Computing and Informatics, Vol. 36, 2015, 1210–1230

AN EFFICIENT CERTIFICATE-BASED DESIGNATED VERIFIER SIGNATURE SCHEME

Jiguo Li, Na Qian, Yichen Zhang

College of Computer and Information
Hohai University
Nanjing, China, 211100
e-mail: {lijiguo, qiannan, zhangyc}@hhu.edu.cn

Xinyi Huang

Fujian Provincial Key Laboratory of Network Security and Cryptology
School of Mathematics and Computer Science
Fujian Normal University
Fuzhou, Fujian, China, 350117
e-mail: xyhuang@fjnu.edu.cn

Abstract. Certificate-based public key cryptography not only solves certificate revocation problem in traditional PKI but also overcomes key escrow problem inherent in identity-based cryptosystems. This new primitive has become an attractive cryptographic paradigm. In this paper, we propose the notion and the security model of certificate-based designated verifier signatures (CBDVS). We provide the first construction of CBDVS and prove that our scheme is existentially unforgeable against adaptive chosen message attacks in the random oracle model. Our scheme only needs two pairing operations, and the signature is only one element in the bilinear group $\mathbb{G}_1$. To the best of our knowledge, our scheme enjoys *shortest* signature length with less operation cost.

Keywords: Public key cryptography, digital signature, certificate-based signature, designated verifier signature, random oracle model

Mathematics Subject Classification 2010: 94A60

1 INTRODUCTION

In traditional digital signatures, a signer with a secret key signs a message and anyone with access to the corresponding public key can verify the validity of the message. A signature verifier can convince any third party about this fact by presenting a digital signature on a message. Sometimes we do not want to present publicly verifiable signatures to other parties, such as certificates for hospital records, income summary, etc. in the real world. We only wish the designated verifier to verify the authentication of the message. In order to solve the above problem, Jackobsson, Sako and Impagliazzo [1] proposed the concept of designated verifier signatures (DVS). In a DVS scheme, the signer can designate a single party, i.e. the designated verifier, and only this designated verifier can be convinced about the validity of the signatures. DVS can be viewed as a tradeoff between ordinary signatures (where signatures are publicly verifiable) and undeniable signatures (where the signer has the full control of signatures). Lipmaa et al. [2] showed that the signer could abuse the disavowal protocol in [1]. In addition, they gave a rigorous formalization of the security for designated-verifier signature schemes, and proposed an efficient designated-verifier signature scheme that was provably unforgeable under a tight reduction to the decisional Diffie-Hellman problem in the random oracle model. Kudla and Paterson [3] showed that non-interactive designated verifier (NIDV) proofs in [1] may have applications outside of the context of undeniable signatures and presented two security models, one for general NIDV proof systems, and the other for NIDV undeniable signatures. They also presented the full domain hash variant of the undeniable signature scheme of Chaum [4] with NIDV confirmation and denial proofs. In order to improve the privacy of users in certification systems, Steinfeld et al. [5] introduced a special type of digital signature scheme called a universal designated-verifier signature (UDVS) scheme. They defined precise security notions for UDVS schemes, proposed an efficient deterministic UDVS scheme based on bilinear pairs, and proved that their scheme was secure in the random oracle model under the hardness of the bilinear Diffie Hellman problem. Furthermore, Steinfeld et al. [6] showed how to efficiently extend the standard Schnorr and RSA signature schemes into UDVS schemes.

DVS was originally introduced under traditional public key cryptosystem (PKC), where the public key of the user is a random string. The central problem in traditional PKC is to prove that a public key is genuine and authentic, and has not been tampered by malicious users. The usual approach is to use a certificate to bind the identity of an entity, its public key and other information. Certificates are generated by a trusted certificate authority (CA), which can be viewed as CA's signature. However, traditional PKC is generally considered to be costly to use and manage. The first construction of identity-based DVS scheme was proposed in [7], whose purpose is to solve the above mentioned problem in traditional PKC. Kang et al. [8] presented a novel identity-based strong designated verifier signature scheme, which has short size of signature, low communication and computational cost. However, in identity-based public key cryptosystem (ID-PKC), the trusted authority, known

as the private key generator (PKG), can generate private keys of all users, and thus private key escrow becomes an inherent problem in ID-PKC. Moreover, private key must be sent over secure channel, which makes private key distribution difficult. Certificateless public key cryptosystem (CL-PKC) [9, 10] was proposed to combine the merits of traditional PKC and ID-PKC. Huang et al. [11] proposed the first notion and construction of DVS in CL-PKC. In 2008, Chen et al. [12] also proposed an efficient certificateless short designated verifier signature scheme. Recently, Yang et al. [13] formalized the notion and the security model for certificateless strong designated verifier signature scheme (CLSDVS) and then presented an efficient CLSDVS scheme. However, CL-PKC also needs a secure channel to transmit partial private keys. In order to solve the above problem, Gentry et al. [14] introduced a new paradigm called certificate-based public key cryptography (CB-PKC) in Eurocrypt 2003. Like traditional PKC, CB-PKC also has a third party (called CA) to generate certificates. But the difference is that the certificate is also necessary for decryption in CB-PKC. A correct decryption needs both the private key and an up-to-date certificate from the CA. Therefore, it overcomes the drawbacks in traditional PKC and ID-PKC. On the other hand, there is no need to keep certificate secret since certificate is used to prove the authenticity and uniqueness of the public key.

In CT-RSA '04, Kang and Park [15] proposed the first certificate-based signature (CBS) scheme, and proved that the scheme was secure in the random oracle model. Li et al. [16] first introduced key replacement attack into CBS and refined the security model of CBS. They showed that one of the CBS schemes in [15] was insecure under the key replacement attack. Furthermore, they constructed a new certificate-based signature scheme, which was existentially unforgeable against adaptive chosen message attacks under the computational Diffie-Hellman assumption in the random oracle model. Compared with the other secure CBS scheme, this scheme enjoys shorter signature length and less operation cost. Au et al. [17] proposed a new notion called certificate-based ring signature, together with a formal security model and a concrete implementation. In 2008, Liu et al. [18] proposed two new certificate-based signature schemes with new features and advantages. The first one was very efficient as it does not require any pairing computation and its security can be proven under discrete logarithm assumption in the random oracle model. Their second scheme can be proven secure without random oracles. Zhang [19] pointed out that their scheme without pairing [18] is insecure and discussed the flaws in their security proof. To overcome the flaws, an improved scheme was proposed with formal security analysis in the random oracle model. Li et al. [20] proposed a provably secure certificate-based proxy signature scheme, and showed their scheme was secure in the random oracle model. Ming and Wang [21] proposed an efficient CBS scheme. Wu et al. [22] introduced a new security model of certificate-based signatures. The model was not only more elaborated when compared with the existing ones, but also defined several new types of adversaries in CBS. Then, they investigated the relationship between certificate-based signatures and certificateless signatures by proposing a generic construction of certificate-based signatures from certificateless signatures. Li et al. [23] presented two new certificate-based signature

schemes secure against key replacement attacks. Their first scheme was existentially unforgeable against adaptive chosen message attacks under the computational Diffie-Hellman assumption in the random oracle model. Compared with the certificate-based signature scheme [15], their scheme enjoyed shorter signature length and less operation cost. Their second scheme was the first construction of certificate-based signature secure against key replacement attacks in the standard model. Recently, Li et al. [24] presented a short certificate-based signature scheme, which requires only one group element in the signature length and one pairing operation in signature generation and verification. Furthermore, they [25] presented a provably secure certificate-based signature scheme without pairings. The proposed schemes have strong applicability in situations with limited bandwidth and power-constrained devices. In addition, they [26] proposed a new certificate-based signcryption scheme with enhanced security features, which performs signature and encryption in a single logical step. In order to solve the key exposure problem, Li et al. proposed forward-secure CBS schemes [27, 28] and certificate-based key-insulated signature schemes [29, 30].

Certificate-based cryptography is a new kind of public key cryptography, which combines the merits of traditional public key infrastructure (PKI) and identity-based cryptography. It does not have the inherent key escrow problem in the identity-based cryptography, and eliminates the certificate revocation problem and third-party queries in the traditional PKI [14]. Certificate-based cryptography could be used to construct an efficient PKI [31] and have attracted a lot of attention since it was proposed. To date, there is no concrete construction of certificate-based designated verifier signature. This paper is aimed at constructing an efficient certificate-based designated verifier signature scheme.

Our Motivation and Contribution. As discussed above, DVS was originally introduced under traditional public key cryptosystem, where the public key of the user is a random string. The central problem in traditional PKC is to prove that a public key is genuine and authentic, and has not been tampered by malicious users. The usual approach is to use a certificate to bind the identity of an entity, its public key and other information. Certificates are generated by CA, which can be viewed as CA's signature. However, traditional PKC is generally considered to be costly to use and manage. Although identity-based DVS scheme overcomes the certificate revocation problem and third-party queries in traditional PKC, it suffers from key escrow problem. In order to solve the above problem, we introduce the notion of certificate-based designated verifier signatures (CBDVS) and propose the first security model of CBDVS. We then present the first construction of CBDVS. Our scheme is existentially unforgeable against adaptive chosen message attacks in the random oracle model.

