**Control and Cybernetics**

vol. **42** (2012) No. 2

# A new strong multiple designated verifiers signature for broadcast propagation[*][†]

by

**Haibo Tian**[1] **and Yi Liu**[2]

[1]School of Information Science and Technology,
Sun Yat-Sen University, Guangzhou, 510275, China
e-mail: sysutianhb@gmail.com
[2]Faculty of Computer, Guangdong University of Technology,
Guangzhou, Guangdong, 510006, China
e-mail: yiliu@gdut.edu.cn

**Abstract:** A strong multiple designated verifiers signature (SMDVS) enables a signer to convince a set of verifiers by generating one signature, of which the verification needs a private key of a verifier. After a brief survey of current SMDVS schemes, we find no schemes suitable to a broadcast propagation, where the simulation needs only one verifier's private key. Motivated by this discovery, we propose a broadcast SMDVS scheme. The new scheme is proven secure in the random oracle model.

**Keywords:** designated verifier signature, asymmetric group key agreement protocol, signature scheme of Schnorr, broadcast propagation.

## 1. Introduction

### 1.1. The background

Jakobsson et al. (1996) proposed the designated verifier signature (DVS). It is a privacy protection method. A DVS consists of a proof that either "the signer has signed on a message" or "the signer has the verifier's secret key". If a designated verifier is confident that her/his private key is kept in secret, the verifier makes sure that a signer has signed on a message. No other parties can be convinced by a DVS since the designated verifier can generate it with her/his private key. It is useful in various commercial cryptographic applications, such as e-voting, copyright protection, etc.

There are some extensions to the DVS concept, including at least strong DVS (SDVS), multiple DVS (MDVS), universal DVS (UDVS) and their combinations.

[*]Submitted: November 2011; Accepted: April 2012

[†]This paper is a full version of the paper with the same title in the proceedings of INCoS 2011.

- In the appendix, Jakobsson et al. (1996) gave a definition of SDVS. It means that a verifier needs to use her/his private key to verify a signature. This concerns the situation where the signature is captured before reaching the verifier. In this case, an adversary can know who is the real signer as there are only two possibilities. Laguillaumie and Vergnaud (2004), and Saeednia (2004) both formalized the notion.
- In the section six, Jakobsson et al. (1996) extended a DVS to an MDVS. It is a convenience for a signer to convince a set of verifiers, and only this set, by generating one signature. It is helpful in multi-party activities like distributed contract signing. Desmedt (2003) proposed the MDVS notion at the rump session of the Crypto conference in 2003. Laguillaumie and Vergnaud (2004) formalized the notion. The concept is suitable for collaborative systems to provide privacy-enabled services.
- Steinfield et al. (2004) proposed the primitive UDVS. There are three players in the setting, namely a trusted certification authority (CA), a user and a verifier. The CA signs a certificate and gives it to the user. The user uses the verifier's public key to transform the CA's signature into a DVS. In user certification systems, the UDVS makes that the verifier cannot convince a third-party verifier about the truth of the statements in a certificate.
- One can combine the above extensions. For example, SMDVS combines the SDVS and MDVS, and UMDVS combines the UDVS and MDVS, and USMDVS combines the UDVS, SDVS and MDVS.

In the field of MDVS, there is a practical issue, namely the propagation of a signature among all verifiers. There are two modes defined by Li et al. (2007): a sequential mode and a broadcast mode.

- In the sequential mode, a signer delivers a signature from verifier $V_1$ to $V_n$ in a step-by-step manner. A simulation algorithm should be executed by a coalition of all verifiers. Otherwise, a successor $V_j$ cannot distinguish whether a signature was generated by the signer or simulated by its predecessor $V_i$ $(i \leq j)$. The bad aspect is that if a verifier $V_i$ is corrupted or captured/controlled by a hostile third party and hence becomes unavailable, the simulation cannot be accomplished. Then the source of a signature can be revealed.
- In the broadcast mode, a signer broadcasts a signature to all verifiers simultaneously. A simulation algorithm can be executed by each verifier independently.

In this paper, we focus on the broadcast SMDVS (B-SMDVS). The features of a B-SMDVS scheme include one-verifier simulation, private verification, and one signature for multiple verifiers. We briefly survey current SMDVS schemes, and then give a new scheme.

## 1.2. Related works

Laguillaumie and Vergnaud (2004) proposed an MDVS scheme based on ring signatures. They claimed that one could transform it to an SMDVS by adding an encryption layer. It needs an $n$-party key agreement protocol to enable the encryption layer. Using the similar method, one can transform MDVS schemes, such as those in Jakobsson et al. (1996), Li et al. (2007), and Chow (2008), into SMDVS schemes. Chow (2006) proposed an identity based SMDVS scheme. Laguillaumie and Vergnaud (2007) proposed an SMDVS without using the encryption layer.

A UDVS simply implies a DVS if we take the CA and the user as one entity. So we survey some works about UMDVS and USMDVS. Ng et al. (2005) proposed two USMDVS schemes based on the UDVS scheme proposed by Steinfield et al. (2004). Ming and Wang (2008) and Shailaja et al. (2006) proposed an USMDVS scheme based on the Gap Bilinear Diffie-Hellman assumption and $q$-Strong Diffie-Hellman assumption, respectively. They proved their schemes in the standard model. Vergnaud (2008) proposed an UMDVS and an USMDVS scheme extended from paring-based signatures. Seo et al. (2008) proposed an identity based USMDVS scheme with constant signature size. Chang (2011) proposed an identity based USMDVS scheme in the multi-signers setting.

None of the above schemes are designed for the broadcast mode. In fact, it is an open problem in Li et al. (2007) to design a scheme suitable to the broadcast mode. The main difficulty may be the simulation of an SMDVS by a verifier independently. From our comparison, only our scheme needs one verifier to simulate a signature. It should be added, though, that we have changed the concept of B-SMDVS by adding an agent entity in its definition.

## 1.3. Contributions

First of all, we give a possible application scenario, the load balancing in the network field. Assume a single Internet service provided by multiple servers. There is a load balancer who receives client's requests and forwards them to one of the back-end servers. An example of this kind of load balancer is shown in Fig. 1. A client can send a B-SMDVS to the load balancer for authentication about its access rights to the back-end servers and for the protection of its privacy. The load balancer then forwards the B-SMDVS to a back-end server which is one of the designated verifiers. The server can verify the B-SMDVS by using its private key and give access rights to the client. At the same time, the server can simulate a B-SMDVS by using its private key so that there is no evidence of the client's visiting.

Secondly, we describe a general B-SMDVS by modifying the traditional MDVS definitions. Especially, we model the "load balancer" as an individual entity in the B-SMDVS scheme. We call it an *agent* since a client directly connects to it and the servers receive client's requests from it. The agent is important in

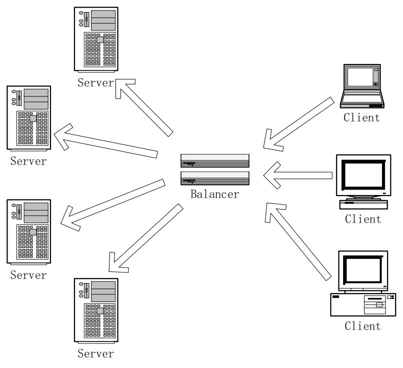

Figure 1. A kind of network load balancer

our design as it receives registrations from servers and publishes public keys to clients. It is assumed to be in the same (virtual) local area network (LAN) as all servers. The agent can also be an application gateway which provides general information to clients.

