{9}$$

To prove equation (9), we can construct an adversary $\mathcal{A}_2$ to break the pseudorandomness of *PRF* with (t_2, ϵ_{prf}), where $t_2 \approx t$. The analysis is identical to the game 2 in lemma 1.

$\mathbf{G_4}$: In this game, we replace function PRF_{K_1} with a truly random function used between signer $pk_{s_1}^*$ and verifier pk_v^*. Similarly, we can have that,

$$|\Pr[X_4] - \Pr[X_3]| \leq \epsilon_{prf}. \tag{10}$$

To obtain the above equation, we can use the same proof strategy as in the transition between $\mathbf{G_2}$ and $\mathbf{G_3}$. Note that signature σ^* that distinguisher $\mathcal{A}$ receives in this game is generated by truly random functions, therefore, he can only randomly guess bit b with $\frac{1}{2}$ probability. Hence, we have the following equation,

$$\Pr[X_4] = \frac{1}{2}. \tag{11}$$

Combining equations from (6) to (11), we obtain that,

$$\begin{aligned}&\Pr_{\mathcal{A},D}^{SDVS}[\textit{PSI}(\textit{multi-user})](\textit{resp.multi-user}^+) \\ &= \Pr[X_0] \leq \sum_{i=1}^{4} |\Pr[X_i] - \Pr[X_{i-1}]| + \Pr[X_4] \\ &< 2mn \cdot \epsilon_{cpa}(\textit{resp. } \epsilon_{cca}) + 2\epsilon_{prf} + \frac{1}{2}.\end{aligned} \tag{12}$$

Because ϵ_{cpa} (resp. ϵ_{cca}) and ϵ_{prf} are all negligible. It's easy to see that the probability of breaking PSI game in multi-user setting (resp. *multi-user*$^+$ setting) is negligibly close to $\frac{1}{2}$, which completes our proof. ■

VI. Instantiations and Comparison

In this section, we give two instantiations called SDVS_1 and SDVS_2, which base on DDH assumption. Besides, we give another two post-quantum safe instantiations, namely, SDVS_3 and SDVS_4, based on LWE assumption.

We employ the well-known Diffie-Hellman key exchange scheme and *PRF* function [19] to instantiate our first SDVS scheme (SDVS_1). Note that this key exchange scheme satisfies our 2-phase *KEM* requirement. Following our construction, the resulting SDVS_1 is the same as the first scheme proposed in [10]. Based on previous analysis, the scheme in [10] is actually secure in multi-user setting. However, since Diffie-Hellman key exchange is not known to be CCA-secure, this scheme is not secure in multi-user$^+$ setting. As for the instantiation in multi-user$^+$ setting, we use a CCA-secure *KEM* scheme proposed in [2] and a *PRF* function [19] to construct SDVS_2, based on DDH assumption,.

As for the lattice-based versions, we construct SDVS_3 scheme, based on a *KEM* [4] and a *PRF* function [3]. The constructed SDVS_4 scheme derives from [26] and [3], based on LWE assumption. We omit details of these constructions here due to page limitation.

In table II, we compare our four instantiations in multi-user (resp. multi-user$^+$) setting with the existing SDVS schemes. We consider pairing, hash, *PRF* and exponentiation operations, denoted by $\mathbf{P}$, $\mathbf{H}$, $\mathbf{R}$ and $\mathbf{E}$ respectively. Please be noted that the figures of our constructed SDVS_2 come from [2] whose conclusion relies on the multi-exponential with a sliding window algorithm described in [18].

We can see from table II that our schemes, SDVS_1 and SDVS_2, are secure in multi-user and multi-user$^+$ setting respectively in the standard model. As for SDVS_3 and SDVS_4, they are quite efficient compared with the lattice-based SDVS scheme under the same security requirement.

VII. Conclusion

In this paper, we consider security of SDVS in the multi-user setting. Specifically, we strengthened the original SDVS models into multi-user (resp. multi-user$^+$) models. We also proposed a generic approach to construct SDVS in these strengthened models from *KEM* and *PRF*, with proving the security of our generic construction in the strengthened models. For comparison, we give two instantiations in each strengthened model respectively and compare their efficiency with the existing SDVS schemes.

The value we need to highlight here is that our method is a generic way of constructing SDVS. Based on the methods proposed in this paper, we can construct different specific schemes satisfying diverse security requirements based on various *KEM* schemes. Besides, any progress made on *KEM* can be applied to construct improved SDVS schemes.

References

[1] Maryam Rajabzadeh Asaar, Ali Vardasbi, and Mahmoud Salmasizadeh. Non-delegatable strong designated verifier signature using a trusted third party without pairings. In *Proceedings of Information Security 2013 (AISC 2013)*, pages 13–25. Australian Computer Society, 2013.

[2] Joonsang Baek, Willy Susilo, Joseph K. Liu, and Jianying Zhou. A new variant of the cramer-shoup kem secure against chosen ciphertext attack. In *Applied Cryptography and Network Security*, pages 143–155. Springer, 2009.

[3] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *EUROCRYPT 2012*, pages 719–737. Springer, 2012.