Organization of the Paper. The rest of the paper is organized as follows. Section 2 gives some definitions that are required in the paper. We present definition and adversarial model of CBDVS in Section 3. In Section 4, we propose an efficient CBDVS scheme. We provide the security proof in Section 5. In Sec-

tion 6, we analyze the performances of the proposed CBDVS scheme. Finally, we conclude the paper in Section 7.

2 PRELIMINARIES

This section reviews the notation of bilinear mapping and some complexity problems that are required to understand the following sections.

Let $\mathbb{G}_1$ denote an additive group of order q and $\mathbb{G}_2$ be a multiplicative group of the same order. Let P denote a generator of $\mathbb{G}_1$. $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is called a bilinear mapping if it satisfies the following properties:

- Bilinear: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1, a, b \in \mathbb{Z}_q$;
- Non-degeneracy: There exist $P, Q \in \mathbb{G}_1$ such that $e(P, Q) \neq 1$;
- Computable: There exists an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

Definition 1 (CDHP (Computational Diffie-Hellman Problem))**.** Given a randomly chosen $P \in \mathbb{G}_1$, as well as $aP, bP \in \mathbb{G}_1$ (for unknown randomly chosen $a, b \in \mathbb{Z}_q^*$), compute abP.

Definition 2 (CBDHP (Computational Bilinear Diffie-Hellman Problem))**.** Given a randomly chosen $P \in \mathbb{G}_1$, as well as $aP, bP, cP \in \mathbb{G}_1$ (for unknown randomly chosen $a, b, c \in \mathbb{Z}_q^*$), compute $e(P, P)^{abc} \in \mathbb{G}_2$.

Definition 3 (DBDHP (Decisional Bilinear Diffie-Hellman Problem))**.** Given a randomly chosen $P \in \mathbb{G}_1$, as well as $aP, bP, cP \in \mathbb{G}_1$ (for unknown randomly chosen $a, b, c \in \mathbb{Z}_q^*$) and $h \in \mathbb{G}_2$, decide whether $h = e(P, P)^{abc}$.

Definition 4 (GBDHP (Gap Bilinear Diffie-Hellman Problem))**.** Given a randomly chosen $P \in \mathbb{G}_1$, as well as $aP, bP, cP \in \mathbb{G}_1$ (for unknown randomly chosen $a, b, c \in \mathbb{Z}_q^*$), compute $e(P, P)^{abc} \in \mathbb{G}_2$ with the help of the DBDH oracle.

3 DEFINITION AND ADVERSARIAL MODEL OF CBDVS

3.1 Definition

There are three parties in a CBDVS scheme: signer ID_A, designated verifier ID_B, and certificate authority (CA). A CBDVS scheme is defined by six algorithms: Setup, UserKeyGen, CertGen, Sign, Verify, Transcript Simulation.

Setup: This algorithm takes the security parameter l as input and returns system parameter *param*. CA selects the master secret key *msk* and computes the master public key *mpk*. CA publishes *param*, *mpk*, but keeps *msk* secret.

UserKeyGen: This algorithm takes *param*, *mpk* as input and returns the user's key pair (S_{ID_i}, P_{ID_i}).

CertGen: This algorithm takes *param*, *mpk*, the user's identity ID_i, the public key P_{ID_i} of ID_i and the master secret key *msk* as input, and returns the user's certificate $cert_{ID_i}$.

Sign: This algorithm takes *param*, *mpk*, a message m, the certificate $cert_{ID_A}$, the private key S_{ID_A} of ID_A and the public key P_{ID_B} of ID_B as input, and returns a signature σ.

Verify: This algorithm takes a message/signature pair (m, σ), S_{ID_B}, $cert_{ID_B}$, P_{ID_A} and *param*, *mpk* as input, and outputs $d \in (acc, rej)$.

Transcript Simulation: This algorithm takes $param, mpk, m, cert_{ID_B}, S_{ID_B}$ and P_{ID_A} as input, and returns a transcript σ.

3.2 Adversarial Model of CBDVS

There are two types of adversaries with different capabilities in CBDVS system:

Type I adversary: A type I adversary $\mathcal{A}_{\mathcal{I}}$ has the ability to replace the public key of any entity with a value of his choice. However, he does not have access to the master secret key and the certificates of the replaced public key. This type of adversary also can deceive anyone into using the replaced public key to verify signatures.

Type II adversary: Type II adversary $\mathcal{A}_{\mathcal{II}}$ simulates a malicious CA. This type of adversary has access to the master secret key and can generate certificates, but cannot perform public key replacement.

We use two games to describe the security of CBDVS as follows.

3.2.1 CBDVS Game 1

The security of CBDVS against a type I adversary $\mathcal{A}_{\mathcal{I}}$ is defined by the game described below:

1. Setup: The challenger runs the **Setup** algorithm to obtain the system parameter *param* and master secret/public key pair (*msk*, *mpk*), publishes *param* and *mpk*.
2. UserCreate: The challenger runs the **UserKeyGen** algorithm to generate the user's public/private key (S_{ID_i}, P_{ID_i}), $i \in \{1, 2, \ldots\}$ for the user ID_i, and returns (S_{ID_i}, P_{ID_i}) to adversary $\mathcal{A}_{\mathcal{I}}$.
3. PublicKey Replace: Adversary $\mathcal{A}_{\mathcal{I}}$ randomly chooses P'_{ID_i} in the public key space and replaces the public key of any user ID_i with P'_{ID_i}.
4. CertGen Queries: When adversary $\mathcal{A}_{\mathcal{I}}$ makes a certificate generation query (ID_i, P_{ID_i}), the challenger runs the **CertGen** algorithm to generate a certificate $cert_{ID_i}$ and returns $cert_{ID_i}$ to $\mathcal{A}_{\mathcal{I}}$.
5. Sign Queries: When adversary $\mathcal{A}_{\mathcal{I}}$ makes a sign query $(m, ID_i, ID_j, P_{ID_i}, P_{ID_j})$, the challenger runs the **Sign** algorithm to generate a signature σ, and returns σ to $\mathcal{A}_{\mathcal{I}}$.

6. Verify Queries: When adversary $\mathcal{A}_{\mathcal{I}}$ makes a verify query $(m, ID_i, ID_j, \sigma, P_{ID_i}, P_{ID_j})$, the challenger runs the **Verify** algorithm to decide whether $(m, \sigma, ID_i, ID_j, P_{ID_i}, P_{ID_j})$ is valid or not, and returns $d \in (acc, rej)$ to $\mathcal{A}_{\mathcal{I}}$.
7. Output: After all queries, $\mathcal{A}_{\mathcal{I}}$ outputs a forgery $(m^*, ID_A, ID_B, \sigma^*, P^*_{ID_A}, P^*_{ID_B})$. We say $\mathcal{A}_{\mathcal{I}}$ wins the game if the forgery satisfies the following requirements:

   (a) σ^* is a valid signature on the message m^* with the signer's identity ID_A, the designated verifier's identity ID_B , and their public keys $(P^*_{ID_A}, P^*_{ID_B})$. Here, $P^*_{ID_A}$ and $P^*_{ID_B}$ are chosen by $\mathcal{A}_{\mathcal{I}}$ and might not be those returned from the oracle **UserCreate**.
   (b) $(m^*, ID_A, ID_B, P^*_{ID_A}, P^*_{ID_B})$ has never been submitted as one of **Sign** queries.
   (c) ID_A and ID_B have never been submitted as one of **CertGen** queries.

A CBDVS scheme is existentially unforgeable against a type I adversary if no polynomial-time adversary can win the above game with a non-negligible probability.

3.2.2 CBDVS Game 2

The security of CBDVS against a type II adversary $\mathcal{A}_{\mathcal{II}}$ is defined by the game described below:

1. Setup: The challenger runs the **Setup** algorithm to obtain the system parameter *param* and master secret/public key pair (*msk*, *mpk*), and sends *param*, *msk* and *mpk* to $\mathcal{A}_{\mathcal{II}}$.
2. UserCreate: The challenger runs the **UserKeyGen** algorithm to generate the user's public/private key (S_{ID_i}, P_{ID_i}), $i \in \{1, 2, \ldots\}$ for the user ID_i, and returns P_{ID_i} to adversary $\mathcal{A}_{\mathcal{II}}$.
3. Corruption: When adversary $\mathcal{A}_{\mathcal{II}}$ makes a corruption query ID_i, the challenger responds with the private key S_{ID_i}.
4. Sign Queries: When adversary $\mathcal{A}_{\mathcal{II}}$ makes a sign query (m, ID_i, ID_j), the challenger runs the **Sign** algorithm to generate a signature σ, and returns σ to adversary $\mathcal{A}_{\mathcal{II}}$.
5. Verify Queries: When adversary $\mathcal{A}_{\mathcal{II}}$ makes a verify query (m, ID_i, ID_j, σ), the challenger runs the **Verify** algorithm to decide whether (m, ID_i, ID_j, σ) is valid or not, and returns $d \in (acc, rej)$ to adversary $\mathcal{A}_{\mathcal{II}}$.
6. Output: After all queries, adversary $\mathcal{A}_{\mathcal{II}}$ outputs a forgery $(m^*, ID_A, ID_B, \sigma^*)$. We say $\mathcal{A}_{\mathcal{II}}$ wins the game if the forgery satisfies the following requirements:

   (a) σ^* is a valid signature on the message m^* with the signer's identity ID_A, the designated verifier's identity ID_B and their public keys.
   (b) (m^*, ID_A, ID_B) has never been submitted as one of **Sign** queries.
   (c) ID_A and ID_B have never been submitted as one of **Corruption** queries.