It seems that there is a trivial solution to design a B-SMDVS. One can simply run a group key agreement protocol among all the verifiers and the agent to share a secret, say $\gamma$. Then the agent uses the value $\gamma$ as a private key, and publishes a public key, say $g'^{\gamma}$ for a group generator $g'$. Then an SDVS scheme can be used to produce a signature where the designated public key is $g'^{\gamma}$. Any verifier with the secret $\gamma$ can verify the signature and simulate a signature.

However, there is a drawback about the solution. Since the public key $g'^{\gamma}$ should not be changed frequently, the protection about the shared secret is important. In the above solution, the shared secret is stored in many servers, which increases the security risk. And if a server loses its shared secret, a new running of the key agreement protocol is needed among all servers. That is, an error in one server will affect other servers.

We provide the first non-trivial version of B-SMDVS. We introduce the asymmetric group key agreement (ASGKA) protocol in Wu et al. (2009). It enables a group of verifiers to share a public key from scratch. If one verifier loses its private key, the verifier recomputes its key pairs and republishes the public key, and then the agent updates its public key. Considering a comment on (S)MDVS schemes by Ushida et al. (2010), we use the well-known scheme in Schnorr (1990) as a basic building block. And we use a new approach, adding a temporal value to the hashing output, to make a signature be simulated by a designated verifier.

One limitation of our scheme is that the set of verifiers should not be changed frequently. Verifiers need to reset their public keys if the group of verifiers changes. The scheme is suitable to multiple servers to provide a single privacy-enabled service.

### 1.4. Organization

Section 2 contains the preliminaries on some hard problems. Section 3 defines a general B-SMDVS and its security model. Section 4 provides the B-SMDVS scheme. The proofs of properties are in Section 5. Section 6 compares our scheme with other related schemes. The last section contains a conclusion.

## 2. Preliminaries

### 2.1. Hard problems

Let **PairGen** be an algorithm that, on inputting a security parameter $k$, outputs a tuple $(p, \mathbb{G}, \mathbb{G}_{\mathbb{T}}, e)$, where $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ have the same prime order $p$, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbb{T}}$ is an efficient non-degenerate bilinear map such that $e(g, g) \neq 1$ for any generator $g$ in $\mathbb{G}$, and $\forall u, v \in \mathbb{Z}$, it holds that $e(g^u, g^v) = e(g, g)^{uv}$.

- **Computational Diffie-Hellman (CDH)** Given big primes $p', q'$ such that $q'|p'-1$, let $\mathbb{G}' = \langle g' \rangle$ with order $q'$. Given random elements $(g'^a, g'^b) \in_R \mathbb{G}'^2$, a CDH problem is to find $g'^c = g'^{ab} \bmod p'$.
- **Decisional Diffie-Hellman (DDH)** Given big primes $p', q'$ such that $q'|p'-1$, let $\mathbb{G}' = \langle g' \rangle$ with order $q'$. Given random elements $(g'^a, g'^b, g'^c) \in_R \mathbb{G}'^3$, a DDH problem is to decide whether $c = ab \bmod q'$.
- **$n$-Bilinear Diffie-Hellman Exponent ($n$-BDHE)** Let $(p, \mathbb{G}, \mathbb{G}_{\mathbb{T}}, e) \leftarrow$ **PairGen**$(k)$. Let $g$ and $h$ be two independent generators of $\mathbb{G}$. Denote $\vec{y}_{g,\alpha,n} = (g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}) \in \mathbb{G}^{2n-1}$, where $g_i = g^{\alpha^i}$ for some unknown $\alpha \in \mathbb{Z}_p^*$. Given a tuple $(g, h, \vec{y}_{g,\alpha,n}, Z \in_R \mathbb{G}_{\mathbb{T}})$, an $n$-BDHE problem is to decide whether $Z = e(g_{n+1}, h)$.

The assumption is that there are no polynomial algorithms to solve the CDH (DDH, $n$-BDHE) problem with a non-negligible advantage $\epsilon$ in time $t$.

## 3. B-SMDVS

### 3.1. The scheme

A B-SMDVS scheme consists of six algorithms (**Setup**, **SKeyGen**, **VKeyGen**, **AKeyGen**, **Sign** and **Verify**).

- **Setup** is a probabilistic algorithm which takes a security parameter $k$ as input and outputs a public parameter $sp$.
- **SKeyGen** is a probabilistic algorithm which takes $sp$ as input and outputs a pair of secret and public keys $(sk_S, pk_S)$ for a signer $S$.
- **VKeyGen** is a probabilistic algorithm which takes the $sp$ as input and outputs a pair of secret and public keys $(sk_i, pk_i)$ for a verifier $V_i$.
- **AKeyGen** is a probabilistic algorithm which takes the $sp$ and public keys $pk_1, \ldots, pk_n$ as input and outputs an agent public key $pk_A$.

- **Sign** is a probabilistic algorithm which takes a message $m$, the signer's secret key $sk_S$, the agent's public key $pk_A$ and the public parameter $sp$ as input, and outputs a $(V_1, \dots, V_n)$-designated verifiers signature $\delta$ on $m$.
- **Verify** is a deterministic algorithm which takes a message $m$, a $(V_1, \dots, V_n)$ - designated verifiers signature $\delta$ on $m$, the signer's public key $pk_S$, the public keys of all verifiers $(pk_1, \dots, pk_n)$, the private key $sk_i$ for the verifier $V_i$, $i \in \{1, \dots, n\}$, and the public parameter $sp$ as input, and outputs 'True' or 'False' by each verifier independently.

A B-SMDVS scheme enjoys four properties, including *correctness*, *unforgeability*, *source hiding*, and *privacy of signer's identity* (PSI).

- **Correctness:** A properly formed $(V_1, \dots, V_n)$-designated verifiers signature must be accepted by the **Verify** algorithm. Moreover, a putative signature is accepted by one verifier if and only if the signature is accepted by all verifiers;
- **Unforgeability:** It is computationally infeasible, without the knowledge of the secret key of either the signer $S$ or *one* of the $(V_1, \dots, V_n)$ verifiers to produce a $(V_1, \dots, V_n)$-designated verifiers signature that is accepted by the **Verify** algorithm of all verifiers.
- **Source hiding:** Given a message $m$ and a $(V_1, \dots, V_n)$-designated verifiers signature $\delta$ of this message, it is infeasible to determine who of the original signer or *one* of the designated verifiers performed this signature, even if all secrets are known.
- **PSI**: Given a message $m$ and a $(V_1, \dots, V_n)$-designated verifiers signature $\delta$ of $m$, it is computationally infeasible, without the knowledge of the secret key of any $V_i$ for $i \in \{1, \dots, n\}$, to determine which signing key out of two possibilities was used to generate $\delta$.

REMARK 1 *The source hiding property is to hide the source of a signature from $n+1$ possible sources. One source is a real signer. The other $n$ sources are the $n$-designated verifiers. It is not meant to hide the possible identities of signers and verifiers to achieve total anonymity. A signature does not reveal its true source, even if a set of public keys are bound to the signature.*

## 3.2. The security model

### Unforgeability

We adapt the model defined in Laguillaumie and Vergnaud (2004). However, *there is no corruption oracle.* Instead, a verification oracle is provided. The unforgeability is defined as follows.

