Wireless Pers Commun (2017) 97:3017–3030
DOI 10.1007/s11277-017-4658-5

Designated Verifier Proxy Re-signature for Deniable and Anonymous Wireless Communications

Jiannan Wei[1] · Guomin Yang[2] · Yi Mu[2]

Published online: 1 September 2017

Abstract In this paper, we introduce a new cryptographic primitive named Designated Verifier Proxy Re-Signature (DVPRS). Different from a normal proxy re-signature, our DVPRS is defined based on the notion of Designated Verifier Signature (DVS) which is very useful in many applications that require "deniable authentication". Since a DVS can only be verified by a designated verifier, in addition to the re-sign algorithm which allows a proxy to use a resign key to change the signer of a DVS on a message, we also define the re-designate-verifier algorithm for DVPRS which allows a proxy to change the designated verifier of a DVS. We present the formal definition, security model, and an efficient construction of DVPRS, and prove its security under some standard assumptions. We show that DVPRS is very useful in many communication and network applications that require deniable and/or anonymous authentication.

Keywords Proxy re-signature · Designated verifier signature · Anonymous communications · Deniable authentication

✉ Jiannan Wei
jnwei@njust.edu.cn

Guomin Yang
gyang@uow.edu.au

Yi Mu
ymu@uow.edu.au

[1] School of Computer Science and Engineering, Nanjing University of Science and Technology, 210094 Nanjing, China

[2] School of Computing and Information Technology, Centre for Information and Computer Security Research, University of Wollongong, Wollongong, NSW 2522, Australia

1 Introduction

Proxy re-signature was introduced by Blaze, Bleumer and Strauss (BBS) in Eurocrypt 1998 [3]. In a proxy re-signature (PRS) scheme, there is a semi-trusted third party called the proxy, who holds a resign key and can use it to transform a valid signature σ_A of *Alice* on message m to a valid signature σ_B of *Bob* on the same message. We say a PRS is *bidirectional* if the proxy can use a resign key to do the transformation in both directions. Also, if the signature transformation can be performed in sequence for multiple times, then we say a PRS is multi-use.

In a normal PRS scheme, a signature is publicly verifiable, no matter it is generated by the sign or resign algorithm. However, in some applications, we require *deniable authentication* where Alice can prove herself to Bob, but Bob cannot transfer this proof to anyone else. In the literature, such a non-transferability property can be achieved by a Designated Verifier Signature (DVS) scheme, which was introduced by Jakobsson, Sako and Impagliazzo in Eurocrypt 1996 [12]. In the same paper, Jakobsson et al. also introduced the notion of Strong DVS (SDVS) which means the identity of a signer can be hidden from outsiders. Below we show that in some applications we need the features of both PRS and (S)DVS.

Suppose Alice and Bob are negotiating a contract on behalf of their companies, denoted by A and B, respectively. In order to make sure that they are receiving the genuine negotiation messages from each other, the conversations must be authenticated. However, they also want to prevent the other party from using the (authenticated) negotiation messages as a tool to bargain with other competitors. Therefore, a DVS is preferable in this case. Now suppose that Alice is on leave and Bob has sent a DVS σ_{BA} to Alice, which can only be verified by Alice. If there exists a proxy who can use a re-designate-verifier key to change the designated verifier in σ_{BA} to another user, say Carol in company A (i.e., the proxy converts σ_{BA} into another DVS σ_{BC}), then Carol can perform the verification on behalf of Alice. Also, if the proxy (can be a different one) has a re-sign key that can be used to convert Carol's DVS σ_{CB} for Bob into another valid DVS σ_{AB} of Alice, then Carol can also compose a response to Bob on behalf of Alice.

In this paper, we introduce a new cryptographic primitive named Designated Verifier Proxy Re-Signature (DVPRS), which has the properties of both PRS and SDVS, and can be used to solve the problem described above. A DVPRS scheme can also be useful in some network applications. Consider a wireless mobile network (WMN) which contains a number of mobile nodes $(M_1, M_2, \ldots, M_n)$ and a server node S. Suppose a mobile node M_i wants to *anonymously* broadcast a message m to a subset of mobile nodes (denoted by $P = \{M_j, M_k, \ldots, M_l\}$) such that all the nodes in P can verify the authenticity of the message m, while other mobile nodes in the network cannot perform the verification or even identify the sender of the message. A simple solution is to let M_i generate a SDVS on the message m for each node in P, and broadcast the message with all the signatures. However, this trivial solution is not suitable for constrained mobile nodes (such as sensors) which have very limited computation and communication power. A DVPRS scheme can provide a better solution in this case. Similar to the previous example, we assume the server node S now serves as a (semi-trusted) proxy for all the mobile nodes. In order to anonymously broadcast a message m to all the mobile nodes in P, M_i generates a SDVS σ_{ij} for the first receiver $M_j \in P$, and then sends (m, σ_{ij}, P) to S, who can then use the re-designate-verifier keys to transform σ_{ij} to σ_{it} for every $M_t \in P$, and then broadcast the message and all the signatures. If there exists a secure channel between M_i and S, then an

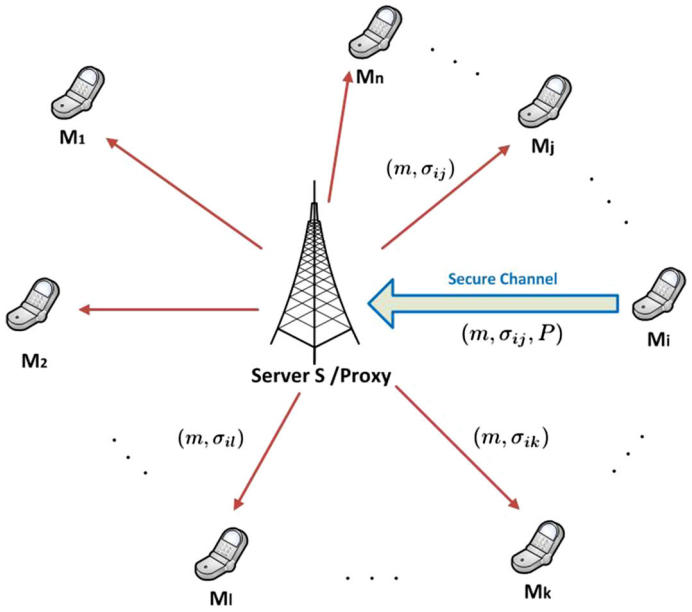

Fig. 1 DVPRS for anonymous wireless communications

outsider cannot even identify the receivers of the message. We illustrate this example in Fig. 1.