A CBDVS scheme is existentially unforgeable against a type II adversary if no polynomial-time adversary can win the above game with a non-negligible probability.

4 OUR SCHEME

Setup: The CA executes the following steps:

Step 1: Takes the security parameter l as input and generates the system parameter $(\mathbb{G}_1, \mathbb{G}_2, q, e, P)$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are group of the same prime order q and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a bilinear mapping.

Step 2: Randomly selects master secret key $msk = s \in \mathbb{Z}_q^*$ and sets master public key $mpk = P_{pub} = sP$.

Step 3: Chooses two cryptographic hash functions $H_0 : \{0,1\}^* \times \mathbb{G}_1 \to \mathbb{G}_1$ and $H_1 : \{0,1\}^* \times \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_1$.

Step 4: Keeps the master key $msk = s$ secret and publishes $params = (\mathbb{G}_1, \mathbb{G}_2, q, e, P, H_0, H_1, P_{pub})$.

UserKeyGen: User ID_i randomly selects his/her private key S_{ID_i}, and calculates his/her public key $P_{ID_i} = S_{ID_i}P$. The user key pair is (S_{ID_i}, P_{ID_i}), $i \in \{A, B\}$.

CertGen: CA uses the system parameter and the master secret key msk to generate the user's certificate.

Step 1: The user ID_i sends CA a data string $data$ which contains the information of user's public key P_{ID_i} and identity ID_i.

Step 2: CA verifies user's information. If everything is correct, CA computes $Q_{ID_i} = H_0(ID_i, P_{ID_i}) \in \mathbb{G}_1$, $cert_{ID_i} = sQ_{ID_i}$, and sends $cert_{ID_i}$ to user ID_i,$i \in \{A, B\}$.

Sign: The signer ID_A performs the following steps to sign a message m for the designated verifier ID_B.

$$\begin{aligned} K_1 &= S_{ID_A} P_{ID_B}, \\ K_2 &= e(cert_{ID_A}, Q_{ID_B}), \\ \sigma &= H_1(m, K_1, K_2). \end{aligned}$$

Verify: The designated verifier ID_B performs the following steps to verify a signature σ on a message m from the signer ID_A.

$$\sigma \stackrel{?}{=} H_1(m, S_{ID_B} P_{ID_A}, e(cert_{ID_B}, Q_{ID_A}))$$

If the above equation holds, ID_B will accept σ as a valid signature. Otherwise, σ is an invalid signature.

Transcript Simulation: ID_B can generate the signature σ' intended for him by performing the following steps.

$$\begin{aligned} K_1' &= S_{ID_B} P_{ID_A}, \\ K_2' &= e(cert_{ID_B}, Q_{ID_A}), \\ \sigma' &= H_1(m, K_1', K_2'). \end{aligned}$$

Correctness: Signatures generated by **Sign** algorithm will always pass the **Verify** algorithm:

$$\begin{aligned} K_1 &= S_{ID_A} P_{ID_B} \\ &= S_{ID_A} S_{ID_B} P \\ &= S_{ID_B} S_{ID_A} P \\ &= S_{ID_B} P_{ID_A}. \end{aligned}$$

$$\begin{aligned} K_2 &= e(cert_{ID_A}, Q_{ID_B}) \\ &= e(sQ_{ID_A}, Q_{ID_B}) \\ &= e(sQ_{ID_B}, Q_{ID_A}) \\ &= e(cert_{ID_B}, Q_{ID_A}). \end{aligned}$$

Therefore: $\sigma = H_1(m, K_1, K_2) = H_1(m, S_{ID_B} P_{ID_A}, e(cert_{ID_B}, Q_{ID_A}))$.

Similarly, signatures generated by **Transcript Simulation** algorithm will always pass the **Verify** algorithm.

5 CBDVS SECURITY ANALYSIS

Theorem 1. If there is a type I adversary $\mathcal{A}_{\mathcal{I}}$ can forge a valid signature of the proposed scheme with success probability ε after making q_{H_0} queries to H_0 oracle, q_{H_1} queries to H_1 oracle, and q_S signing queries, then there exists an algorithm $\mathcal{B}$ who can use to solve an instance of the GBDH problem with probability at least $(1/q_{H_0})^2(1 - 1/(2^l - q_{H_1} - q_S))\varepsilon$. ($l$ is the security parameter of CBDVS scheme.)

Proof. Given a random instance $(P, P_1 = aP, P_2 = bP, P_3 = cP)$ of GBDH problem, we will show how $\mathcal{B}$ can use $\mathcal{A}_{\mathcal{I}}$ to obtain the value of $e(P, P)^{abc} \in \mathbb{G}_2$ with the help of the DBDH oracle. We assume that $\mathcal{A}_{\mathcal{I}}$ is well-behaved in the sense that $\mathcal{A}_{\mathcal{I}}$ will never repeat the same queries in the simulation.

Setup: In this game, $\mathcal{B}$ will set the system parameters. $\mathcal{B}$ starts by setting $P_{pub} = cP$ and returns $\{\mathbb{G}_1, \mathbb{G}_2, q, e, P, P_{pub}\}$ to $\mathcal{A}_{\mathcal{I}}$.

UserCreate: $\mathcal{B}$ maintains a list L-list which consists of the tuples $(ID_i, s_{ID_i}, P_{ID_i}, a_i)$ described as follows. The list is initially empty. When $\mathcal{A}_{\mathcal{I}}$ queries the oracle with a request ID_i, $\mathcal{B}$ creates user ID_i. $\mathcal{B}$ randomly chooses $s_{ID_i} \in \mathbb{Z}_q^*$, and

computes $P_{ID_i} = s_{ID_i}P$. Then $\mathcal{B}$ adds $(ID_i, s_{ID_i}, P_{ID_i}, 0)$ into L-list, and returns (s_{ID_i}, P_{ID_i}) to adversary $\mathcal{A}_{\mathcal{I}}$.

PublicKey Replace: $\mathcal{A}_{\mathcal{I}}$ randomly selects $P'_{ID_i} \in \mathbb{G}_1$ to replace the public key of the user ID_i, $\mathcal{B}$ will search L-list,

1. If there is an item $(ID_i, *, *, *)$ in L-list, $\mathcal{B}$ rewrites this tuples as $(ID_i, \perp, P'_{ID_i}, 1)$.
2. Otherwise, $\mathcal{B}$ adds $(ID_i, \perp, P'_{ID_i}, 1)$ into L-list.

H_0 **oracle:** $\mathcal{B}$ maintains a list H_0-list which consists of the tuples $(ID_i, P_{ID_i}, \alpha_{ID_i}, Q_{ID_i})$ described as follows. The list is initially empty. $\mathcal{B}$ first chooses two integers (u, v) from 1 to q_{H_0}. When $\mathcal{A}_{\mathcal{I}}$ queries the H_0 oracle with a request (ID_i, P_{ID_i}),

1. If $i = u$, $\mathcal{B}$ randomly selects $\alpha_{ID_i} \in \mathbb{Z}_q^*$ and computes $Q_{ID_i} = \alpha_{ID_i}aP$. Then $\mathcal{B}$ adds $(ID_i, P_{ID_i}, \alpha_{ID_i}, Q_{ID_i})$ into H_0-list and returns Q_{ID_i} to $\mathcal{A}_{\mathcal{I}}$ as the answer.
2. Else if $i = v$, $\mathcal{B}$ randomly selects $\alpha_{ID_i} \in \mathbb{Z}_q^*$ and computes $Q_{ID_i} = \alpha_{ID_i}bP$. Then $\mathcal{B}$ adds $(ID_i, P_{ID_i}, \alpha_{ID_i}, Q_{ID_i})$ into H_0-list and returns Q_{ID_i} to $\mathcal{A}_{\mathcal{I}}$ as the answer.
3. Otherwise, $\mathcal{B}$ randomly selects $\alpha_{ID_i} \in \mathbb{Z}_q^*$ and computes $Q_{ID_i} = \alpha_{ID_i}P$. Then $\mathcal{B}$ adds $(ID_i, P_{ID_i}, \alpha_{ID_i}, Q_{ID_i})$ into H_0-list and returns Q_{ID_i} to $\mathcal{A}_{\mathcal{I}}$ as the answer.

CertGen Queries: $\mathcal{A}_{\mathcal{I}}$ requests the certificate of (ID_i, P_{ID_i}), $\mathcal{B}$ search H_0-list.

1. If $i = u$ or $i = v$, $\mathcal{B}$ aborts the game and the simulation fails.
2. Else, $\mathcal{B}$ computes $cert_{ID_i} = \alpha_{ID_i}P_{pub}$, and returns $cert_{ID_i}$ to adversary $\mathcal{A}_{\mathcal{I}}$.