Table II
COMPARISON BETWEEN OUR INSTANTIATIONS AND EXISTING SDVS SCHEMES

	SDVS_1	SDVS_2	[13]	[25]	[16]		SDVS_3	SDVS_4	[20]
Signing Cost	$1\mathbf{E}+1\mathbf{R}$	$2.78\mathbf{E}+1\mathbf{R}+1\mathbf{H}$	$1\mathbf{E}+1\mathbf{H}$	$2\mathbf{E}+1\mathbf{H}$	$2\mathbf{H}+1\mathbf{P}$	PK Size (MB)	2.99×10^{-2}	21.82	1.34
Verification Cost	$1\mathbf{E}+1\mathbf{R}$	$1\mathbf{E}+1\mathbf{R}$	$1\mathbf{E}+1\mathbf{H}$	$2\mathbf{E}+1\mathbf{H}$	$2\mathbf{H}+1\mathbf{P}$	SK Size (MB)	1.49×10^{-2}	8.44	49.59
Hardness Assumption	DDH	DDH	GDH	CDH+DDH	GBDH	Hardness Assumption	LWE	LWE	LWE
Standard Model	$\surd$	$\surd$	$\times$	$\times$	$\times$	Standard Model	$\surd$	$\surd$	$\surd$
Multi-user^{+}	$\times$	$\surd$	$\times$	$\times$	$\times$	Multi-user^{+}	$\times$	$\surd$	$\times$

[4] Joppe Bos, Craig Costello, Leo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from lwe. In *SIGSAC*, pages 1006–1018. ACM, 2016.

[5] Jie Cai, Han Jiang, Pingyuan Zhang, Zhihua Zheng, Guangshi Lyu, and Qiuliang Xu. An efficient strong designated verifier signature based on r-sis assumption. *IEEE Access*, 7:3938–3947, 2019.

[6] David Chaum and Hans van Antwerpen. Undeniable signatures. In Gilles Brassard, editor, *CRYPTO' 89 Proceedings*, pages 212–216. Springer New York, 1990.

[7] Yvo Desmedt, Claude Goutier, and Samy Bengio. Special uses and abuses of the fiat-shamir passport protocol (extended abstract). In *CRYPTO '87*, pages 21–39. Springer, 1987.

[8] Yvo Desmedt and Moti Yung. Weaknesses of undeniable signature schemes. In *EUROCRYPT '91*, pages 205–220. Springer, 1991.

[9] Shuquan Hou, Xinyi Huang, Joseph K. Liu, Jin Li, and Li Xu. Universal designated verifier transitive signatures for graph-based big data. *Information Sciences*, 318(10):144–156, 2015.

[10] Qiong Huang, Willy Susilo, and Duncan S. Wong. Non-delegatable identity-based designated verifier signature. Cryptology ePrint Archive, Report 2009/367, 2009.

[11] Qiong Huang, Guomin Yang, Duncan S. Wong, and Willy Susilo. Efficient strong designated verifier signature schemes without random oracle or with non-delegatability. *International Journal of Information Security*, 10(6):373, Aug 2011.

[12] Xinyi Huang, Willy Susilo, Yi Mu, and Futai Zhang. Short (identity-based) strong designated verifier signature schemes. In *ISPEC 2006*, pages 214–225. Springer, 2006.

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

[14] Markus Jakobsson, Kazue Sako, and Russell Impagiazzo. Designated verifier proofs and their applications. In *EUROCRYPT 1996*, pages 143–154. Springer, 1996.

[15] Fabien Laguillaumie and Damien Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In *Security in Communication Networks*, pages 105–119. Springer, 2004.

[16] Fabien Laguillaumie and Damien Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In *Security in Communication Networks*, pages 105–119. Springer, 2005.

[17] Han-Yu Lin, Tzong-Sun Wu, and Yi-Shiung Yeh. A dl based short strong designated verifier signature scheme with low computation. *Journal of Information Science and Engineering*, 27(2):451–463, 2012.

[18] Avanzi Roberto M. The complexity of certain multi-exponentiationtechniques in cryptography. *Journal of Cryptology*, 18(4):357–373, Sep 2005.

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

[20] Geotae Noh and Ik Rae Jeong. Strong designated verifier signature scheme from lattices in the standard model. *Security and Communication Networks*, 9(18):6202–6214, 2016.

[21] Shahrokh Saeednia, Steve Kremer, and Olivier Markowitch. An efficient strong designated verifier signature scheme. In *ICISC 2003*, pages 40–54. Springer, 2003.

[22] Willy Susilo, Fangguo Zhang, and Yi Mu. Identy-based strong designated verifier signature schemes. In *ACISP 2004*, pages 313–324. Springer, 2004.

[23] Haibo Tian, Xiaofeng Chen, Fangguo Zhang, Baodian Wei, Zhengtao Jiang, and Yi Liu. An efficient identity-based strong designated verifier signature without delegatability. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 81–88, 2012.

[24] Haibo Tian and Jin Li. A short non-delegatable strong designated verifier signature. *Frontiers of Computer Science*, 8(3):490–502, 2014.

[25] Raylin Tso, Takeshi Okamoto, and Eiji Okamoto. Practical strong designated verifier signature schemes based on double discrete logarithms. In *Information Security and Cryptology*, pages 113–127. Springer, 2005.

[26] Jiang Zhang, Yu Yu, Shuqin Fan, and Zhenfeng Zhang. Improved lattice-based cca2-secure pke in the standard model. Cryptology ePrint Archive, Report 2019/149, 2019. https://eprint.iacr.org/2019/149.