Based on the applications of a DVPRS given above, we can (informally) summarise the desirable security features of a DVPRS scheme as follows.

- *Unforgeable*: similar to the unforgeability requirement of a DVS, only the signer or the designated verifier can produce a valid signature. In particular, the proxy cannot forge a valid DVS from scratch even if he has the re-sign and re-designate-verifier keys.
- *Non-transferable*: the designated verifier cannot use a DVS to convince a third party that the DVS is generated by the signer.
- *Anonymity/Invisibility*: a DVS cannot reveal the identity of the signer or the designated verifier.
- *Transparency*: a DVS generated by the signing algorithm should be computationally indistinguishable from those generated by the resign and re-designate-verifier algorithms.

Similar to a normal PRS, we can also have unidirectional and bidirectional DVPRS. In a unidirectional DVPRS, given a resign (or re-designate-verifier) key $rk_{A\to B}$, the proxy can only convert a signature of A to a new signature of B, but not the other direction. However, in a bidirectional DVPRS, the proxy can use $rk_{A\to B}$ to do the conversion in both directions. A unidirectional DVPRS can provide better security while a bidirectional DVPRS is more efficient in terms of key management. In some large-scale systems, such as the example illustrated in Fig. 1, a bidirectional DVPRS is more suitable since the number of rekeys maintained by the server can be reduced significantly given the large number of users in the system. Also, in this example, it is natural to allow the server to do conversion in both directions.

1.1 Our Contributions

In this paper, we formalise the notion of DVPRS and propose the first practical DVPRS scheme. The contributions of this work can be summarised as follows.

- We define the formal definition of DVPRS and present a formal security model for capturing the (informal) security requirements of a DVPRS scheme listed above.
- We also propose a multi-use bidirectional DVPRS scheme that is proven secure under the standard Bilinear Diffi–Hellman (BDH) and Decisional Bilinear Diffie–Hellman (DBDH) assumptions in the random oracle model. Our proposed DVPRS scheme is very efficient. In particular, the resign and re-designate-verifier algorithms only require one exponentiation operation.

1.2 Related Work

Designated verifier signature. DVS was introduced by Jakobsson, Sako and Impagliazzo in [12]. In the same paper, they also proposed the concept of Strong DVS (SDVS), which means no third party can tell the identity of the signer who generated a given SDVS. The authors also suggested a generic approach to construct SDVS based on public-key encryption. The idea is rather simple, the signer first generates a DVS and then encrypts the signature using the receiver's public key. In this way the signer's identity can be protected by the encryption function. The formal definition of a SDVS was proposed by Laguillaumie and Vergnaud [14] where the "privacy of signer's identity" was formally defined.

Some variants of DVS were introduced following the seminal work of Jakobsson et al. [12], such as identity-based DVS (IBDVS) [7, 8, 13, 20], short DVS [10, 11] where the size of a DVS is short, and multi-verifier DVS (MDVS) [15], in which the signer can designate multiple parties as verifiers. The notion of universal designated verifier signature (UDVS) [9, 19, 21, 23] was introduced by Steinfeld et al. [19] in Asiacrypt 2003. In a UDVS, a user can transform a standard digital signature produced by a signer into a DVS for any designated verifier.

We should note that our proposed DVPRS scheme shares some similarity with the short DVS scheme proposed by Huang et al. [10, 11]. In their scheme, the DVS is constructed using $H(m\|g^{xy})$ where (g^x, x) and (g^y, y) are the key pairs of the signer and the designated verifier, respectively. However, the scheme cannot support the resign or re-designated-verifier operation due to the hash function. Our scheme resolves this problem by replacing the hash function with a bilinear map function. We should also note that the security models and proofs are completely different between this work and [10, 11] since we need to consider proxy resigning and re-designating in our scheme.

Proxy re-signature Proxy re-signature (PRS) was proposed by Blaze, Bleumer and Strauss [3] in Eurocrypt 1998. However, this notion was ignored by the cryptographic research community until in CCS 2005 [1], Ateniese and Hohenberger showed that proxy re-signature is very useful in many applications such as sharing web certificates and authenticating a network path. Ateniese and Hohenberger also discovered a weakness in Blaze et al.'s PRS: the resign key can be derived from a valid signature and its transformed re-signature. Ateniese and Hohenberger also proposed the formal definition of security for a PRS, and two PRS schemes based on bilinear maps. Their first scheme, which is based on the Boneh-Lynn-Shacham (BLS) short signature [4], is bidirectional and multi-use, while their second scheme is unidirectional and single-use.

Since the work of Ateniese and Hohenberger [1], many other proxy re-signature schemes were proposed. In [17], based on the technique of Waters [22], Shao et al. proposed two efficient bidirectional multi-use PRS schemes (one under the identity-based setting) in the standard model. However, a security flaw in these two schemes was later discovered by Chow et al. [5]. In the same paper, Chow et al. presented a generic construction of unidirectional PRS using a new primitive named homomorphic compartment signature. Libert and Vergnaud [16] also proposed two unidirectional multi-hop PRS schemes, one in the random oracle model [2] and the other in the standard model, based on some newly defined assumptions. One major issue in Libert and Vergnaud's constructions is that the signature size grows linearly in the number of resign operations. In [18], Shao et al. further improved the security model for unidirectional PRS schemes, and showed that several previous schemes are still secure in their new model.