H_1 **oracle:** Algorithm $\mathcal{B}$ maintains a list H_1-list which consists of the tuples $(m_i, K_{1i}, K_{2i}, ID_m, ID_n, \sigma_i, P^i_{ID_m}, P^i_{ID_n})$ explained as below ($P^i_{ID_m}$ and $P^i_{ID_n}$ are the public keys of ID_m and ID_n, respectively, in the i^{th} times sign query). The list is initially empty. When $\mathcal{A}_{\mathcal{I}}$ makes a request (m_i, K_{1i}, K_{2i}) to the H_1 oracle, $\mathcal{B}$ will search H_1-list.

1. If there is an item $(m_j, K_{1j}, K_{2j}, *, *, \sigma_j, *, *)$ in H_1-list such that $m_i = m_j$, $(K_{1i}, K_{2i}) \neq (K_{1j}, K_{2j})$. Algorithm $\mathcal{B}$ randomly selects $\sigma_i \in \mathbb{G}_1$ such that there is no item $(*, *, *, *, *, \sigma_i, *, *)$ in the H_1-list. Then $\mathcal{B}$ adds $(m_i, K_{1i}, K_{2i}, \perp, \perp, \sigma_i, \perp, \perp)$ into the H_1-list, and returns σ_i to adversary $\mathcal{A}_{\mathcal{I}}$.
2. If there is an item $(m_j, \perp, \perp, ID_m, ID_n, \sigma_j, P^j_{ID_m}, P^j_{ID_n})$ in H_1-list such that $m_i = m_j$. $\mathcal{B}$ first tests whether $e(P^j_{ID_m}, P^j_{ID_n}) \stackrel{?}{=} e(P, K_{1i})$ and whether $(Q^j_{ID_m}, Q^j_{ID_n}, P_{pub}, K_{2i})$ is a BDH tuple. $Q^j_{ID_m}$ is determined by the signer ID_m's public key $P^j_{ID_m}$. $Q^j_{ID_n}$ is determined by the designated verifier ID_n's public key $P^j_{ID_n}$.
   (a) If the above two conditions hold, $\mathcal{B}$ rewrites this form as $(m_i, K_{1i}, K_{2i}, ID_m, ID_n, \sigma_j, P^j_{ID_m}, P^j_{ID_n})$, and returns σ_j to adversary $\mathcal{A}_{\mathcal{I}}$.

(b) Otherwise, $\mathcal{B}$ chooses a random $\sigma_i \in \mathbb{G}_1$ such that there is no item $(*,*,*,*,*,\sigma_i,*,*)$ in H_1-list. Then $\mathcal{B}$ adds $(m_i, K_{1i}, K_{2i}, \perp, \perp, \sigma_i, \perp, \perp)$ into H_1-list and returns σ_i to adversary $\mathcal{A}_{\mathcal{I}}$.

3. Otherwise, $\mathcal{B}$ chooses a random $\sigma_i \in \mathbb{G}_1$ such that there is no item $(*,*,*,*,*,\sigma_i,*,*)$ in H_1-list. Then $\mathcal{B}$ adds $(m_i, K_{1i}, K_{2i}, \perp, \perp, \sigma_i, \perp, \perp)$ into H_1-list and returns σ_i to adversary $\mathcal{A}_{\mathcal{I}}$.

Sign Oracle: In this game, $\mathcal{B}$ will simulate the sign algorithm. At any time $\mathcal{A}_{\mathcal{I}}$ can query the sign algorithm. After receiving $\mathcal{A}_{\mathcal{I}}$'s request $(m_i, ID_m, ID_n, P^i_{ID_m}, P^i_{ID_n})$, $\mathcal{B}$ checks the H_1-list:

1. If there is an item $(m_j, K_{1j}, K_{2j}, \perp, \perp, \sigma_j, \perp, \perp)$ in H_1-list such that $m_i = m_j$, $\mathcal{B}$ first tests whether $e(P^i_{ID_m}, P^i_{ID_n}) \stackrel{?}{=} e(P, K_{1j})$ and $(Q^i_{ID_m}, Q^i_{ID_n}, P_{pub}, K_{2j})$ is a BDH tuple. Suppose the above two conditions hold, $\mathcal{B}$ rewrites this form as $(m_i, K_{1j}, K_{2j}, ID_m, ID_n, \sigma_j, P^i_{ID_m}, P^i_{ID_n})$ and returns σ_j to adversary $\mathcal{A}_{\mathcal{I}}$.
2. If there is an item $(m_j, K_{1j}, K_{2j}, ID_p, ID_k, \sigma_j, P^j_{ID_p}, P^j_{ID_k})$ in H_1-list such that $m_i = m_j$ but $(ID_p, ID_k, P^j_{ID_p}, P^j_{ID_k}) \neq (ID_m, ID_n, P^i_{ID_m}, P^i_{ID_n})$, $\mathcal{B}$ first tests whether $e(P^i_{ID_m}, P^i_{ID_n}) \stackrel{?}{=} e(P, K_{1j})$ and whether $(Q^i_{ID_m}, Q^i_{ID_n}, P_{pub}, K_{2j})$ is a BDH tuple. Suppose the above two conditions hold, $\mathcal{B}$ rewrites this form as $(m_i, K_{1j}, K_{2j}, ID_m, ID_n, \sigma_j, P^i_{ID_m}, P^i_{ID_n})$ and returns σ_j to adversary $\mathcal{A}_{\mathcal{I}}$.
3. If there is an item $(m_j, \perp, \perp, ID_p, ID_k, \sigma_j, P^j_{ID_p}, P^j_{ID_k})$ in H_1-list such that $m_i = m_j$, but $(ID_p, ID_k, P^j_{ID_p}, P^j_{ID_k}) \neq (ID_m, ID_n, P^i_{ID_m}, P^i_{ID_n})$, $\mathcal{B}$ first tests whether $e(P^i_{ID_m}, P^i_{ID_n}) \stackrel{?}{=} e(P^j_{ID_p}, P^j_{ID_k})$ and whether $e(Q^i_{ID_m}, Q^i_{ID_n}) \stackrel{?}{=} e(Q^j_{ID_p}, Q^j_{ID_k})$. Suppose the above two conditions hold, $\mathcal{B}$ adds $(m_i, \perp, \perp, ID_m, ID_n, \sigma_j, P^i_{ID_m}, P^i_{ID_n})$ into H_1-list and returns σ_j to adversary $\mathcal{A}_{\mathcal{I}}$.
4. Otherwise, $\mathcal{B}$ chooses a random $\sigma_i \in \mathbb{G}_1$ such that there is no item $(*,*,*,*,*,\sigma_i,*,*)$ in H_1-list. Then $\mathcal{B}$ adds $(m_i, \perp, \perp, ID_m, ID_n, \sigma_i, P^i_{ID_m}, P^i_{ID_n})$ into H_1-list and returns σ_i to adversary $\mathcal{A}_{\mathcal{I}}$.

Verify Oracle: In this game, $\mathcal{B}$ will simulate the verify algorithm. At any time $\mathcal{A}_{\mathcal{I}}$ can query the verify algorithm. After receiving $\mathcal{A}_{\mathcal{I}}$'s request $(m_i, ID_m, ID_n, \sigma_i, P^i_{ID_m}, P^i_{ID_n})$, $\mathcal{B}$ checks the H_1-list:

1. If there is an item $(m_i, *, *, ID_m, ID_n, \sigma_i, P^i_{ID_m}, P^i_{ID_n})$ in H_1-list, $\mathcal{B}$ will accept it as a valid signature.
2. If there is an item $(m_j, K_{1j}, K_{2j}, ID_p, ID_k, \sigma_j, P^j_{ID_p}, P^j_{ID_k})$ in H_1-list such that $m_i = m_j$, $\sigma_i = \sigma_j$ and $(ID_p, ID_k, P^j_{ID_p}, P^j_{ID_k}) \neq (ID_m, ID_n, P^i_{ID_m}, P^i_{ID_n})$, $\mathcal{B}$ verifies whether $e(P^i_{ID_m}, P^i_{ID_n}) \stackrel{?}{=} e(P, K_{1j})$ and $(Q^i_{ID_m}, Q^i_{ID_n}, P_{pub}, K_{2j})$ is a BDH tuple. If the above two conditions hold, $\mathcal{B}$ adds $(m_i, K_{1j}, K_{2j}, ID_m, ID_n, \sigma_i, P^i_{ID_m}, P^i_{ID_n})$ into H_1-list and accepts it as a valid signature.
3. If there is an item $(m_j, K_{1j}, K_{2j}, \perp, \perp, \sigma_j, \perp, \perp)$ in H_1-list such that $m_i = m_j$, $\sigma_i = \sigma_j$, $\mathcal{B}$ verifies whether $e(P^i_{ID_m}, P^i_{ID_n}) \stackrel{?}{=} e(P, K_{1j})$ and whether

$(Q^i_{ID_m}, Q^i_{ID_n}, P_{pub}, K_{2j})$ is a BDH tuple. If the above two conditions hold, $\mathcal{B}$ rewrites this form as $(m_i, K_{1j}, K_{2j}, ID_m, ID_n, \sigma_j, P^i_{ID_m}, P^i_{ID_n})$ and accepts it as a valid signature.