DEFINITION 1 *Let $(V_1, \dots, V_n)$ be $n$ entities, $k$ and $t$ be integers and $\epsilon$ be a real in $[0, 1]$. Let $k$ be the security parameter of a B-SMDVS scheme. Let $\mathcal{A}$ be an adversary who tries to existentially forge a signature under chosen message attack (EF-CMA). The random experiment, $\boldsymbol{Exp}^{ef-cma}_{B-SMDVS,\mathcal{A}}(k)$, is defined as:*

1. *$sp \leftarrow \boldsymbol{Setup}(k)$;*

2. *For $i = 1, \ldots, n$, do $(pk_i, sk_i) \leftarrow$ **VKeyGen**$(sp)$ for each designated verifier $V_i$;*
3. *$pk_A \leftarrow$ **AKeyGen**$(sp, pk_1, \ldots, pk_n)$;*
4. *$(pk_S, sk_S) \leftarrow$ **SKeyGen**$(sp)$ for a signer $S$;*
5. *$(m^*, \delta^*) \leftarrow \mathcal{A}^{\mathcal{H},\Sigma,\Upsilon}(sp, pk_1, \ldots, pk_n, pk_A, pk_S)$, where $\mathcal{H}$ is the hashing oracle, $\Sigma$ is the signing oracle, and $\Upsilon$ is the verifying oracle;*
6. *Return*

$$\bigvee_{i=1}^{n} \textbf{\textit{Verify}}(sp, m^*, \delta^*, pk_S, pk_1, \ldots, pk_n, sk_i) \ . \qquad (1)$$

*The success probability of the adversary $\mathcal{A}$ is defined as:*

$$\textbf{\textit{Suc}}^{ef-cma}_{B-SMDVS,\mathcal{A}}(k) = Pr[\textbf{\textit{Exp}}^{ef-cma}_{B-SMDVS,\mathcal{A}}(k) = \textbf{\textit{True}}] \ . \qquad (2)$$

*A B-SMDVS scheme is said to be $(k, t, \epsilon)$-EF-CMA secure, if no adversary $\mathcal{A}$ running in time $t$ has a success probability $\textbf{\textit{Suc}}^{ef-cma}_{B-SMDVS,\mathcal{A}}(k) \geq \epsilon$.*

## Source hiding

The source hiding property is referred to Rivest et al. (2001) by Laguillaumie and Vergnaud (2004). We modify it to the B-SMDVS setting.

DEFINITION 2 *Even an infinitely powerful adversary with access to an unbounded number of chosen-message signatures produced by the same signer or one of the n-designated verifiers cannot guess the source of a signature with any advantage, and cannot link additional signatures to the same source.*

## PSI

DEFINITION 3 *Let $(V_1, \ldots, V_n)$ be $n$ entities, $k$ and $t$ be integers and $\epsilon$ be a real in $[0, 1]$. Let $k$ be the security parameter of a B-SMDVS scheme. Let $\mathcal{A}$ be an* PSI-CMA*-adversary against a B-SMDVS scheme. The random experiment, $\textbf{\textit{Exp}}^{psi-cma}_{B-SMDVS,\mathcal{A}}(k)$, is defined as:*

1. *$sp \leftarrow$ **Setup**$(k)$;*
2. *For $i = 1, \ldots, n$, do $(pk_i, sk_i) \leftarrow$ **VKeyGen**$(sp)$ for each designated verifier $V_i$;*
3. *$pk_A \leftarrow$ **AKeyGen**$(sp, pk_1, \ldots, pk_n)$;*
4. *$(pk_{S0}, sk_{S0}) \leftarrow$ **SKeyGen**$(sp)$ for a signer $S0$;*
5. *$(pk_{S1}, sk_{S1}) \leftarrow$ **SKeyGen**$(sp)$ for a signer $S1$;*
6. *$(m^*, I^*) \leftarrow \mathcal{A}^{\mathcal{H},\Sigma_0,\Sigma_1,\Upsilon}($**find**$, sp, pk_A, pk_{S0}, pk_{S1}, pk_1, \ldots, pk_n)$, where $\mathcal{H}$ is the hashing oracle, $\Sigma_0$ and $\Sigma_1$ are signing oracles, and $\Upsilon$ is a verifying oracle;*
7. *Flip a fair coin $b \in_R \{0, 1\}$ and generate $\delta^* \leftarrow$ **Sign**$(sp, m^*, sk_{Sb}, pk_A)$;*
8. *Compute $b' \leftarrow \mathcal{A}^{\mathcal{H},\Sigma_0,\Sigma_1,\Upsilon}($**guess**$, sp, m^*, I^*, \delta^*, pk_1, \ldots, pk_n, pk_A, pk_{S0}, pk_{S1})$.*

*9. Return $b'$.*
*The advantage of $\mathcal{A}$ is defined as:*

$$\boldsymbol{Adv}^{psi-cma}_{B-SMDVS,\mathcal{A}}(k) = |Pr(b' = b) - 1/2| \ . \qquad (3)$$

*A B-SMDVS is said to be $(k, t, \epsilon)$-PSI-CMA secure, if no adversary $\mathcal{A}$ running in time $t$ has an advantage $\boldsymbol{Adv}^{psi-cma}_{B-SMDVS,\mathcal{A}}(k) \geq \epsilon$.*

## 4. The B-SMDVS Scheme

The scheme is derived from the ASGKA protocol in Wu et al. (2009) and the signature scheme in Schnorr (1990).

- **Setup:** Let $(p, \mathbb{G}, \mathbb{G}_{\mathbb{T}}, e) \leftarrow$**PairGen**$(k)$, $\mathbb{G} = \langle g \rangle$. Let $p', q'$ be big primes such that $p' = 2q' + 1$. Let $\mathbb{G}'$ be a group of order $q'$ and $\mathbb{G}' = \langle g' \rangle$. Let $H_1 : \{0,1\}^* \to \mathbb{G}$, $H_2 : \{0,1\}^* \to \mathbb{Z}^*_{q'}$, and $H_3 : \mathbb{G}_{\mathbb{T}} \to \mathbb{Z}^*_{q'}$ be secure hash functions. The system parameter $sp$ is $(p, \mathbb{G}, \mathbb{G}_{\mathbb{T}}, e, p', q', \mathbb{G}', H_1, H_2, H_3)$.
- **SKeyGen:** A signer $S$ randomly selects $x_S \in_R \mathbb{Z}^*_{q'}$ and computes $y_S = g'^{x_S}$. The public key is $y_S$ and the private key is $x_S$.
- **VKeyGen:** For $i \in \{1, \ldots, n\}$, a verifier $V_i$ selects $r_i \in_R \mathbb{Z}^*_p$ and $X_i \in_R \mathbb{G}$. The verifier computes $A_i = e(g, X_i)$, $R_i = g^{-r_i}$, $\delta_{ij} = X_i H_1(V_j)^{r_i}$ for $j \in \{1, \ldots, n\}$. The verifier publishes $(A_i, R_i, \delta_{i1}, \ldots, \delta_{i(i-1)}, \delta_{i(i+1)}, \ldots, \delta_{in})$ as the public key $pk_i$. The secret key $sk_i$ is $\delta_{ii}$.
- **AKeyGen:** An agent $P$ randomly selects $t \in_R \mathbb{Z}^*_p$, computes $c_1 = g^t, c_2 = (\prod_{i=1}^n (R_i))^t, \gamma = H_3(\prod_{i=1}^n (A_i)^t), y_A = g'^{\gamma}$.The public key $pk_A$ is $(c_1, c_2, y_A)$.
- **Sign:** On inputting a message $m$, the system parameters $sp$, the signer's private key $x_S$, the public key of the agent $pk_A$, the signer computes a signature $\delta$ as follows.
  1. Randomly select $\alpha, \beta \in_R \mathbb{Z}^*_{q'}$ and compute $\Theta = g'^{\alpha}$ and $\Lambda = g'^{\beta}$.
  2. Compute $s_e = H_2(m||\Theta) + \Lambda \bmod q'$ and $s_s = \alpha - x_S s_e \bmod q'$.
  3. Compute $c_3 = y_A^{\beta}$ and set $\delta = (c_1, c_2, c_3, s_e, s_s)$
- **Verify:** On inputting a message $m$, the signature $\delta$ on $m$, the public key of the signer $y_S$, the private key $sk_i$ of a verifier $V_i$, and the public keys of all verifiers $pk_j$, $j \in \{1, \ldots, n\}$, the verifier $V_i$ verifies the signature $\delta$ as follows:
  1. Compute $k_d = \prod_{j=1}^n \delta_{ji}$, and $\gamma' = H_3(e(k_d, c_1)e(H_1(V_i), c_2))$.
  2. Compute $\Theta = g'^{s_s} y_S^{s_e}$, $\Lambda = c_3{}^{\gamma'^{-1}}$ and $s'_e = H_2(m||\Theta) + \Lambda \bmod q'$.
  3. If $s'_e \neq s_e$, produce 'False'. Otherwise, produce 'True'.