1.3 Paper Organization

The rest of this paper is organized as follows. In Sect. 2, we give the definition and formal security model for DVPRS. We then present our multi-use bidirectional DVPRS scheme in Sect. 3, and prove its security in Sect. 4. We conclude the paper in Sect. 5.

2 DVPRS: Formal Definition and Security Model

2.1 Formal Definition of Designated Verifier Proxy Re-signature

Definition 1 A designated verifier proxy re-signature scheme consists of a tuple of probabilistic polynomial-time (PPT) algorithms $\mathcal{DVPRS} = (KeyGen, ReSKey, ReDVKey, Sign, ReSign, ReDV, Verify)$ which are defined below.

- $KeyGen(1^{\lambda})$: Given the security parameter 1^{λ} as input, this algorithm outputs a pair of public and private keys (pk, sk).
- $ReSKey(pk_A, sk_A, pk_B, sk_B)$: The re-sign key generation algorithm takes as input the original signer's public key pk_A and private key sk_A, and the new signer's public key pk_B and private key sk_B, and outputs a resign key $rsk_{A\to B}$.
- $ReDVKey(pk_C, sk_C, pk_D, sk_D)$: The re-designate-verifier key generation algorithm takes as input the origianl designated verifier's public key pk_C and private key sk_C, and the new designated verifier's public key pk_D and private key sk_D, and outputs a re-designate-verifier key $rvk_{C\to D}$.
- $Sign(sk_A, pk_C, m)$: The designated verifier signing algorithm takes the signer's private key sk_A, and the designated verifier's public key pk_C, and a message m as input and outputs a signature σ_{AC}.
- $ReSign(rsk_{A\to B}, \sigma_{AC}, m)$: The resign algorithm takes as input a resign key $rsk_{A\to B}$, and a designated verifier signnature σ_{AC} for a message m, and outputs a new designated verifier signature σ_{BC} for the same designated verifier and message.
- $ReDV(rvk_{C\to D}, \sigma_{AC}, m)$: The re-designate-verifier algorithm takes as input a re-designate-verifier key $rvk_{C\to D}$, and a designated verifier signnature σ_{AC} for a message m, and outputs a new signature σ_{AD} for the same message but a new designated verifier.
- $Verify(pk_A, sk_C, m, \sigma_{AC})$: The verification algorithm takes as input the signer's public key and the designated verifier's secret key, a message m, and a designated verifier signature σ_{AC}, and outputs '1' if the signature is valid, or '0' otherwise.

Correctness: For any $\lambda \in \mathbb{N}$, $(pk_A, sk_A), (pk_B, sk_B), (pk_C, sk_C), (pk_D, sk_D)$ generated by running $KeyGen(1^\lambda)$,

$\sigma_{AC} \leftarrow Sign(sk_A, pk_C, m)$,
$rsk_{A\to B} \leftarrow ReSKey(pk_A, sk_A, pk_B, sk_B)$,
$rvk_{C\to D} \leftarrow ReDVKey(pk_C, sk_C, pk_D, sk_D)$, the following conditions must hold:

$$\begin{aligned} Verify(pk_A, sk_C, m, \sigma_{AC}) &= 1; \\ Verify(pk_B, sk_C, m, ReSign(rsk_{A\to B}, \sigma_{AC}, m)) &= 1; \\ Verify(pk_A, sk_D, m, ReDV(rvk_{C\to D}, \sigma_{AC}, m)) &= 1. \end{aligned}$$

2.2 Security Models

There are three different entities involved in a DVPRS, namely, signer, verifier, and proxy. Below we define three basic security requirements of a DVPRS, namely unforgeability, invisibility, and transparency.

2.2.1 Unforgeability

Unforgeability means no outsider or proxy can forge a valid signature on behalf of an honest signer. Formally, we say a DVPRS scheme is unforgeable if for any $\lambda, n \in \mathbb{N}$ and any PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c} \{(pk_i, sk_i) \leftarrow KeyGen(1^\lambda)\}_{i\in[1,n]}, \\ PK = \{pk_i\}_{1\le i\le n}, \quad (t, v, m^*, \sigma^*) \\ \leftarrow \mathcal{A}^{\mathcal{O}_{ReSKey}, \mathcal{O}_{ReDVKey}, \mathcal{O}_{Sign}, \mathcal{O}_{Verify}}(PK) \\ : Verify(pk_t, sk_v, m^*, \sigma^*) = 1 \wedge m^* \notin Q \end{array}\right] \le \epsilon(\lambda)$$

where Q denote the set of messages that have appeared in the oracle queries made by $\mathcal{A}$, and $\epsilon(\cdot)$ is a negligible function. The oracles are defined as follows.

- $\mathcal{O}_{ReSKey}$: by making a query with input (i, j) $(i \neq j)$ to this oracle, the adversary obtains $rsk_{i\to j} \leftarrow ReSKey(pk_i, sk_i, pk_j, sk_j)$;
- $\mathcal{O}_{ReDVKey}$: by making a query with input (i, j) $(i \neq j)$ to this oracle, the adversary obtains $rvk_{i\to j} \leftarrow ReDVKey(pk_i, sk_i, pk_j, sk_j)$;
- $\mathcal{O}_{Sign}$: by making a query with input (i, j, m) $(i \neq j)$ to this oracle, the adversary obtains $\sigma \leftarrow Sign(sk_i, pk_j, m)$;
- $\mathcal{O}_{Verify}$: by making a query with input (i, j, m, σ) to this oracle, the adversary obtains $b \leftarrow Verify(pk_i, sk_j, m, \sigma)$.

Remark 1 In the unforgeability game, we don't allow *ReSign* or *ReDV* oracle queries since we consider the proxy as a potential adversary. The adversary can use *ReSKey* and *ReDVKey* to obtain the resign and re-designate-verifier keys, and hence can perform *ReSign* and *ReDV* operations by itself.