4. If there is an item $(m_j, \perp, \perp, ID_p, ID_k, \sigma_j, P^j_{ID_p}, P^j_{ID_k})$ in H_1-list such that $m_i = m_j$, $\sigma_i = \sigma_j$ and $(ID_p, ID_k, P^j_{ID_p}, P^j_{ID_k}) \neq (ID_m, ID_n, P^i_{ID_m}, P^i_{ID_n})$, $\mathcal{B}$ verifies whether $e(P^i_{ID_m}, P^i_{ID_n}) \stackrel{?}{=} e(P^j_{ID_p}, P^j_{ID_k})$ and $e(Q^i_{ID_m}, Q^i_{ID_n}) \stackrel{?}{=} e(Q^j_{ID_p}, Q^j_{ID_k})$ hold. If the above two conditions hold, $\mathcal{B}$ adds $(m_i, \perp, \perp, ID_m, ID_n, \sigma_j, P^i_{ID_m}, P^i_{ID_n})$ into H_1-list and accepts it as a valid signature.
5. Otherwise, $\mathcal{B}$ queries the sign algorithm with $(m_i, ID_m, ID_n, P^i_{ID_m}, P^i_{ID_n})$ and obtains σ'_i.
   (a) If $\sigma_i = \sigma'_i$, $\mathcal{B}$ accepts it as a valid signature.
   (b) Otherwise, $\mathcal{B}$ rejects it as an invalid signature.

Output: $\mathcal{A}_{\mathcal{I}}$ outputs a valid designated verifier signature $(m^*, ID_A, ID_B, \sigma^*, P^*_{ID_A}, P^*_{ID_B})$ which satisfies the following requirements:

1. $(m^*, ID_A, ID_B, P^*_{ID_A}, P^*_{ID_B})$ has never been submitted as one of **Sign** queries.
2. ID_A and ID_B have never been submitted as one of **CertGen** queries.

Since $(m^*, ID_A, ID_B, \sigma^*, P^*_{ID_A}, P^*_{ID_B})$ is a valid message/signature pair, which means there is an item $(m^*, *, *, *, *, \sigma^*, *, *)$ in H_1-list with probability $1 - 1/(2^l - q_{H_1} - q_S)$. By the definition of adversary model, $(m^*, ID_A, ID_B, P^*_{ID_A}, P^*_{ID_B})$ cannot be queried to the sign oracle, so σ^* is returned as the hash value of $\mathcal{A}_{\mathcal{I}}$' query. That is to say, there is an item $(m^*, K^*_1, K^*_2, *, *, \sigma^*, *, *)$ in the H_1-list such that $e(P, K^*_1) = e(P^*_{ID_A}, P^*_{ID_B})$ and $K^*_2 = e(Q^*_{ID_A}, cert^*_{ID_B})$.

If $\{ID_A, ID_B\} = \{ID_u, ID_v\}$ then $e(Q^*_{ID_A}, cert^*_{ID_B}) = e(P, P)^{abc\alpha^*_{ID_A}\alpha^*_{ID_B}} = K^*_2$. Since $K^*_2, \alpha^*_{ID_A}, \alpha^*_{ID_B}$ are all known to B, B can compute $e(P, P)^{abc} = K^{*(\alpha^*_{ID_A}\alpha^*_{ID_B})^{-1}}_2$. This occurs with probability at least $(1/q_{H_0})^2$, where q_{H_0} is the number of queries sent to the H_0 oracle.

Therefore, $\mathcal{B}$ can solve the given instance of the GBDH problem with probability at least $(1/q_{H_0})^2(1 - 1/(2^l - q_{H_1} - q_S))\varepsilon$.

□

Theorem 2. If there is a type II adversary $\mathcal{A}_{\mathcal{II}}$ can forge a valid signature of the proposed scheme with success probability ε after making q_{UC} user creation queries, q_{H_1} queries to H_1 oracle and q_S signing queries, then there exists an algorithm $\mathcal{B}$ who can use $\mathcal{A}_{\mathcal{II}}$ to solve an instance of the CDH problem with probability at least $\varepsilon' = (1/q_{UC})^2(1 - 1/(2^l - q_{H_1} - q_S))\varepsilon$. ($l$ is the security parameter of CBDVS scheme).

Proof. Given a random instance $(P, P_1 = aP, P_2 = bP)$ of CDH problem, we will show how $\mathcal{B}$ can use $\mathcal{A}_{\mathcal{II}}$ to obtain the value of $abP \in \mathbb{G}_1$. We assume that $\mathcal{A}_{\mathcal{II}}$ will never repeat the same queries in the simulation.

Setup: In this game, $\mathcal{B}$ will set the system parameters. $\mathcal{B}$ randomly chooses $s \in \mathbb{Z}_q^*$ and computes $P_{pub} = sP$. $\mathcal{B}$ returns $\{\mathbb{G}_1, \mathbb{G}_2, q, e, s, P, P_{pub}\}$ to adversary $\mathcal{A}_{\mathcal{II}}$.

UserCreate: $\mathcal{B}$ maintains a list L-list which consists of the tuples $(ID_i, s_{ID_i}, P_{ID_i}, a_i)$ described as follows. The list is initially empty. $\mathcal{B}$ first chooses two integers (u, v) from 1 to q_{UC}. When $\mathcal{A}_{\mathcal{II}}$ queries the oracle with a request ID_i:

1. If $i = u$, $\mathcal{B}$ adds $(ID_i, \perp, P_1)$ into L-list, and returns P_1 to $\mathcal{A}_{\mathcal{II}}$.
2. If $i = v$, $\mathcal{B}$ adds $(ID_i, \perp, P_2)$ into L-list, and returns P_2 to $\mathcal{A}_{\mathcal{II}}$.
3. Otherwise, $\mathcal{B}$ randomly chooses $s_{ID_i} \in \mathbb{Z}_q^*$ and computes $P_{ID_i} = s_{ID_i}P$. Then $\mathcal{B}$ adds $(ID_i, s_{ID_i}, P_{ID_i})$ into L-list, and returns P_{ID_i} to adversary $\mathcal{A}_{\mathcal{II}}$.

Corruption: When adversary $\mathcal{A}_{\mathcal{II}}$ makes a corruption query of a created user ID_i:

1. If $i = u$ or $i = v$, $\mathcal{B}$ aborts the game and the simulation fails.
2. Otherwise, $\mathcal{B}$ returns the private key s_{ID_i} to the adversary $\mathcal{A}_{\mathcal{II}}$.

H_0 **oracle:** At any time $\mathcal{A}_{\mathcal{II}}$ can query the random oracle H_0. $\mathcal{B}$ maintains a list H_0-list which consists of the tuples (ID_i, Q_{ID_i}) described as follows. The list is initially empty. When $\mathcal{A}_{\mathcal{II}}$ queries the oracle H_0 with the request ID_i, $\mathcal{B}$ randomly chooses $Q_{ID_i} \in \mathbb{G}_1$, adds (ID_i, Q_{ID_i}) into L-list, and returns Q_{ID_i} to adversary $\mathcal{A}_{\mathcal{II}}$.

H_1 **oracle:** At any time $\mathcal{A}_{\mathcal{II}}$ can query the random oracle H_1. $\mathcal{B}$ maintains a list H_1-list which consists of the tuples $(m_i, K_{1i}, K_{2i}, ID_m, ID_n, \sigma_i)$ described as follows. The list is initially empty. When $\mathcal{A}_{\mathcal{II}}$ makes a request (m_i, K_{1i}, K_{2i}) to the oracle H_1, $\mathcal{B}$ will search H_1-list.

1. If there is an item $(m_i, K_{1j}, K_{2j}, \perp, \perp, \sigma_j)$ in H_1-list such that $(K_{1i}, K_{2i}) \neq (K_{1j}, K_{2j})$, $\mathcal{B}$ randomly chooses $\sigma_i \in \mathbb{G}_1$ such that there is no item $(*, *, *, *, *, \sigma_i)$ in H_1-list. Then $\mathcal{B}$ adds $(m_i, K_{1i}, K_{2i}, \perp, \perp, \sigma_j)$ into H_1-list, and returns σ_i to adversary $\mathcal{A}_{\mathcal{II}}$.
2. If there is an item $(m_j, \perp, \perp, ID_m, ID_n, \sigma_j)$ in H_1-list such that $m_i = m_j$, $\mathcal{B}$ verifies whether $e(P_{ID_m}, P_{ID_n}) \stackrel{?}{=} e(P, K_{1i})$ and $K_{2i} \stackrel{?}{=} e(Q_{ID_m}, Q_{ID_n})$.
   (a) If the above two conditions hold, $\mathcal{B}$ rewrites this form as $(m_i, K_{1i}, K_{2i}, ID_m, ID_n, \sigma_i)$ and returns σ_i to adversary $\mathcal{A}_{\mathcal{II}}$.
   (b) Otherwise, $\mathcal{B}$ randomly chooses $\sigma_i \in \mathbb{G}_1$ such that there is no item $(*, *, *, *, *, \sigma_i)$ in H_1-list. Then $\mathcal{B}$ adds $(m_i, K_{1i}, K_{2i}, \perp, \perp, \sigma_i)$ into H_1-list and returns σ_i to adversary $\mathcal{A}_{\mathcal{II}}$.
3. If there is an item $(m_j, K_{1j}, K_{2j}, ID_m, ID_n, \sigma_j)$ in H_1-list such that $(m_j, K_{1j}, K_{2j}) = (m_i, K_{1i}, K_{2i})$, $\mathcal{B}$ returns σ_j to adversary $\mathcal{A}_{\mathcal{II}}$.
4. Otherwise, $\mathcal{B}$ randomly chooses $\sigma_i \in \mathbb{G}_1$ such that there is no item $(*, *, *, *, *, \sigma_i)$ in the H_1-list, adds $(m_i, K_{1i}, K_{2i}, \perp, \perp, \sigma_i)$ into H_1-list and returns σ_i to $\mathcal{A}_{\mathcal{II}}$.