Remark 2 *The simulation algorithm is provided in the proof of the source hiding property.*

## 5. Security proof

### Source hiding property

Proposition 1 *There is a simulation algorithm for each of the $n$-designated verifiers so that nobody can distinguish the source of a signature between a real signer and one of the $n$-designated verifiers.*

*Proof.* The simulation algorithm takes as input the system parameter $sp$, a message $m$, the public keys of verifiers $pk_1, \dots, pk_n$, the public key of the agent $pk_A$, the public key of the signer $pk_S$, and the private key of one verifier $V_i$, $i \in \{1, \dots, n\}$. It produces an output $\delta_v$. It is specified as follows.

1. Randomly select $s_s, s_e \in_R \mathbb{Z}^*_{q'}$.
2. Compute $\Theta = g'^{s_s} y_S^{s_e}$.
3. Compute $\Lambda_p = s_e - H_2(m||\Theta)$.
4. If $\Lambda_p, \Lambda_p + q', \Lambda_p + 2q' \notin \mathbb{G}'$, return to step 1 and reselect $s_e$. Else set $\Lambda = \Lambda_p + \tau q' \in \mathbb{G}'$, $\tau \in \{0, 1, 2\}$, and continue.
5. Compute $k_d = \prod_{j=1}^{n} \delta_{ji}$, and $\gamma = H_3(e(k_d, c_1)e(H_1(V_i), c_2))$.
6. Compute $c_3 = \Lambda^\gamma$.
7. Set $\delta_v = (c_1, c_2, c_3, s_e, s_s)$.

The tuple $(c_1, c_2, c_3)$ in $\delta_v$ appears with the same probability as in $\delta$ produced by a real signer for a fixed value $\Lambda$. This is simply because $e(k_d, c_1)e(H_1(V_i), c_2) = \prod_{i=1}^{n}(A_i)^t$, which is the basic equation from Wu et al. (2009).

We next consider the probability of $(s_e, s_s, \Lambda)$ appearing in a simulation and in a real running. Suppose a possible tuple $(\hat{s_e}, \hat{s_s}, \hat{\Lambda})$ from the samples of all valid signatures. Consider the probability of the tuple being produced by a signer. Since $\hat{s_e} = H_2(m||g'^{\alpha}) + g'^{\beta} \bmod q'$, $\hat{s_s} = \alpha - x_S s_e \bmod q'$ and $\hat{\Lambda} = g'^{\beta}$, the randomness is over the variables $\alpha, \beta \in_R \mathbb{Z}_{q'}$ for the fixed private key $x_S$, message $m$ and system parameters $sp$. The probability is about $1/q' \cdot 1/(\omega_0 q') = 1/\omega_0 q'^2$, where $\omega_0 = \#\Omega_0/q'$, and $\Omega_0 = \{s_e | s_e = H_2(m||g'^{\alpha}) + g'^{\beta} \bmod q' \wedge \beta \in \mathbb{Z}^*_{q'}\}$ for fixed $\alpha$, which is the set of $s_e$ when $\beta$ goes through the group $\mathbb{Z}^*_{q'}$.

Consider that the tuple is produced by one of the $n$-designated verifiers. Since $\hat{s_e} = s_e$, $\hat{s_s} = s_s$ and $\hat{\Lambda} = s_e - H_2(m||g'^{s_s} y_S^{s_e}) + \tau q'$, $\tau \in \{0, 1, 2\}$, the randomness is over the variables $s_s, s_e \in_R \mathbb{Z}^*_{q'}$ and the reselecting of $s_e$ for fixed public key $y_S$, message $m$ and system parameters $sp$. The probability is about $1/q' \cdot 1/\omega_1 q' = 1/\omega_1 q'^2$ where $\omega_1 = \#\Omega_1/q'$, and $\Omega_1 = \{s_e | s_e - H_2(m||g'^{s_s} y_S^{s_e}) + \tau q' \in \mathbb{G}' \wedge \tau \in \{0, 1, 2\}\}$, which is the set of $s_e$ such that there is a group element in $\mathbb{G}'$ for fixed $s_s$.

If $s_e \in \Omega_0$, there is a value $\beta$ such that $g'^{\beta} = s_e + \tau p' - H_2(m||g'^{\alpha})$. Since $0 < s_e < q'$ and $0 < H_2(m||g'^{\alpha}) < q'$ and $p' = 2q' + 1$, $0 < s_e + \tau p' - H_2(m||g'^{\alpha}) < p'$ for $\tau \in \{0, 1, 2\}$. This is the condition in the set $\Omega_1$ for $s_e \in \Omega_1$.

If $s_e \in \Omega_1$, there is a value $\beta \in \mathbb{Z}^*_{q'}$ such that $g'^{\beta} + H_2(m||g'^{s_s} y_S^{s_e}) + \tau q' = s_e$. This implies $s_e \in \Omega_0$.

So, the two sets, $\Omega_0$ and $\Omega_1$, are equivalent, and $\#\Omega_0 = \#\Omega_1$. Hence, a signature may be produced by a signer or by one of the $n$ designated verifiers with an equal probability. This conclusion does not depend on any complexity-theoretic assumptions or on the randomness of an oracle. ■

**Unforgeability property**

PROPOSITION 2 *Suppose the hash function $H_2$ is a random oracle, and $q_h$ is the number of hashing queries. Suppose the CDH problem is $(t', \epsilon')$ unsolvable, and the $n$-BDHE problem is $(t'', \epsilon'')$ unsolvable. Then, the B-SMDVS scheme is $(k, t, \epsilon)$-EF-CMA secure with $q_s$ signing queries and $q_v$ verifying queries.*

*Proof.* Suppose a CDH problem instance $(g', g'^{\alpha}, g'^{\beta})$ and an $n$-BDHE problem instance $(g, h, \vec{y}_{g,\alpha,n}, Z)$. Suppose a simulator $\mathcal{S}$ who tries to solve a CDH problem or an $n$-BDHE problem. Suppose an adversary $\mathcal{A}$ who claims to $(k, t, \epsilon)$ forge a new signature of the B-SMDVS scheme, where $\epsilon$ is non-negligible in time $t$ with security parameter $k$. Let $P$ denote the time for the pairing evaluation cost and $\tau$ denote the time for exponentiation.