2.2.2 *Invisibility*

Invisibility means without the help from either the signer or the designated verifier, a DVS should look like a random string (i.e., the anonymity of the signer and the verifier is well preserved). However, different from the unforgeability game, in the invisibility game we don't treat the proxy as an adversary. The reason is that the proxy will perform the resign and re-designate-verifier operations, and hence will certainly know the identities of the previous signer (or verifier) and the next one. However, we should allow the adversary to access the ReSign and ReDV oracles that are defined below.

- $\mathcal{O}_{ReSign}$: the adversary makes a query with input (k, j, σ, m, i) to this oracle. If $Verify(pk_k, sk_j, m, \sigma) = 1$, the oracle returns $\sigma' \leftarrow ReSign(rsk_{k\to i}, \sigma, m)$ to $\mathcal{A}$; otherwise, a special symbol '$\perp'$ is returned.
- $\mathcal{O}_{ReDV}$: the adversary makes a query with input (i, k, σ, m, j) to this oracle. If $Verify(pk_i, sk_k, m, \sigma) = 1$, the oracle returns $\sigma' \leftarrow ReDV(rvk_{k\to j}, \sigma, m)$ to $\mathcal{A}$; otherwise, a special symbol '$\perp'$ is returned.
- $\mathcal{O}_{Sign}$ and $\mathcal{O}_{Verify}$ are the same as in the unforgeability game.

We define invisibility via the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- *Setup*: $\mathcal{C}$ runs the key generation algorithm to generate (pk_i, sk_i) for all $1 \le i \le n$, and returns all the public keys to the adversary $\mathcal{A}$.
- Training Phase 1: $\mathcal{A}$ is allowed to make queries to $\mathcal{O}_{Sign}$, $\mathcal{O}_{Verify}$, $\mathcal{O}_{ReSign}$, and $\mathcal{O}_{ReDV}$ defined above.
- *Challenge Phase*: The adversary $\mathcal{A}$ outputs (i, j, m^*) such that a signature (denoted by σ_{i,j,m^*}) with signer i, designated verifier j and message m^* has not been explicitly or implicitly returned to $\mathcal{A}$ in any oracle queries. That is, $\mathcal{A}$ is not allowed to directly obtain σ_{i,j,m^*} through a signing query, or a sequence of Sign and Re-sign (or ReDV) oracle queries.
  The challenger $\mathcal{C}$ tosses a random coin $b \in \{0, 1\}$. If $b = 0$, $\mathcal{C}$ computes a valid signature $\sigma \leftarrow Sign(sk_i, pk_j, m^*)$ and sets $\sigma^* = \sigma$. Otherwise, $\mathcal{C}$ selects a fake signature R uniformly at random from the signature space and sets $\sigma^* = R$. Finally $\mathcal{C}$ returns σ^* to $\mathcal{A}$.
- *Training Phase 2*: Same as Training Phase 1 with the restriction that $\mathcal{A}$ is not allowed to directly obtain σ_{i,j,m^*} through a signing query (or a sequence of Sign and Re-sign (or ReDV) oracle queries), and cannot query the verification oracle with the challenge signature (i, j, m^*, σ^*).
- *Output Phase*: $\mathcal{A}$ outputs a bit b'.

The advantage of $\mathcal{A}$ is defined as

$$Adv_{\mathcal{A}}^{invis}(\lambda) = \Pr[b' = b] - 1/2.$$

We say a DVPRS scheme has invisibility if $Adv_{\mathcal{A}}^{invis}(\lambda)$ is negligible in λ for any PPT adversary $\mathcal{A}$.

Remark 2 In the invisibility game, we don't allow the adversary to make a signing query with regards to the challenge identities and message (i, j, m^*). Such a restriction is necessary when we consider a deterministic signature scheme, since otherwise the adversary can trivially win the invisibility game. However, we should remark that for a randomised signature scheme, it is possible to remove this restriction.

Signer & Verifier Anonymity. Another security notion that is related to invisibility is signer anonymity. Informally, signer anonymity means a DVS σ_i generated by user i is indistinguishable from a DVS σ_j generated by user j. It is easy to see that invisibility implies anonymity since if both σ_i and σ_j are indistinguishable from random signatures, then by a hybrid argument they are also indistinguishable from each other. A similar result has been obtained in [6] for undeniable and designated confirmer signatures. Similarly, we can also define verifier anonymity, and by a similar analysis, we can conclude that invisibility also implies verifier anonymity.

2.2.3 Transparency

The definition of invisibility above only ensures that an original DVS generated by the Sign algorithm is invisible. In order to capture the requirement that signatures derived from the ReSign and ReDV algorithms are also invisible, below we define the requirement of Transparency, which means signatures derived from the ReSign and ReDV algorithms should be indistinguishable from those generated from the Sign algorithm.

We say a DVPRS has transparency if for any $\lambda \in \mathbb{N}$ and any message m, and any PPT algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$

$$\Pr\left[\begin{array}{c} b \leftarrow \{0,1\}, \\ \{(pk_i, sk_i) \leftarrow KeyGen(1^\lambda)\}_{i\in\{A,B,C\}}, \\ rsk_{A\to B} \leftarrow ReSKey(pk_A, sk_A, pk_B, sk_B), \\ \sigma_0 \leftarrow Sign(sk_B, pk_C, m), \\ \sigma_1 \leftarrow ReSign(rsk_{A\to B}, Sign(sk_A, pk_C, m), m), \\ b' \leftarrow \mathcal{A}_1(1^\lambda, pk_A, pk_B, pk_C, \sigma_b) : b' = b \end{array}\right] \leq \frac{1}{2} + \epsilon(\lambda),$$

$$\Pr\left[\begin{array}{c} b \leftarrow \{0,1\}, \\ \{(pk_i, sk_i) \leftarrow KeyGen(1^\lambda)\}_{i\in\{A,B,C\}}, \\ rvk_{B\to C} \leftarrow ReDVKey(pk_B, sk_B, pk_C, sk_C), \\ \sigma_0 \leftarrow Sign(sk_A, pk_C, m), \\ \sigma_1 \leftarrow ReDV(rvk_{B\to C}, Sign(sk_A, pk_B, m), m), \\ b' \leftarrow \mathcal{A}_2(1^\lambda, pk_A, pk_B, pk_C, \sigma_b) : b' = b \end{array}\right] \leq \frac{1}{2} + \epsilon(\lambda),$$

where $\epsilon(\cdot)$ is a negligible function.