Sign Oracle: In this game, $\mathcal{B}$ will simulate the sign algorithm. At any time $\mathcal{A}_{\mathcal{II}}$ can query the sign algorithm. For a signing request (m_i, ID_m, ID_n):

1. If $ID_m \notin \{ID_u, ID_v\}$, $\mathcal{B}$ computes $K_{1i} = S_{ID_m} P_{ID_n}$ and $K_{2i} = e(cert_{ID_m}, Q_{ID_n})$. Then, $\mathcal{B}$ checks the H_1-list.
   (a) If there is an item $(m_i, K_{1i}, K_{2i}, \perp, \perp, \sigma_i)$ in H_1-list, $\mathcal{B}$ rewrites this form as $(m_i, K_{1i}, K_{2i}, ID_m, ID_n, \sigma_i)$ and returns σ_i to adversary $\mathcal{A}_{\mathcal{II}}$.
   (b) If there is an item $(m_i, K_{1i}, K_{2i}, ID_p, ID_k, \sigma_i)$ in H_1-list such that $(ID_p, ID_k) \neq (ID_m, ID_n)$, $\mathcal{B}$ adds $(m_i, K_{1i}, K_{2i}, ID_m, ID_n, \sigma_i)$ into H_1-list and returns σ_i to adversary $\mathcal{A}_{\mathcal{II}}$.
   (c) Otherwise, $\mathcal{B}$ randomly chooses $\sigma_i \in \mathbb{G}_1$ such that there is no item $(*, *, *, *, *, \sigma_i)$ in H_1-list. Then $\mathcal{B}$ adds $(m_i, K_{1i}, K_{2i}, ID_m, ID_n, \sigma_i)$ into H_1-list and returns σ_i to adversary $\mathcal{A}_{\mathcal{II}}$.
2. If $ID_n \notin \{ID_u, ID_v\}$, $\mathcal{B}$ computes $K_{1i} = s_{ID_m} P_{ID_n}$, $K_{2i} = e(cert_{ID_m}, Q_{ID_n})$ and checks H_1-list.
   (a) If there is an item $(m_i, K_{1i}, K_{2i}, \perp, \perp, \sigma_i)$ in H_1-list, $\mathcal{B}$ rewrites this form as $(m_i, K_{1i}, K_{2i}, ID_m, ID_n, \sigma_i)$ and returns σ_i to adversary $\mathcal{A}_{\mathcal{II}}$.
   (b) If there is an item $(m_i, K_{1i}, K_{2i}, ID_p, ID_k, \sigma_i)$ in H_1-list such that $(ID_p, ID_k) \neq (ID_m, ID_n)$, $\mathcal{B}$ adds $(m_i, K_{1i}, K_{2i}, ID_m, ID_n, \sigma_i)$ into H_1-list and returns σ_i to adversary $\mathcal{A}_{\mathcal{II}}$.
   (c) Other, $\mathcal{B}$ randomly chooses $\sigma_i \in \mathbb{G}_1$ such that there is no item $(*, *, *, *, *, \sigma_i)$ in H_1-list. Then $\mathcal{B}$ adds $(m_i, K_{1i}, K_{2i}, ID_m, ID_n, \sigma_i)$ into H_1-list and returns σ_i to adversary $\mathcal{A}_{\mathcal{II}}$.
3. Otherwise,$(ID_m, ID_n) = (ID_u, ID_v)$, $\mathcal{B}$ checks H_1-list.
   (a) If there is an item $(m_j, K_{1j}, K_{2j}, \perp, \perp, \sigma_j)$ in H_1-list such that $m_i = m_j$, $\mathcal{B}$ verifies whether $e(P_{ID_m}, P_{ID_n}) \stackrel{?}{=} e(P, K_{1j})$ and $K_{2j} \stackrel{?}{=} e(sQ_{ID_m}, Q_{ID_n})$.
      i If the above two conditions hold, $\mathcal{B}$ rewrites this form as $(m_i, K_{1j}, K_{2j}, ID_m, ID_n, \sigma_j)$ and returns σ_j to adversary $\mathcal{A}_{\mathcal{II}}$.
      ii Otherwise, $\mathcal{B}$ randomly chooses $\sigma_i \in \mathbb{G}_1$ such that there is no item $(*, *, *, *, *, \sigma_j)$ in H_1-list. Then $\mathcal{B}$ adds $(m_i, \perp, \perp, ID_m, ID_n, \sigma_i)$ into H_1-list and returns σ_i to adversary $\mathcal{A}_{\mathcal{II}}$.
   (b) If there is an item $(m_j, K_{1j}, K_{2j}, ID_p, ID_k, \sigma_j)$ in H_1-list such that $m_i = m_j$ and $(ID_p, ID_k) \neq (ID_m, ID_n)$, $\mathcal{B}$ verifies whether $e(P_{ID_m}, P_{ID_n}) \stackrel{?}{=} e(P, K_{1j})$ and $K_{2j} \stackrel{?}{=} e(sQ_{ID_m}, Q_{ID_n})$.
      i If the above two conditions hold, $\mathcal{B}$ adds $(m_i, K_{1j}, K_{2j}, ID_m, ID_n, \sigma_j)$ into H_1-list and returns σ_j to adversary $\mathcal{A}_{\mathcal{II}}$.
      ii Otherwise, $\mathcal{B}$ randomly chooses $\sigma_i \in \mathbb{G}_1$ such that there is no item $(*, *, *, *, *, \sigma_i)$ in H_1-list. Then $\mathcal{B}$ adds $(m_i, \perp, \perp, ID_m, ID_n, \sigma_i)$ into H_1-list and returns σ_i to adversary $\mathcal{A}_{\mathcal{II}}$.

(c) Otherwise, $\mathcal{B}$ randomly chooses $\sigma_i \in \mathbb{G}_1$ such that there is no item $(*,*,*,*,*,\sigma_i)$ in H_1-list. Then $\mathcal{B}$ adds $(m_i, \perp, \perp, ID_m, ID_n, \sigma_i)$ into H_1-list and returns σ_i to adversary $\mathcal{A}_{\mathcal{II}}$.

Verify Oracle: In this game, $\mathcal{B}$ will simulate the verify algorithm. At any time $\mathcal{A}_{\mathcal{II}}$ can query the verify algorithm. After receiving $\mathcal{A}_{\mathcal{II}}$'s request $(m_i, ID_m, ID_n, \sigma_i)$, $\mathcal{B}$ checks the H_1-list:

1. If there is an item $(m_i, K_{1i}, K_{2i}, ID_m, ID_n, \sigma_i)$ or $(m_i, \perp, \perp, ID_m, ID_n, \sigma_i)$ in H_1-list, $\mathcal{B}$ will accept it as a valid signature.
2. Else, if there is an item $(m_i, K_{1i}, K_{2i}, \perp, \perp, \sigma_i)$ in H_1-list, $\mathcal{B}$ verifies whether $e(P_{ID_m}, P_{ID_n}) \stackrel{?}{=} e(P, K_{1i})$ and $K_{2i} \stackrel{?}{=} e(sQ_{ID_m}, Q_{ID_n})$. If the above two conditions hold, $\mathcal{B}$ rewrites this form as $(m_i, K_{1i}, K_{2i}, ID_m, ID_n, \sigma_i)$ and accepts it as a valid signature.
3. Else, there is an item $(m_j, K_{1j}, K_{2j}, ID_p, ID_k, \sigma_j)$ in H_1-list such that $m_i = m_j$, $\sigma_i = \sigma_j$ and $(ID_p, ID_k) \neq (ID_m, ID_n)$, $\mathcal{B}$ verifies whether $e(P_{ID_m}, P_{ID_n}) \stackrel{?}{=} e(P, K_{1j})$ and $K_{2j} \stackrel{?}{=} e(sQ_{ID_m}, Q_{ID_n})$. If the above two conditions hold, $\mathcal{B}$ adds $(m_i, K_{1j}, K_{2j}, ID_m, ID_n, \sigma_i)$ into H_1-list and accepts it as a valid signature.
4. Otherwise, $\mathcal{B}$ queries the sign algorithm with (m_i, ID_m, ID_n), and obtains the signature σ_i'.
   (a) If $\sigma_i = \sigma_i'$, $\mathcal{B}$ will accept it as a valid signature.
   (b) Otherwise, $\mathcal{B}$ rejects it as an invalid signature.