$\mathcal{S}$ runs a series of games with $\mathcal{A}$ as follows:

- **Game 0:**
  1. $\mathcal{S}$ runs **Setup** to get the parameter $sp$ and sends it to $\mathcal{A}$.
  2. $\mathcal{S}$ sets $pk_S = g'^{\alpha}$ as the public key of a signer and sends it to $\mathcal{A}$.
  3. $\mathcal{S}$ runs **VKeyGen** to generate key pairs $(sk_i, pk_i)$ for all $i \in \{1, \ldots, n\}$. $\mathcal{S}$ sends all public keys to $\mathcal{A}$.
  4. $\mathcal{S}$ runs **AKeyGen** to produce $pk_A = (c_1, c_2, y_A)$ and sends it to $\mathcal{A}$.
  5. $\mathcal{S}$ answers the signing, verifying, and hashing queries by oracles $\Sigma$, $\Upsilon$, $\mathcal{H}$.
     (a) **Signing queries** $\mathcal{S}$ installs a private key of one verifier on $\Sigma$ oracle. $\mathcal{A}$ sends a message $m$ to $\Sigma$. $\Sigma$ uses the simulation algorithm to generate a signature $\delta$ as a reply. During the simulation, to compute the step 3, $\Sigma$ queries $\mathcal{H}$ with $(m, \Theta)$ for the computation of $H_2$. Considering Proposition 1, the reply is totally qualified.
     (b) **Verifying queries** $\mathcal{S}$ installs private keys of all verifiers on $\Upsilon$ oracle. $\Upsilon$ runs the **Verify** algorithm for each verifier. If there is a 'True' output, $\Upsilon$ returns 'True'. During the verification, to compute the step 2, $\Upsilon$ queries $\mathcal{H}$ with $(m, \Theta)$ for the computation of $H_2$.
     (c) **Hashing queries** $\mathcal{H}$ maintains a list $H_{list}$, empty at the beginning. When $\mathcal{A}$ or $\Sigma$ sends a query $(m, \Theta)$, $\mathcal{H}$ checks whether the pair is in the $H_{list}$. If it is, $\mathcal{H}$ replies to $\mathcal{A}$ or $\Sigma$ with the recorded value, $ran$, of the matching entry. If it is not, an new entry $(m, \Theta, ran \in_R \mathbb{Z}_{q'}^{*})$ is recorded in the $H_{list}$. The value $ran$ is the reply to $\mathcal{A}$ or $\Sigma$. When $\Upsilon$ sends a query $(m, \Theta)$, $\mathcal{H}$ finds the pair in the $H_{list}$ and replies to $\Upsilon$ with the recorded value, $ran$, of the matching entry.

The game is the same as the experiment $\mathbf{Exp}^{euf-cma}_{MDVS,\mathcal{A}}(k)$ for adversary $\mathcal{A}$. According to $\mathcal{A}$'s claim, a forged signature $\delta^* = (c_1, c_2, c_3^*, s_e^*, s_s^*)$ on a message $m^*$ can be produced in time $t$ with a non-negligible probability $\epsilon$. By the general forking lemma in Bellare and Neven (2006), $\mathcal{A}$ can forge another signature $\delta' = (c_1, c_2, c_3', s_e', s_s')$ on the same message $m^*$ with a non-negligible probability $\epsilon(\epsilon/q_h - 1/q')$.
Suppose a probability $\xi$ with which $(s_e^*, s_s^*) \neq (s_e', s_s')$. $\mathcal{S}$ can extract the private key $\alpha$ and compute $g'^{\beta\alpha}$ as a CDH answer. The success probability of $\mathcal{S}$ in this case is $\epsilon' = \xi\epsilon(\epsilon/q_h - 1/q)$. The runtime of $\mathcal{C}$ is $t' < t + 2(2q_s (P+\tau) + 2q_v n(P+\tau))$.
The event of $(s_e^*, s_s^*) = (s_e', s_s')$ with probability $1-\xi$ leads to the following games:

- **Game 1** This game is intended to show that the value $y_A$ can just be a random value.
  1. $\mathcal{S}$ runs **Setup** to get the parameter $sp$ and sends it to $\mathcal{A}$.
  2. $\mathcal{S}$ runs **SKeyGen** to produce a signer's key pair $(sk_S, pk_S)$ and sends $pk_S$ to $\mathcal{A}$.
  3. With the $n$-BDHE problem instance $(g, h, \vec{y}_{g,\alpha,n}, Z)$, $\mathcal{S}$ produces public keys of verifiers and the agent according to the method in Wu et al. (2009).
     – For a verifier $V_j$, $\mathcal{S}$ randomly selects $v_j \in \mathbb{Z}_p$ and computes $h_j = g_j g^{v_j}$. Then $\mathcal{S}$ randomly selects $i^* \in \{1, \ldots, n\}$,$a_i, r_i \in \mathbb{Z}_p^*$. Let $S_{i^*} = \{1, \ldots, n\} \setminus \{i^*\}$. Compute

$$\begin{aligned} R_{i^*} &= g^{r_{i^*}}(\prod_{k\in S_{i^*}} g_{n+1-k}), \delta_{i^*,j} \\ &= g^{a_{i^*}} g_j^{-r_{i^*}} (\prod_{k\in S_{i^*}}^{k\neq j} g_{n+1-k+j}^{-1}) R_{i^*}^{-v_j} (j \neq i^*). \end{aligned} \tag{4}$$

     For $i \neq i^*$, compute

$$\begin{aligned} R_i &= g^{r_i} g_{n+1-i}^{-1}, \delta_{i,j} \\ &= g^{a_i} g_j^{-r_i} g_{n+1-i+j} R_i^{-v_j} \text{ for } j \neq i. \end{aligned} \tag{5}$$

     Then $\forall i \in \{1, \ldots, n\}$, the following is defined

$$A_i = e(\delta_{i,j}, g) e(h_j, R_i) \text{ for } j \neq i. \tag{6}$$

     – For the agent, $\mathcal{S}$ computes

$$(c_1 = h, c_2 = h^{\sum_{i=1}^n r_i}, \gamma = H_3(Ze(g,h)^{\sum_{i=1}^n a_i}), y_A = g'^{\gamma}). \tag{7}$$

     $\mathcal{S}$ then sends all public keys to $\mathcal{A}$.