3 An Efficient DVPRS Scheme

3.1 Preliminaries

Let G_1 denote an additive cyclic group with prime order q defined by an elliptic curve over a finite field, and G_2 a multiplicative cyclic group of the same order. Let $e : G_1 \times G_1 \to G_2$ be a bilinear map with the following properties:

1. *Bilinearity*: $e(aP, bQ) = e(P, Q)^{ab}$ for any $P, Q \in G_1$, and $a, b \in Z_q^*$.
2. *Non-degeneracy*: There exists $P \in G_1$ and $Q \in G_1$ such that $e(P, Q) \neq 1$.
3. *Computability*: There is an efficient algorithm to compute $e(P, Q)$ for any $P, Q \in G_1$.

The following computational problems related to bilinear map are believed to be intractable.

Definition 2 (*Bilinear Diffie–Hellman (BDH) Problem*) Given $(P, aP, bP, cP) \in G_1^4$, where P is a generator of G_1 and a, b, c are randomly chosen from Z_q, compute $e(P,P)^{abc}$.

Definition 3 (*Decisional Bilinear Diffie–Hellman (DBDH) Problem*) Given $(P, aP, bP, cP) \in G_1^4$, distinguish $e(P,P)^{abc}$ from $e(P,P)^r$, where P is a generator of G_1 and a, b, c, r are randomly chosen from Z_q.

3.2 Our DVPRS Scheme

Let e, G_1, G_2, g, q denote a bilinear group defined as above where g is a generator of G_1, and $H : \{0,1\}^* \rightarrow G_1$ a cryptographic hash function. Our DVPRS works as follows.

- *KeyGen*(1^λ): on input the security parameter 1^λ, randomly select $x_1, x_2 \in \mathbb{Z}_q$, output the public and secret key pair as $(pk = (g^{x_1}, g^{x_2}), sk = (x_1, x_2))$.
- *ReSKey*(pk_A, sk_A, pk_B, sk_B): on input $pk_A = (g^{a_1}, g^{a_2}), sk_A = (a_1, a_2)$ and $pk_B = (g^{b_1}, g^{b_2}), sk_A = (b_1, b_2)$, return the resign key as $rsk_{A\rightarrow B} = b_1/a_1 \bmod q$.
- *ReDVKey*(pk_C, sk_C, pk_D, sk_D): on input $pk_C = (g^{c_1}, g^{c_2}), sk_C = (c_1, c_2)$ and $pk_D = (g^{d_1}, g^{d_2}), sk_D = (d_1, d_2)$, return the re-designate-verifier key as $rvk_{C\rightarrow D} = d_2/c_2 \bmod q$.
- *Sign*(sk_A, pk_C, m): the signing algorithm takes a signer's private key $sk_A = (a_1, a_2)$, a designated verifier's public key $pk_C = (g^{c_1}, g^{c_2})$, and a message m as input, and outputs a designate verifier signature $\sigma_{AC} = e(H(m), g^{c_2})^{a_1}$.
- *ReSign*$(rsk_{A\rightarrow B}, \sigma_{AC}, m)$: the resign algorithm takes as input a resign key $rsk_{A\rightarrow B} = b_1/a_1 \bmod q$, and a designated verifier signnature $\sigma_{AC} = e(H(m), g^{c_2})^{a_1}$ for a message m, and outputs a new designated verifier signature $\sigma_{BC} = \sigma_{AC}^{rsk_{A\rightarrow B}} = e(H(m), g^{c_2})^{b_1}$.
- *ReDV*$(rvk_{C\rightarrow D}, \sigma_{AC}, m)$: the re-designate-verifier algorithm takes as input a re-designate-verifier key $rvk_{C\rightarrow D} = d_2/c_2 \bmod q$, and a designated verifier signnature $\sigma_{AC} = e(H(m), g^{c_2})^{a_1}$ for a message m, and outputs a new designated verifier signature $\sigma_{AD} = \sigma_{AC}^{rvk_{C\rightarrow D}} = e(H(m), g^{d_2})^{a_1}$.
- *Verify*$(pk_A, sk_C, m, \sigma_{AC})$: on input the signer's public key $pk_A = (g^{a_1}, g^{a_2})$ and the designated verifier's secret key $sk_C = (c_1, c_2)$, a message m, and a designated verifier signature $\sigma_{AC} = e(H(m), g^{c_2})^{a_1}$, if $e(\sigma_{AC}, g) = e(H(m), g^{a_1})^{c_2}$, output '1'; else, output '0'.

It is easy to see that our proposed DVPRS scheme is bidirectional, since given a resign key $rsk_{A\rightarrow B} = b_1/a_1$, we can directly derive $rsk_{B\rightarrow A} = rsk_{A\rightarrow B}^{-1} = a_1/b_1$. The re-designate-verifier key has the same structure and hence is also bidirectional.

Remark 3 The correctness of our scheme can be verified easily. Our scheme can do the resign and re-designate-verifier operations separately. Therefore, the resign key and re-designate-verifier key can be distributed to different proxies who can do their respective operations seperately. Also, if a proxy holds both a resign key $rsk_{A\rightarrow B}$ and a re-designate-verifier key $rvk_{C\rightarrow D}$, then given a signature σ_{AC}, the proxy can convert it into σ_{BD} in one shot via

$$\sigma_{BD} = \sigma_{AC}^{rsk_{A\to B}\cdot rvk_{C\to D}}$$

rather than running the *ReSign* and *ReDV* algorithms seperately.