Output: $\mathcal{A}_{\mathcal{II}}$ outputs a valid designated verifier signature $(m^*, ID_A, ID_B, \sigma^*)$ which satisfies the following requirements:

1. (m^*, ID_A, ID_B) has never been submitted as one of **Sign** queries.
2. ID_A and ID_B have never been submitted as one of **Corruption** queries.

Since $(m^*, ID_A, ID_B, \sigma^*)$ is a valid message/signature pair, which means there is an item $(m^*, *, *, *, *, \sigma^*)$ in H_1-list with probability $1 - 1/(2^l - q_{H_1} - q_S)$. By the definition of adversary model, (m^*, ID_A, ID_B) cannot be queried to the sign oracle, so σ^* is returned as the hash value of $\mathcal{A}_{\mathcal{II}}$'s query. That is to say, there is an item $(m^*, K_1^*, K_2^*, *, *, \sigma^*)$ in the H_1-list such that $K_1^* = s_{ID_A} P_{ID_B} = s_{ID_A} s_{ID_B} P$ and $K_2^* = e(sQ_{ID_A}, Q_{ID_B})$. If $\{ID_A, ID_B\} = \{ID_u, ID_v\}$, then $K_1^* = abP$. This occurs with probability at least $(1/q_{UC})^2$, where q_{UC} is the number of queries sent to the user creation oracle.

The probability that $\mathcal{B}$ does not fail during the simulation is $(1-1/(2^l-q_{H_1}-q_S))$. Therefore, $\mathcal{B}$ can solve the given instance of CDH problem with probability at least $(1/q_{UC})^2(1 - 1/(2^l - q_{H_1} - q_S))\varepsilon$.

□

6 EFFICIENCY ANALYSIS

We use the following table to show the computation cost and communication bandwidth of our scheme. The following notations will be used in Table 1: $|\mathbb{G}_1|$: bit length of elements in $\mathbb{G}_1$; $|\mathbb{G}_2|$: bit length of elements in $\mathbb{G}_2$; BP: bilinear mapping operation; M: scalar multiplication in $\mathbb{G}_1$; E: exponentiation in $\mathbb{G}_1$.

	Sign	**Verify**	**Sign length**
Our scheme	$1BP + 1M$	$1BP + 1M$	$\|\mathbb{G}_1\|$

Table 1. Efficiency analysis

The above analysis shows that our scheme only needs two pairing operations and the signature is only one element in the bilinear group $\mathbb{G}_1$ (about 171 bit, please refer to [32] in detail). Thus our scheme enjoys less operation cost and *shortest* signature length.

7 CONCLUSION

In this paper, we introduced the notion of certificate-based designated verifier signatures (CBDVS). The security of CBDVS against two types of adversaries is formally defined. Furthermore, we proposed an efficient CBDVS scheme. We show that the scheme is existentially unforgeable against adaptive chosen message attacks in the random oracle model. Efficiency analysis shows that the proposed scheme has low computation cost and low communication bandwidth, which makes our scheme possess strong applicability in situations with limited bandwidth and power-constrained devices, for example, RFID devices [33], attribute-based signature [34] and cloud computing [35, 36, 37, 38, 39, 40, 41, 42, 43].

Acknowledgments

We would like to thank anonymous referees for their helpful comments and suggestions. This research was supported by the National Natural Science Foundation of China (61672207, 61272542, 61472083, 61202450), the Fundamental Research Funds for the Central Universities (2016B10114), Jiangsu Provincial Natural Science Foundation of China (BK20161511), the Priority Academic Program Development of Jiangsu Higher Education Institutions and Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology. Fok Ying Tung Education Foundation (141065), Ph.D. Programs Foundation of Ministry of Education of China (20123503120001), Program for New Century Excellent Talents in Fujian University (JA14067), Distinguished Young Scholars Fund of Department of Education, Fujian Province, China (JA13062).

REFERENCES

[1] Jakobsson, M.—Sako, K.—Impagliazzo, R.: Designated Verifier Proofs and Their Applications. In: Maurer, U. (Ed.): Advances in Cryptology – EUROCRYPT '96. Springer-Verlag, Lecture Notes in Computer Science, Vol. 1070, 1996, pp. 143–154.

[2] Lipmaa H.—Wang G.—Bao, F.: Designated Verifier Signature Schemes: Attacks, New Security Notions and a New Construction. In: Caires, L. et al. (Eds.): Automata, Languages and Programming (ICALP 2005). Springer-Verlag, Lecture Notes in Computer Science, Vol. 3580, 2005, pp. 459–471.

[3] Kudla, C.—Paterson, K. G.: Non-Interactive Designated Verifier Proofs and Undeniable Signatures. In: Smart, N. P. (Ed.): Cryptography and Coding. Proceedings of 10th IMA International Conference, 2005. Springer-Verlag, Lecture Notes in Computer Science, Vol. 3796, 2005, pp. 136–154.

[4] Chaum, D.: Zero-Knowledge Undeniable Signatures. In: Damgard, I. B. (Ed.): Advances in Cryptology – EUROCRYPT '90. Springer-Verlag, Lecture Notes in Computer Science, Vol. 473, 1990, pp. 458–464.

[5] Steinfeld, R.—Bull, L.—Wang, H.—Pieprzyk, J.: Universal Designated-Verifier Signatures. In: Laih, C. S. (Ed.): Advances in Cryptology – ASIACRYPT 2003. Springer-Verlag, Lecture Notes in Computer Science, Vol. 2894, 2003, pp. 523–542.

[6] Steinfeld, R.—Wang, H.—Pieprzyk, J.: Efficient Extension of Standard Schnorr/RSA Signatures into Universal Designated-Verifier Signatures. In: Bao, F. et al. (Eds.): Public Key Cryptography – PKC 2004. Springer-Verlag, Lecture Notes in Computer Science, Vol. 2947, 2004, pp. 86–100.

[7] Susilo, W.—Zhang, F.—Mu, Y.: Identity-Based Strong Designated Verifier Signature Schemes. In: Wang, H., Pieprzyk, J., Varadharajan, V. (Eds.): Information Security and Privacy. Proceedings of the ACISP 2004. Springer-Verlag, Lecture Notes in Computer Science, Vol. 3108, 2004, pp. 313–324.

[8] Kang, B. Y.—Boyd, C.—Dawson, E.: A Novel Identity-Based Strong Designated Verifier Signature Scheme. The Journal of Systems and Software, Vol. 82, 2009, No. 2, pp. 270–273.

[9] Al-Riyami, S. S.—Paterson, K. G.: Certificateless Public Key Cryptography. In: Laih, C. S. (Ed.): Advances in Cryptology – ASIACRYPT 2003. Springer-Verlag, Lecture Notes in Computer Science, Vol. 2894, 2003, pp. 452–473.

[10] Li, J. G.—Huang, X. Y.—Mu, Y.—Wu, W.: Cryptanalysis and Improvement of an Efficient Certificateless Signature Scheme. Journal of Communications and Networks, Vol. 10, 2008, No. 1, pp. 10–17.

[11] Huang, X. Y.—Susilo, W.—Mu, Y.—Zhang, F. T.: Certificateless Designated Verifier Signature Schemes. Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA '06), Vol. 2, 2006, pp. 15–19.

[12] Chen, H.—Song, R. S.—Zhang, F. T.—Song, F. G.: An Efficient Certificateless Short Designated Verifier Signature Scheme. Proceedings of the 4th Interna-

tional Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '08), 2008, pp. 1–6.

[13] Yang, B.—Hu, Z. M.—Xiao, Z. B.: Efficient Certificateless Strong Designated Verifier Signature Scheme. Proceedings of Computational Intelligence and Security (CIS '09), 2009, pp. 432–436.

[14] Gentry, C.: Certificate-Based Encryption and the Certificate Revocation Problem. In: Biham, E. (Ed.): Advances in Cryptology – EUROCRYPT 2003. Springer-Verlag, Lecture Notes in Computer Science, Vol. 2656, 2003, pp. 272–293.

[15] Kang, B. G.—Park, J. H.—Hahn, S. G.: A Certificate-Based Signature Scheme. In: Okamoto, T. (Ed.): Topics in Cryptology – CT-RSA 2004. Springer-Verlag, Lecture Notes in Computer Science, Vol. 2964, 2004, pp. 99–111.

[16] Li, J. G.—Huang, X. Y.—Mu, Y.—Susilo W.—Wu, Q. H.: Certificate-Based Signature: Security Model and Efficient Construction. In: Lopez, J., Samarati, P., Ferrer, J. L. (Eds.): Public Key Infrastructure (EuroPKI 2007). Springer-Verlag, Lecture Notes in Computer Science, Vol. 4582, 2007, pp. 110–125.

[17] Au, M. H.—Liu, J. K.—Susilo, W.—Yuen, T. H.: Certificate Based (Linkable) Ring Signature. In: Dawson, E., Wong, D. S. (Eds.): Information Security Practice and Experience (ISPEC 2007). Springer-Verlag, Lecture Notes in Computer Science, Vol. 4464, 2007, pp. 79–92.