4. $\mathcal{S}$ answers the signing, verifying, and hashing queries by oracles $\Sigma, \Upsilon, \mathcal{H}$.
   (a) **Signing queries** $\mathcal{S}$ installs the private key $sk_S$ of the signer on $\Sigma$ oracle. $\mathcal{A}$ sends a message $m$ to $\Sigma$. $\Sigma$ uses the **Sign** algorithm to generate a signature $\delta$ as a reply. At the step 2 of the algorithm, $\Sigma$ queries $\mathcal{H}$ with $(m, \Theta)$ for the computation of $H_2$.
   (b) **Verifying queries** $\mathcal{S}$ installs the value $\gamma$ on $\Upsilon$ oracle. $\Upsilon$ runs the steps 2 and 3 of the **Verify** algorithm. $\Upsilon$ replies to $\mathcal{A}$ with the output of the algorithm. During the verification, to compute the step 2, $\Upsilon$ queries $\mathcal{H}$ with $(m, \Theta)$ for the computation of $H_2$.
   (c) **Hashing queries** proceeds as in Game 0.

   According to the conclusion in Wu et al. (2009), if $Z = e(g_{n+1}, h)$, the Game 1 is identical to Game 0. Otherwise, the value $y_A$ is just a random value. If the adversary can distinguish Game 1 from Game 0, it helps $\mathcal{S}$ to answer the $n$-BDHE problem directly. $\mathcal{S}$ has an advantage, $\epsilon''$, which is the same advantage as $\mathcal{A}$ to distinguish Game 1 from Game 0. The runtime of $\mathcal{C}$ is $t'' < t + 3q_s\tau + 2q_v\tau$.

- **Game 2** In this game, $y_A$ is directly set as a random value and the verifying oracle only checks the signing list.
  1. $\mathcal{S}$ runs **Setup** to get the parameter $sp$ and sends it to $\mathcal{A}$.
  2. $\mathcal{S}$ runs **SKeyGen** to produce a signer's key pair $(sk_S, pk_S)$ and sends $pk_S$ to $\mathcal{A}$.
  3. $\mathcal{S}$ runs **VKeyGen** to produce verifiers' key pairs and sends public keys to $\mathcal{A}$.
  4. $\mathcal{S}$ randomly selects $t \in_R \mathbb{Z}_p^*$, computes $c_1 = g^t, c_2 = (\prod_{i=1}^{n}(R_i))^t$, and sets $y_A = g'^{\beta}$, and sends them to $\mathcal{A}$.
  5. $\mathcal{S}$ answers the signing, verifying, and hashing queries by oracles $\Sigma, \Upsilon, \mathcal{H}$.
     (a) **Signing queries** $\mathcal{S}$ does the same thing as in Game 1. However, $\mathcal{S}$ maintains a $S_{list}$ that stores each signature produced by the oracle $\Sigma$.
     (b) **Verifying queries** The $\Upsilon$ oracle uses the $S_{list}$. If the query is in the $S_{list}$, it replies to $\mathcal{A}$ with 'True'. Otherwise, it replies with 'False'.
     (c) **Hashing queries** It is the same as in Game 0.

  If $\mathcal{A}$ can successfully forge a signature, it can distinguish Game 2 from Game 1 using the verifying oracle $\Upsilon$.

  **Reduction of the forgery:** Suppose the forged signature is ($c_1$,$c_2$, $c_3^*$,$s_e^*$, $s_s^*$) on $m^*$. If the signature is valid, $\mathcal{S}$ computes $\Theta^* = g'^{s_s^*} y_S^{s_e^*}$, finds out the hash value $ran^*$ of the pair $(m^*, \Theta^*)$ in the $H_{list}$.

Then, $\mathcal{S}$ runs $\mathcal{A}$ again with a fixed random tape. When $\mathcal{A}$ queries $(m^*, \Theta^*)$ for hashing, $\mathcal{H}$ replies to $\mathcal{A}$ with the value $ran' = s_e^* - g'^{\alpha} \bmod q'$. By the general forking lemma in Bellare and Neven (2006), $\mathcal{A}$ has a non-negligible probability, $\epsilon(\epsilon/q_h - 1/q')$, of producing another forged signature $(c_1, c_2, c_3', s_e', s_s')$ on $m^*$.

If $s_e^* = s_e'$, then $\Lambda' = s_e' - ran' = s_e^* - ran' = g'^{\alpha} \bmod q'$. It is expected that $c_3 = g'^{\beta\alpha}$ as the value $y_A = g'^{\beta}$. $\mathcal{S}$ takes $c_3$ as a reply to the CDH problem.

There is a bad event turning the CDH solution wrong. Since $|q'| < |p'|$, even if $c \neq ab \bmod q'$, it is possible that $g'^{ab} = g'^{c} \bmod q'$. Fortunately, for the value $g'^{c}$ and another random selected value $g'^{c'}$, the probability of $g'^{c} = g'^{c'} \bmod q'$ is $q'/p'$.

The event $s_e^* = s_e'$ has the probability $1 - \xi$. The success probability of $\mathcal{S}$ is $\epsilon' = (1-\xi)(1-q'/p')\epsilon(\epsilon/q_h - 1/q')$. The runtime of $\mathcal{S}$ is $t' < t + 3q_s\tau$.

Then we can conclude that $\mathcal{A}$ cannot distinguish Game 2 from Game 1 with a non-negligible advantage. $\mathcal{A}$ should produce a forged signature with a non-negligible probability $\epsilon$. Then the 'reduction of the forgery' can be applied again.

The success event of $\mathcal{S}$ is a product of the events that $\mathcal{A}$ cannot distinguish Game 2 from Game 1, and Game 1 from Game 0, and that $\mathcal{A}$ forges successfully, and that the rewinding is successful, and that $s_e^* = s_e'$, and $\Lambda' = g'^{\alpha} \bmod p'$. So $\epsilon' = (1-\epsilon)(1-\epsilon'')\epsilon(\epsilon/q_h - 1/q')(1-\xi)(1-q'/p')$. The runtime in this case is also $t' < t + 3q_s\tau$. ∎

## PSI property

PROPOSITION 3 *Suppose the hash function $H_2$ is a random oracle, the DDH problem is hard, the n-BDHE problem is hard, then the B-SMDVS scheme is $(k, t, \varepsilon)$-PSI-CMA secure.*

*Proof.* Suppose a DDH problem instance $(g', g'^{a}, g'^{b}, g'^{c})$ and an $n$-BDHE problem instance $(g, h, \vec{y}_{g,\alpha,n}, Z)$. Suppose a simulator $\mathcal{S}$ who tries to solve a DDH problem or an $n$-BDHE problem. Suppose an adversary $\mathcal{A}$ who claims to $(k, t, \epsilon)$ distinguish the real signer of a signature of the B-SMDVS scheme, where $\epsilon$ is non-negligible in time $t$ with security parameter $k$. Let $P$ denote the time for a pairing evaluation cost and $\tau$ denote the time for exponentiation.

$\mathcal{S}$ runs a series of games with $\mathcal{A}$ as follows:

- **Game 0:**
  1. $\mathcal{S}$ runs **Setup**, **VKeyGen**, **AKeyGen** in the same way as in Game 0 in the unforgeability proof.
  2. $\mathcal{S}$ runs **SKeyGen** for signers $S0$ and $S1$. The key pair of $S0$ is $(x_{S0}, y_{S0})$, and of $S1$: $(x_{S1}, y_{S1})$. The public keys are given to $\mathcal{A}$.