Remark 4 Our DVPRS requires both sk_A and sk_B in order to generate $rsk_{A\to B}$ or $rvk_{A\to B}$. In order to make sure that the secret key is not leaked to other users, we can use the following method that has been used by other proxy resignature schemes: to compute $rsk_{A\to B} = b_1/a_1$, the proxy chooses a random number r and sends it to B, who computes and sends $r \cdot b_1$ to A. A then computes and sends rb_1/a_1 to the proxy, who finally removes r and derives b_1/a_1.

4 Security Analysis

In this section, we provide formal security analysis for our proposed DVPRS scheme. We prove that our new DVPRS scheme can achieve unforgeability, invisibility, and transparency under some standard assumptions.

Theorem 1 *The proposed DVPRS scheme is unforgeable under the Bilinear Diffie–Hellman assumption.*

Proof If there exist an adversary $\mathcal{A}$ which performs at most q_H hash queries and can break the unforgeability of the proposed DVPRS scheme with non-negligible probability ϵ, we can construct another adversary $\mathcal{A}'$ that can solve the BDH problem with probability ϵ/q_H. On input a BDH tuple (g, g^a, g^b, g^c), the algorithm $\mathcal{A}'$ simulates the designated verifier proxy re-signature unforgeability game for $\mathcal{A}$ as follows:

- *Setup:* The simulator $\mathcal{A}'$ randomly chooses $\alpha_i, \beta_i \in \mathbb{Z}_p$ for $1 \le i \le n$, and sets the public key of user i as $\{pk_i = ((g^a)^{\alpha_i}, (g^c)^{\beta_i})\}$. $\mathcal{A}'$ gives all the user public keys to $\mathcal{A}$. Let q_H denote the maximum number of hash queries that $\mathcal{A}$ will make in the game. $\mathcal{A}'$ randomly selects an index $j \in [1, q_H]$, and mantains a hash table which records the input and output of each hash query.
- *Training:* $\mathcal{A}'$ answers $\mathcal{A}$'s queries as follows:
  - *Hash queries*: For each hash query made by $\mathcal{A}$, $\mathcal{A}'$ first checks the hash table to see if an entry for the same input exists in the hash table. If so, the same hash output will be returned to $\mathcal{A}$. For the i-th hash query, if $i = j$, then $\mathcal{A}'$ returns g^b to $\mathcal{A}$; otherwise, $\mathcal{A}'$ selects a random $y_i \in \mathbb{Z}_q$ and outputs g^{y_i} to $\mathcal{A}$.
  - *Signing queries*: For each signing query with input (i, j, m), $\mathcal{A}'$ first simulates a hash query on message m, if the hash output is g^b (i.e., m is the input of the j-th hash query), then $\mathcal{A}'$ aborts the game; otherwise, the hash output has the form g^y where y is chosen by $\mathcal{A}'$, then a signature $\sigma = e(g^a, g^c)^{y\alpha_i\beta_j}$ is returned to $\mathcal{A}$.
  - *Verification queries*: For each verification query with input (i, j, m, σ), $\mathcal{A}'$ first simulates a signing query with input (i, j, m) to obtain a signature σ', since the signing algorithm is deterministic, $\mathcal{A}'$ returns '1' if $\sigma = \sigma'$, and '0' otherwise.
  - *ReSKey queries*: For each query to $\mathcal{O}_{ReSKey}$ with input (i, j), $\mathcal{A}'$ returns $rsk_{i\leftarrow j} = (a\alpha_j)/(a\alpha_i) = \alpha_j/\alpha_i$ to $\mathcal{A}$.
  - *ReDVKey queries*: For each query to $\mathcal{O}_{ReDVKey}$ with input (i, j), $\mathcal{A}'$ returns $rvk_{i\leftarrow j} = \beta_j/\beta_i$ to $\mathcal{A}$.

– *Forgery*: When $\mathcal{A}$ output a forgery (t, v, m^*, σ^*), if m^* is not the input of the j-th hash query, $\mathcal{A}'$ aborts the game; otherwise, $\mathcal{A}'$ outputs $(\sigma^*)^{\alpha_t^{-1}\beta_v^{-1}}$ as the answer for the BDH problem.

□

Analysis If m^* has never appeared in any hash query, then the probability that σ^* is valid is only $1/2^k$. Therefore, with overwhelming probability, m^* must appear in one of the hash queries. Since $\mathcal{A}'$ randomly selects $j \in [1, q_H]$, the probability that m^* appears in the j-th hash query is $1/q_H$. Under the condition that $\mathcal{A}'$ guesses the index j correctly, the game simulated by $\mathcal{A}'$ is perfect, and if $\mathcal{A}$ can produce a valid forgery $\sigma^* = e(g^b, g^{c\beta_v})^{a\alpha_t}$, then $\mathcal{A}'$ can solve the BDH problem since $(\sigma^*)^{\alpha_t^{-1}\beta_v^{-1}} = e(g, g)^{abc}$.

Thus, if $\mathcal{A}$ can win the unforgeability game with probability ϵ, then $\mathcal{A}'$ can solve the BDH problem with probability ϵ/q_H.

Theorem 2 *The proposed DVPRS scheme is invisible under the Decisional Bilinear Diffie–Hellman assumption.*

Proof If there exists an adversary $\mathcal{D}$ which is able to distinguish a real signature from a random signature in the invisibility game, we can construct another algorithm $\mathcal{C}$ who can solve the Decisional Bilinear Diffie–Hellman problem with a non-negligible advantage. □

$\mathcal{C}$ is given a DBDH challenge (g, g^a, g^b, g^c, t) such that t is either $e(g, g)^{abc}$ or $e(g, g)^z$, where a, b, c, z are random elements of $\mathbb{Z}_q$. $\mathcal{C}$ simulates the game for $\mathcal{D}$ as follows.