[18] Liu, J.—Baek, J.—Susilo, W.—Zhou, J.: Certificate-Based Signature Schemes without Pairings or Random Oracles. In: Wu, T.-C. et al. (Eds.): Information Security (ICS 2008). Springer-Verlag, Lecture Notes in Computer Science, Vol. 5222, 2008, pp. 285–297.

[19] Zhang, J. H.: On the Security of a Certificate-Based Signature Scheme and Its Improvement with Pairings. In: Bao, F., Li, H., Wang, G. (Eds.): Information Security Practice and Experience (ISPEC 2009). Springer-Verlag, Lecture Notes in Computer Science, Vol. 5451, 2009, pp. 47–58.

[20] Li, J. G.—Xu, L. Z. Zhang, Y. C.: Provably Secure Certificate-Based Proxy Signature Schemes. Journal of Computers, Vol. 4, 2009, No. 6, pp. 444–452.

[21] Ming, Y.—Wang, Y. M.: Efficient Certificate-Based Signature Scheme. Proceedings of the Fifth International Conference on Information Assurance and Security. Vol. 2, 2009, pp. 87–90.

[22] Wu, W.—Mu, Y.—Susilo, W.—Huang, X. Y.: Certificate-Based Signatures: New Definitions and a Generic Construction from Certificateless Signatures. In: Chung, K. I., Sohn, K., Yung, M. (Eds): Information Security Applications (WISA 2008). Springer-Verlag, Lecture Notes in Computer Science, Vol. 5379, 2009, pp. 99–114.

[23] Li, J. G.—Huang, X. Y.—Mu, Y.—Susilo, W.—Wu, Q. H.: Constructions of Certificate-Based Signature Secure against Key Replacement Attacks. Journal of Computer Security, Vol. 18, 2010, No. 3, pp. 421–449.

[24] Li, J. G.—Huang, X. Y.—Zhang, Y. C.—Xu, L. Z.: An Efficient Short Certificate-Based Signature Scheme. Journal of Systems and Software, Vol. 85, 2012, No. 2, pp. 314–322.

[25] Li, J. G.—Wang, Z. W.—Zhang, Y. C.: Provably Secure Certificate-Based Signature Scheme without Pairings. Information Sciences, Vol. 233, 2013, No. 6, pp. 313–320.

[26] Li, J. G.—Huang, X. Y.—Hong, M. X.—Zhang, Y. C.: Certificate-Based Signcryption with Enhanced Security Features. Computers and Mathematics with Applications, Vol. 64, 2012, No. 6, pp. 1587–1601.

[27] Li, J. G.—Zhang, Y. C.—Teng, H. Y.: A Forward-Secure Certificate-Based Signature Scheme in the Standard Model. In: Xiang, Y., Lopez, J., Kuo, C. C. J., Zhou, W. (Eds): Cyberspace Safety and Security (CSS 2012). Springer-Verlag, Lecture Notes in Computer Science, Vol. 7672, 2012, pp. 362–376.

[28] Li, J. G.—Teng, H. Y.—Huang, X. Y.—Zhang, Y. C.—Zhou, J.: A Forward-Secure Certificate-Based Signature Scheme. The Computer Journal, Vol. 58, 2015, No. 4, pp. 853–866. doi: 10.1093/comjnl/bxt141

[29] Du, H. T.—Li, J. G.—Zhang, Y. C.—Li, T.—Zhang, Y. X.: Certificate-Based Key-Insulated Signature. In: Xiang, Y., Pathan, M., Tao, X., Wang, H. (Eds): Data and Knowledge Engineering (ICDKE 2012). Springer-Verlag, Lecture Notes in Computer Science, Vol. 7696, 2012, pp. 206–220.

[30] Li, J. G.—Du, H. T.—Zhang, Y. C.—Li, T.—Zhang, Y. X.: Provably Secure Certificate-Based Key-Insulated Signature Scheme. Concurrency and Computation: Practice and Experience, Vol. 26, 2014, pp. 1546–1560.

[31] Guttman, P.: PKI: Its Not Dead, Just Resting. IEEE Computer, Vol. 35, 2002, pp. 41–49.

[32] Boneh, D.—Lynn, B.—Shacham, H.: Short Signatures from the Weil Pairing. Journal of Cryptology, Vol. 17, 2004, No. 4, pp. 297–319.

[33] Qian, Z. Z.—Chen, C.—You, I.—Lu S. L.: ACSP: A Novel Security Protocol Against Counting Attack for UHF RFID Systems. Computers and Mathematics with Applications, Vol. 63, 2012, pp. 492–500.

[34] Su, J. S.—Cao, D.—Zhao, B. K.—Wang, X. F.—You, I.: ePASS: An Expressive Attribute-Based Signature Scheme with Privacy and an Unforgeability Guarantee for the Internet of Things. Future Generation Computer Systems, Vol. 33, 2014, pp. 11–18.

[35] Li, J.—Chen, X.F: Efficient Multi-User Keyword Search over Encrypted Data in Cloud Computing. Computing and Informatics, Vol. 32, 2013, No. 4, pp. 723–738.

[36] Nguyen, B. M.—Tran, V.—Hluchy, L.: A Generic Development and Deployment Framework for Cloud Computing and Distributed Applications. Computing and Informatics, Vol. 32, 2013, pp. 461–485.

[37] Yan, H.—Li, J.—Han J.—Zhang Y.: A Novel Efficient Remote Data Possession Checking Protocol in Cloud Storage. IEEE Transactions on Information Forensics and Security, Vol. 12, 2017, pp. 78–88.

[38] Fu, Z.—Sun, X.—Liu, Q.—Zhou, L.—Shu, J.: Achieving Efficient Cloud Search Services: Multi-Keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing. IEICE Transactions on Communications, Vol. 98, 2015, pp. 190–200.

[39] Xia, Z.—Wang, X.—Sun, X.—Wang, Q.: A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data. IEEE Transactions on Parallel and Distributed Systems, Vol. 27, 2015, pp. 340–352.

[40] Li, J.—Yao, W.—Zhang, Y.—Qian, H.—Han, J.: Flexible and Fine-Grained Attribute-Based Data Storage in Cloud Computing. IEEE Transactions on Services Computing, 2016, DOI: 10.1109/TSC.2016.2520932.

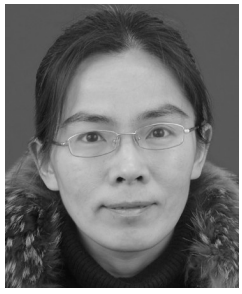
[41] Li, J.—Lin, X.—Zhang, Y.—Han, J.: KSF-OABE: Outsourced Attribute-Based Encryption with Keyword Search Function for Cloud Storage. IEEE Transactions on Services Computing, 2016, DOI: 10.1109/TSC.2016.2542813.

[42] Ren, Y.—Shen, J.—Wang, J.—Han, J.—Lee, S.: Mutual Verifiable Provable Data Auditing in Public Cloud Storage. Journal of Internet Technology, Vol. 16, 2015, pp. 317–323.

[43] Fu, Z.—Ren, K.—Shu, J.—Sun, X.—Huang, F.: Enabling Personalized Search over Encrypted Outsourced Data with Efficiency Improvement. IEEE Transactions on Parallel and Distributed Systems, Vol. 27, 2016, pp. 2546–2559.

Jiguo Li received his B.Sc. degree in mathematics from Heilongjiang University, Harbin, China in 1996, M.Sc. degree in mathematics and his Ph.D. degree in computer science from Harbin Institute of Technology, Harbin, China in 2000 and 2003, respectively. He was a visiting scholar in the School of Computer Science and Software Engineering and the Centre for Computer and Information Security Research, University of Wollongong, Australia between 2006 and 2007. He was a visiting scholar in the Institute for Cyber Security (ICS) of UTSA, USA between 2013 and 2014. He is currently Professor in the College of Computer and Information Engineering, Hohai University, Nanjing, China. He is currently holding the National Natural Science Foundation of China, the Fundamental Research Funds for the Central Universities, and the "Six Talent Peaks Program" of Jiangsu Province of China. His research interests include cryptography, network security, wireless security and trusted computing, etc. He has published more than 90 referred research papers and two books. He has served as PC member of several international conferences, and served as the reviewer of some international journals and conferences.

Na Qian received her B.Sc. degree and M.Sc. degree from the College of Computer and Information Engineering, Hohai University, Nanjing, China in 2007 and 2010, respectively. Her research interests include cryptography and network security.



Yichen Zhang received her B.Sc. degree in computer science from the Qiqihar University, Qiqihar, China in 1995. She is currently lecturer and Ph.D. student in the College of Computer and Information Engineering, Hohai University, Nanjing, China. Her research interests include cryptography and network security. She has published more than 20 refereed research papers in international conferences and journals.

Xinyi Huang received his Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. He is currently Professor at the Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Mathematics and Computer Science, Fujian Normal University, China. His research interests include cryptography and information security. He has published over 80 research papers in refereed international conferences and journals. His work has been cited more than 1 500 times at Google Scholar (h-index: 22). He is in the Editorial Board of International Journal of Information Security (IJIS, Springer) and has served as the program/general chair or program committee member in over 60 international conferences.