3. $\mathcal{S}$ answers the signing, verifying, and hashing queries by oracles $\Sigma_0, \Sigma_1$,
$\Upsilon, \mathcal{H}$.
   (a) **Signing queries** $\mathcal{S}$ installs the private key $sk_{S0}$ on the oracle $\Sigma_0$, and $sk_{S1}$ on $\Sigma_1$. $\mathcal{A}$ sends a message $m$ to $\Sigma_0$ or $\Sigma_1$. The signing oracles use the **Sign** algorithm to generate a signature $\delta$ as a reply. They query $\mathcal{H}$ with $(m, \Theta)$ for the computation of $H_2$.
   (b) **Verifying queries and Hashing queries** They are the same as in the Game 0 of the unforgeability proof.
4. $\mathcal{A}$ outputs a message $m^*$ and a state information $I^*$ after a sufficient number of queries. $\mathcal{A}$ gives the message $m^*$ to $\mathcal{S}$.
5. $\mathcal{S}$ flips a fair coin $b \in_R \{0, 1\}$ and uses the oracle $\Sigma_b$ to sign the message $m^*$. The signature $\delta^*$ is given to $\mathcal{A}$.
6. $\mathcal{A}$ takes as input the state information $I^*$, the message-signature pair $(m^*, \delta^*)$, and all public parameters, outputs a guess $b' \in \{0, 1\}$. Before the final output, $\mathcal{A}$ continues to use all oracles with a restriction of not querying the pair $(m^*, \delta^*)$ for verification.

   The game is the same as the experiment $\mathbf{Exp}^{psi-cma}_{B-SMDVS,\mathcal{A}}(k)$. According to the claim of $\mathcal{A}$, the advantage of $\mathcal{A}$, $\mathbf{Adv}^{psi-cma}_{B-SMDVS,\mathcal{A}}$, is $\epsilon$ in time $t$.

- **Game 1** This game is intended to show that the value $y_A$ can just be a random value.
  1. $\mathcal{S}$ runs **Setup**, **SKeyGen** in the same way as in Game 0.
  2. $\mathcal{S}$ produces public keys of verifiers and the agent in the same way as in the Game 1 in the unforgeability proof.
  3. $\mathcal{S}$ answers the signing, verifying, and hashing queries by oracles $\Sigma_0, \Sigma_1$,
  $\Upsilon, \mathcal{H}$.
     (a) **Signing queries and Hashing queries** They are the same as in the Game 0.
     (b) **Verifying queries** It is the same as in the Game 1 in unforgeability proof.
  4. Other steps are kept unchanged.

     According to the conclusion in Wu et al. (2009), if $Z = e(g_{n+1}, h)$, the Game 1 is identical to Game 0. Otherwise, the value $y_A$ is just a random value. If the adversary can distinguish Game 1 from Game 0, it helps $\mathcal{S}$ to answer the $n$-BDHE problem directly.
- **Game 2** $\mathcal{S}$ tries to solve a DDH problem in this game.

1. $\mathcal{S}$ runs **Setup**, **SKeyGen**, **VKeyGen** in the same way as in Game 0.
2. $\mathcal{S}$ randomly selects $t \in_R \mathbb{Z}_p^*$, computes $c_1 = g^t, c_2 = (\prod_{i=1}^n (R_i))^t$, and sets $y_A = g'^b$, and sends them to $\mathcal{A}$.
3. $\mathcal{S}$ answers the signing, verifying, and hashing queries by oracles $\Sigma_0, \Sigma_1$,
$\Upsilon, \mathcal{H}$.
   (a) **Signing queries** $\mathcal{S}$ does the same thing as in Game 1. However, $\mathcal{S}$ maintains a $S_{list}$ that stores each signature produced by the oracles $\Sigma_{S0}$ and $\Sigma_{S1}$.
   (b) **Verifying queries** The $\Upsilon$ oracle uses the $S_{list}$. If the query is in the $S_{list}$, it replies to $\mathcal{A}$ with 'True'. Otherwise, it replies with 'False'.
   (c) **Hashing queries** It is the same as in the Game 1.
4. $\mathcal{A}$ outputs a message $m^*$ and a state information $I^*$ after a sufficient number of queries. $\mathcal{A}$ gives the message $m^*$ to $\mathcal{S}$.
5. $\mathcal{S}$ flips a fair coin $b \in_R \{0, 1\}$ and computes the $\delta^*$ as follows.
   (a) Randomly select $\alpha \in_R \mathbb{Z}_{q'}^*$ and compute $\Theta = g'^{\alpha}$. And set $\Lambda = g'^a$.
   (b) Compute $s_e = H_2(m||\Theta) + \Lambda \bmod q'$ and $s_s = \alpha - x_S sk_{Sb} \bmod q'$.
   (c) Set $c_3 = g'^c$ and $\delta = (c_1, c_2, c_3, s_e, s_s)$
6. $\mathcal{A}$ takes as input the state information $I^*$, the message-signature pair $(m^*, \delta^*)$, and all public parameters, outputs a guess $b' \in \{0, 1\}$. Before the final output, $\mathcal{A}$ continues to use all oracles with a restriction of not querying the pair $(m^*, \delta^*)$ for verification.

At first, $\mathcal{A}$ cannot distinguish Game 2 from Game 1 by the verifying oracle $\Upsilon$. This is simply because the Proposition 2 tell us that the probability of successful forgery is negligible. If all valid signatures queried to the oracle $\Upsilon$ are produced by signing oracles, the behaviors of $\Upsilon$ are the same as in the Game 1. Note that the challenging signature is not allowed to query $\Upsilon$.
Secondly, if the tuple $(g', g^a, g'^b, g'^c)$ is a DDH tuple, the signature $(m^*, \delta^*)$ is a 'valid' signature in Game 1. Otherwise, it is not a signature produced by the Game 1. So, if $\mathcal{A}$ can distinguish Game 1 from Game 0, $\mathcal{S}$ solves the DDH problem.

- **Game 3:** Now, the challenging signature is produced without using any private keys.
  1. $\mathcal{S}$ produces the challenging signature as follows.
     (a) Randomly select $s_s, s_e \in_R \mathbb{Z}_{q'}^*$.
     (b) Compute $\Theta = g'^{s_s} y_S^{s_e}$.

(c) Compute $\Lambda_p = s_e - H_2(m||\Theta)$.

(d) If $\Lambda_p, \Lambda_p + q', \Lambda_p + 2q' \notin \mathbb{G}'$, return to step 1 and reselect $s_e$. Else set $\Lambda = \Lambda_p + \tau q' \in \mathbb{G}'$, $\tau \in \{0, 1, 2\}$, and continue.

(e) Set $c_3 = g'^c$ and set $\delta = (c_1, c_2, c_3, s_e, s_s)$.

2. Other steps are kept unchanged. ■

$\mathcal{A}$ cannot distinguish Game 3 from Game 2 due to the source hiding property.

However, it is meaningless for $\mathcal{A}$ to claim who is the signer in Game 3 since anybody can generate the challenging signature. So there is no advantage in this game.

This gives us the conclusion that $\mathcal{A}$ has only a negligible advantage to distinguish the signer of a signature.

## 6. Comparisons

We compare our scheme with other related schemes in Table 1. The table includes (U)MDVS and (U)SMDVS schemes. If there are more than one related scheme in one reference, we give each scheme one row.

If the size of a signature does not rely on the number of verifiers, we write ‘Fixed’ in the ‘Sig. size’ column. The ‘Variable’ denotes the opposite situation. If the verification does not need private keys, we write ‘No’ in the ‘Ver.’ column. If it needs one private key, we write ‘One’. The ‘All’ means that the **Verify** algorithm claims all verifiers’ private keys as input. The ‘Protocol’ means that all verifiers need to execute a protocol with group communications to verify a signature.