– *Setup*: The simulator first randomly chooses two indices $1 \le i, j \le n$ where $i \ne j$. $\mathcal{C}$ then sets the public key of user i as $pk_i = (g^a, g^{y_i})$ where y_i is randomly chosen from $\mathbb{Z}_q$. $\mathcal{C}$ also sets the public key of user j as $pk_j = (g^{x_j}, g^b)$ where x_j is randomly chosen from $\mathbb{Z}_q$. For all the other users $t \ne i, j$, $\mathcal{C}$ randomly chooses x_t, y_t from $\mathbb{Z}_q$ and sets $pk_t = (g^{x_t}, g^{y_t})$. $\mathcal{C}$ gives all the user public keys to $\mathcal{D}$. Let q_H denote the maximum number of hash queries that $\mathcal{D}$ will make in the game. $\mathcal{C}$ randomly selects an index $k \in [1, q_H]$, and maintains a hash table which records the input and output of each hash query. $\mathcal{C}$ guesses that the adversary will output a challenge with signer i, designated verifier j, and a message that is the input to the k-th hash query. Since i, j, k are randomly chosen, the chance that $\mathcal{C}$ guesses the challenge correctly is at least $1/n(n-1)q_H$. For simplicity, below we assume that $\mathcal{C}$'s guess is correct.
  $\mathcal{C}$ answers all the oracle queries in the Training Phase 1 as follows.
– *Hash queries*: For each hash query made by $\mathcal{D}$, $\mathcal{C}$ first checks the hash table to see if an entry for the same input exists in the hash table. If so, the same hash output will be returned to $\mathcal{D}$. Otherwise, if it is the k-th hash query, let m^* denote the hash input provided by $\mathcal{D}$, $\mathcal{C}$ sets the hash output as $H(m^*) = g^c$. Otherwise, $\mathcal{C}$ selects a random $r \in \mathbb{Z}_q$ and returns g^r as the hash value of the message to $\mathcal{D}$.
– *Signing queries*: For each signing query with input (i', j', m'), if $\mathcal{C}$ guesses the challenge correctly, then $(i', j', m') \ne (i, j, m^*)$, and $\mathcal{C}$ can simulate the signature using the bilinear map as follows:
  – if $(i', j') = (i, j)$ and $m' \ne m^*$, then $\mathcal{C}$ can simulate the signature as $\sigma = e(g^a, g^b)^r$ where $H(m') = g^r$ according to the simulation of the hash queries;
  – if $m' = m^*$ and $i' \ne i, j' = j$, or $i' = i, j' \ne j$, then $\mathcal{C}$ can simulate the signature as $\sigma = e(g^c, g^b)^{x_{i'}}$ if $i' \ne i, j' = j$, or $\sigma = e(g^c, g^a)^{y_{j'}}$ if $i' = i, j \ne j'$;

  - for cases, $\mathcal{C}$ can answer it easily by using the secret keys.

- *Verification queries*: If the adversary makes a verification query with input (i', j', m', σ') such that $(i', j', m') \neq (i, j, m^*)$, then $\mathcal{C}$ simulates a signing query for (i', j', m') to obtain σ. If $\sigma' = \sigma$, return '1'; otherwise, return '0'. If $(i', j', m') = (i, j, m^*)$, then $\mathcal{C}$ directly returns '0'. Since our DVPRS is unforgeable, we can see that the simulation for the verification oracle is indistinguishable from a real simulation.
- *ReSign* and *ReDV queries*: Since the adversary is not allowed to make a ReSign or ReDV query such that the output is a signature with regards to (i, j, m^*), and our DVPRS is deterministic, $\mathcal{C}$ can answer the ReSign and ReDV queries by simulating a signing query to generate the answer for the ReSign or ReDV query.

Challenge: If the adversary does not output (i, j, m^*) as the challenge, then $\mathcal{C}$ aborts the game. Otherwise, $\mathcal{C}$ sets $\sigma^* = t$, and returns it to $\mathcal{D}$.

$\mathcal{C}$ simulates the Training Phase 2 same as in Training Phase 1.

Finally, $\mathcal{C}$ outputs whatever $\mathcal{D}$ outputs.

Analysis If $\mathcal{C}$ guesses the indices (i, j, k) correctly, then the simulation is perfect.

- If $t = e(g, g)^{abc}$, then $\sigma^* = e(g, g)^{abc}$ is a valid signature on the message m^*.
- If $t = e(g, g)^z$, then $\sigma^* = e(g, g)^z$ is a random element in the signature space.

Therefore, if $\mathcal{D}$ has a non-negligible advantage ϵ to guess correctly in the invisibility game, $\mathcal{C}$ can solve the DBDH problem with advantage at least $\epsilon' \geq \epsilon / n^2 q_H$.

Theorem 3 *The proposed DVPRS scheme has perfect transparency.*

The proof is straightforward since our Sign, ReSign and ReDV algorithms are all deterministic. That is,

$$\sigma_{BC} = Sign(sk_B, pk_C, m) \tag{1}$$

$$\sigma'_{BC} = ReSign(rsk_{A \to B}, Sign(sk_A, pk_C, m), m) \tag{2}$$

(1) and (2) are identical, and

$$\sigma_{AD} = Sign(sk_A, pk_D, m) \tag{3}$$

$$\sigma'_{AD} = ReDV(rvk_{C \to D}, Sign(sk_A, pk_C, m), m) \tag{4}$$

(3) and (4) are also identical.

5 Conclusion

We introduced a new cryptographic primitive named Designated Verifier Proxy Re-Signature (DVPRS) in this paper. We defined the formal definition and security model for this new primitive, and proposed an efficient construction that is proven secure in the random oracle model. We also showed that DVPRS is very useful in network and communication applications. Our future work on this topic includes: (1) a practical DVPRS scheme that can be proven secure without random oracles; and (2) the construction of a practical unidirectional DVPRS scheme.

Acknowledgments This work was supported by National Natural Science Foundation of China under Grant 61702268.

References

1. Ateniese, G., & Hohenberger, S. (2005). Proxy re-signatures: new definitions, algorithms, and applications. In *Proceedings of the 12th ACM conference on Computer and communications security*, pp. 310–319. ACM.
2. Bellare, M., & Rogaway, P. (1993). Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pp. 62–73. ACM.
3. Blaze, M., Bleumer, G., & Strauss, M. (1998). Divertible protocols and atomic proxy cryptography. In *Advances in Cryptology - EUROCRYPT'98*, pp. 127–144. Springer.
4. Boneh, D., Lynn, B., & Shacham, H. (2001). Short signatures from the weil pairing. In *Advances in Cryptology - ASIACRYPT 2001*, pp. 514–532. Springer.
5. Chow, S. S., & Phan, R. C. W. (2008). Proxy re-signatures in the standard model. In *Information Security Conference*, pp. 260–276. Springer
6. Galbraith, S.D., & Mao, W. (2003). Invisibility and anonymity of undeniable and confirmer signatures. In *Topics in Cryptology - CT-RSA 2003*, pp. 80–97. Springer.
7. Huang, Q., Yang, G., Wong, D. S., & Susilo, W. (2011). Efficient strong designated verifier signature schemes without random oracle or with non-delegatability. *International Journal of Information Security*, *10*(6), 373–385.
8. Huang, Q., Yang, G., Wong, D. S., & Susilo, W. (2011). Identity-based strong designated verifier signature revisited. *Journal of Systems and Software*, *84*(1), 120–129.
9. Huang, X., Susilo, W., Mu, Y., & Wu, W. (2008). Secure universal designated verifier signature without random oracles. *International Journal of Information Security*, *7*(3), 171–183.
10. Huang, X., Susilo, W., Mu, Y., & Zhang, F. (2006). Short (identity-based) strong designated verifier signature schemes. In *Information Security Practice and Experience, Second International Conference, ISPEC 2006, Hangzhou, China, April 11-14, 2006, Proceedings, Lecture Notes in Computer Science*, vol. 3903, pp. 214–225. Springer.
11. Huang, X., Susilo, W., Mu, Y., & Zhang, F. (2008). Short designated verifier signature scheme and its identity-based variant. *International Journal of Network Security*, *6*(1), 82–93.
12. Jakobsson, M., Sako, K., & Impagliazzo, R. (1996). Designated verifier proofs and their applications. In *Advances in Cryptology - EUROCRYPT*, pp. 143–154. Springer.
13. Kang, B., Boyd, C., & Dawson, E. (2009). Identity-based strong designated verifier signature schemes: attacks and new construction. *Computers & Electrical Engineering*, *35*(1), 49–53.
14. Laguillaumie, F., & Vergnaud, D. (2004). Designated verifier signatures: anonymity and efficient construction from any bilinear map. In *4th International Conference on Security in Communication Networks,* pp. 105–119. Springer.
15. Laguillaumie, F., & Vergnaud, D. (2004). Multi-designated verifiers signatures. In *International Conference on Information and Communications Security*, pp. 495–507.
16. Libert, B., & Vergnaud, D. (2008). Multi-use unidirectional proxy re-signatures. In *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 511–520. ACM.
17. Shao, J., Cao, Z., Wang, L., & Liang, X. (2007). Proxy re-signature schemes without random oracles. In *Progress in Cryptology - INDOCRYPT 2007*, pp. 197–209. Springer.
18. Shao, J., Feng, M., Zhu, B., Cao, Z., & Liu, P. (2010). The security model of unidirectional proxy re-signature with private re-signature key. In *Australasian Conference on Information Security and Privacy*, pp. 216–232. Springer.
19. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J. (2003). Universal designated-verifier signatures. In *Advances in Cryptology - Asiacrypt 2003*, pp. 523–542. Springer.
20. Susilo, W., Zhang, F., & Mu, Y. (2004). Identity-based strong designated verifier signature schemes. In *Australasian Conference on Information Security and Privacy*, pp. 313–324. Springer.
21. Vergnaud, D. (2006). New extensions of pairing-based signatures into universal designated verifier signatures. In *ICALP*, pp. 58–69.
22. Waters, B. (2005). Efficient identity-based encryption without random oracles. In: *Advances in Cryptology - EUROCRYPT*, pp. 114–127.

23. Zhang, R., Furukawa, J., & Imai, H. (2005). Short signature and universal designated verifier signature without random oracles. In *Applied Cryptography and Network Security*, pp. 483–498.

Jiannan Wei received the M.S. degree from Zhengzhou University, China, in 2012, and Ph.D. degree in School of Computing and Information Technology, University of Wollongong, Australia, in 2016. She is currently an assistant professor at the School of Computer Science and Engineering, Nanjing University of Science and Technology. Her major research interests include public key cryptography, privacy-preserving digital signatures, wireless network security.

Guomin Yang graduated with a Ph.D. in Computer Science from the City University of Hong Kong in 2009. He worked as a Research Scientist at the Temasek Laboratories of the National University of Singapore (NUS) from Sep 2009 to May 2012. He is currently a Senior Lecturer and ARC DECRA Fellow at the School of Computing and Information Technology, University of Wollongong. His research mainly focuses on Applied Cryptography and Network Security.

Yi Mu received his Ph.D. from the Australian National University in 1994. He currently is professor and the co-director of Centre for Computer and Information Security Research at University of Wollongong, Australia. His current research interests include information security and cryptography. Yi Mu is the editor-in-chief of International Journal of Applied Cryptography and serves as associate editor for ten other international journals. He is a senior member of the IEEE and a member of the IACR.