If the simulation algorithm needs all private keys of verifiers, we write ‘All’ in the ‘Sim.’ column. If it needs one private key, we write ‘One’. If there is no formal proof, we write ‘-’ in the column of ‘Model’. The ‘KEA’ denotes a non-black box assumption, Knowledge-of-Exponent Assumption. The ‘RO’ denotes the random oracle model. The ‘Std.’ denotes the standard model.

From the Table 1, we observe the following points:

- Our scheme is the only one that needs one verifier’s private key at the simulation algorithm.
- Among all the SMDVS schemes in the table, there are four schemes in Chow (2006), Shailaja et al. (2006), Laguillaumie and Vergnaud (2007), and Vergnaud (2008), where the **Verify** algorithm only needs one verifier’s private key. All of them have a variable signature size.

The above observations make our scheme unique. Our scheme has a fixed signature size, and only needs one verifier’s private key in the verification and simulation algorithm.

Table 1. Comparisons with related schemes

| Schemes | Type | Sig. size | Ver. | Sim. | Model |
|---|---|---|---|---|---|
| Jakobsson et al. (1996) | MDVS | Fixed | No | All | - |
| Laguillaumie and Vergnaud (2004) | MDVS | Fixed | No | All | - |
| Li et al. (2007) | MDVS | Fixed | No | All | KEA |
| Chow (2008) | MDVS | Fixed | No | All | - |
| Chow (2006) | ID-SMDVS | Variable | One* | All | RO |
| Laguillaumie and Vergnaud (2007) | SMDVS | Variable | One | All | RO |
| Ng et al. (2005) | USMDVS | Variable | All | All | - |
| Ng et al. (2005) | USMDVS | Fixed | Protocol | All | RO |
| Ming and Wang (2008) | USMDVS | Fixed | All | All | Std. |
| Shailaja et al. (2006) | USMDVS | Variable | One | All | Std. |
| Vergnaud (2008) | UMDVS | Fixed | No | All | Std. |
| Vergnaud (2008) | USMDVS | Variable | One* | All | RO |
| Seo et al. (2008) | ID-USMDVS | Fixed | Protocol | All | RO |
| Chang (2011) | ID-USMDVS | Fixed | All | All | RO |
| Our Scheme | B-SMDVS | Fixed | One | One | RO |

* Each verifier needs two times paring evaluations with all other verifiers

## 7. Conclusion

This paper proposed a strong multiple designated verifiers signature for broadcast mode. It is suitable to application scenarios where multiple servers collaboratively provide a privacy-enabled service with a common application gateway for clients.

## Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant Nos. 60970144, 60803135, 61070168, 61003244,61103199), the Fundamental Research Funds for the Central Universities (Grant Nos.10lgpy31, 11lgpy71, 11lgzd06, 10lgzd14), the Doctoral Fund of Ministry of Education of China for New Teachers (Grant No. 20090171120006), the Beijing Municipal Natural Science Foundation (Grant No. 4112052), and the Industry-Education-Research Cooperation Project of Guangdong Province and the Ministry of Education (Grant No.2011A090200068). We would like to express our thanks to the chairs, committee, and reviewers of the security track of the INCoS 2011 conference.

## 8. References

Bellare, M. and Neven, G. (2006) Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma. In: *Proceedings of the 13th ACM Conference on CCS*, Alexandria, Virginia. ACM Press, 390–399.

Chang, T.Y. (2011) An ID-based multi-signer universal designated multi-verifier signature scheme. *Information and Computation*, **209**(7), 1007–1015.

Chow, S.S.M. (2006) Identity-based Strong Multi-Designated Verifiers Signatures. *Lecture Notes in Computer Science*, **4043**, 257–259.

CHOW, S.S.M. (2008) Multi-Designated Verifiers Signatures Revisited. *International Journal of Network Security*, **7**(3), 348–357.

DESMEDT, Y. (2003)Verifier-Designated Signatures. In: *Rump Session of CRYPTO 2003*, Santa Barbara, California. Springer.

JAKOBSSON, M., SAKO, K. and IMPAGLIAZZO, R. (1996) Designated Verifier Proofs and Their Applications. *Lecture Notes in Computer Science*, **1070**, 143–154.

LAGUILLAUMIE, F. and VERGNAUD, D. (2004) Designated verifiers signature: anonymity and efficient construction from any bilinear map. *Lecture Notes in Computer Science*, **3352**, 107–121.

LAGUILLAUMIE, F. and VERGNAUD, D. (2004) Multi-Designated Verifiers Signatures. *Lecture Notes in Computer Science*, **3269**, 495–507.

LAGUILLAUMIE, F. and VERGNAUD, D. (2007) Multi-designated Verifiers Signatures: Anonymity without Encryption. *Information Processing Letters*, **102**(2–3), 127–132.

LI, Y., SUSILO, W., MU, Y., and PEI, D. (2007) Designated Verifier Signature: Definition, Framework and New Constructions. *Lecture Notes in Computer Science*, **4611**, 1191–1200.

MING, Y. and WANG, Y. (2008) Universal designated multi verifier signature scheme without random oracles. *Wuhan Univ. J. Nat. Sci.*, **13**(6), 685–691.

NG, C.Y., SUSILO, W. and MU, Y. (2005) Universal designated multi verifiers signature schemes. In: *Proceedings of the ICPADS 2005*, Fukuoka, Japan. IEEE, 305–309.

RIVEST, R.L., SHAMIR, A. and TAUMAN, Y. (2001) How to leak a secret. *Lecture Notes in Computer Science*, **2248**, 552–565.

SAEEDNIA, S., KRAMER, S., and MARKOVITCH, O. (2004) An efficient strong designated verifier signature scheme. *Lecture Notes in Computer Science*, **2971**, 40–54.

SCHNORR, C.P. (1990) Efficient identification and signatures for smart cards. *Lecture Notes in Computer Science*, **435**, 239–252.

SEO, S.H., HWANG, J.Y., CHOI, K.Y. and LEE, D.H. (2008) Identity-based universal designated multi-verifiers signature schemes. *Comput. Standards & Interfaces*, **30**(5), 288–295.

SHAILAJA, G., KUMAR, K.P. and SAXENH, A. (2006) Universal designated multi verifier signature without random oracles. In: *Proceedings of the ICIT 2006*, Bhubaneswar, India. IEEE, 168–171.

STEINFIELD, R., WANG, H. and PIERPRZYK, J. (2004) Efficient Extension of Standard Schnorr/RSA signatures into Universal Designated-Verifier Signatures. *Lecture Notes in Computer Science*, **2947**, 86–100.

USHIDA, M., IZU, T., TAKENAKA, M. and OHTA, K. (2010) Multiple Designated Verifiers Signatures Reconsidered. In: *Proceedings of the fifth International conference on Availability, Reliability and Security,* Kraków, Poland. IEEE, 586–590.

Vergnaud, D. (2008) New Extensions of Pairing-based Signatures into Universal (Multi) Designated Verifier Signatures. *Lecture Notes in Computer Science*, **4052**, 58–69.

Wu, Q., Mu, Y., Susilo, W., Qin, B., and Domingo-Ferrer, J. (2009) Asymmetric Group Key Agreement. *Lecture Notes in Computer Science*, **5479**, 153–170.