Certificateless Designated Verifier Proxy Signature

Cholun Kim [1]

[1]. Faculty of Mathematics, Kim Il Sung University, Pyongyang, Democratic People's Republic of Korea

E-mail: cu.kim1019@ryongnamsan.edu.kp

Abstract

Proxy signature (PS) is a kind of digital signature, in which an entity called original signer can delegate his signing rights to another entity called proxy signer. Designated verifier signature (DVS) is a kind of digital signature where the authenticity of any signature can be verified by only one verifier who is designated by the signer when generating it. Designated verifier proxy signature (DVPS) combines the idea of DVS with the concept of proxy signature (PS) and is suitable for being applied in many scenarios from e-tender, e-voting, e-auction, e-health and e-commerce, etc. Many DVPS schemes have been proposed and Identity-based DVPS (IBDVPS) schemes have also been discussed. Certificateless public-key cryptography (CL-PKC) is acknowledged as an appealing paradigm because there exists neither the certificate management issue as in traditional PKI nor private key escrow problem as in Identity-based setting. A number of certificateless designated verifier signature (CLDVS) schemes and many certificateless proxy signature (CLPS) schemes have been proposed. However, to the best of our knowledge, the concept of Certificateless Designated Verifier Proxy Signature (CLDVPS) has not been appeared in the literature.

In this paper, we formalize the definition and the security model of CLDVPS schemes. We then construct the first CLDVPS scheme and prove its security.

Keywords: certificateless public-key cryptography; designated verifier proxy signature; certificateless designated verifier proxy signature; random oracle model; provable security; bilinear pairings.

1. Introduction

Digital signature is one of the most important primitives in cryptography and serves as a powerful tool in information security, including authentication, data integrity and non-repudiation in a public channel. It falls into the category of public-key cryptography (PKC), where each user has two keys, i.e., a public key and a corresponding private key, and it is an important issue to guarantee the authenticity of public keys which match with private keys. Digital signature schemes allow any signer to sign a message using its private key in such a way that any other (i.e., verifier) who has an authentic copy of the public key associated with the signer can verify the origin and the integrity of the message. In traditional public key setting such as PKI, *certificates* generated by a *trusted third party* (TTP) are used to distribute authentic public keys, but it is costly to use and manage them, and brings about new problems. In order to bypass those problems, Shamir [51] introduced the concept of *Identity-based Signature* (IBS) along with *Identity-based encryption* (IBE). In IBS, the public key of a user is its identity information (e.g., name, email address, IP address, or social security number) which is approved publicly and doesn't need to be certified by any TTP, but all private keys are only generated by a TTP called *private key generator* (PKG). Therefore, the PKG can freely generate any user's signature on any message and this results in the *private key escrow problem* which is inherent in the identity-based setting. In 2003, Al-Riyami and Paterson [1] introduced *certificateless* public-key cryptography (CL-PKC) and the first *certificateless signature* (CLS) scheme to solve the private key escrow problem in the identity-based setting and eliminate the need of certificates in the conventional PKI. In the certificateless setting, a TTP named a *key generation center* (KGC) is also required for a user to generate its private key, but unlike a PKG in IBS schemes, the KGC only produces a partial private key for any user by using the user's identity and a master secret key, and transfers it to the user securely. Then the user generates its own full private key by combining the partial private key generated by KGC with a secret value chosen by itself. As a result, the KGC cannot know the full private key of any user and the key escrow problem in the identity-based setting can be solved. Meanwhile, the public key for each user in the CLS schemes is no longer its identity, but it can be certified by the structure of the CL-PKC without any certificate.

Proxy signature, introduced by Mambo et al. [44] in 1996, is a kind of digital signature, in which a user called original signer can delegate his signing right to another user called proxy signer and the proxy signer can sign messages on behalf of the original signer, in case of temporal absence of him, lack of his time, etc. Proxy signature schemes are widely used in distributed systems where delegation of signing rights is quite common, including cloud computing, global distribution networks and electronic commerce. On the basis of the delegation mode, proxy signature schemes can be classified three types [44]: *full delegation*, *partial delegation* and *delegation by warrant*. In the *full delegation*, the original signer entirely grants his private key (signing key) to the proxy signer. Therefore, the proxy signer has all signing right of the original signer and such schemes have no non-repudiation. In the *partial delegation*, a proxy signer obtains a new key, called *proxy signing key*, which is derived from the private key (signing key) of the original signer and the public or private key of the proxy signer. Obviously, proxy signatures which is generated with proxy signing key are distinguished from the original signer's signatures, but the proxy signature with partial delegation still suffers from the problem that the proxy signer has no limit on which messages and when he can sign. In the *delegation by warrant*, this problem can be solved. Warrants are usually written by the original signers and can include identities of the original signer and the proxy signer, the delegation period, the associated public keys, and other information. Actually, it is possible to contain any types of security policies that specify the restrictions under which the delegation is valid, such as the formats of messages to be signed, the number of signatures to be generated by proxy signer, etc. The original signers prepare the warrants at their will and sign them to certify the legitimacy of the proxy signers and be able to delegate the limited signing right to the proxy signers. Therefore, most of works in the literature focus on the proxy signature schemes with the delegation by warrant. The warrant signed by the original signer is called the *delegation*.

In general, a secure proxy signature scheme should satisfy the following security requirements [44]: (1) *Strong Unforgeability*: Only the designated proxy signer can create a valid proxy signature and even the original signer cannot. (2) *Verifiability*: Anyone can verify the proxy signature and its conformance to the original signer's agreement or delegation. (3) *Strong Identifiability*: From the proxy signature, anyone can determine the identity of corresponding proxy signer. (4) *Strong Undeniability*: A proxy signer cannot deny his or her signature ever generated against anyone. (5) *Distinguishability*: Anyone can distinguish the proxy signature from the original signer's normal signature. (6) *Prevention of misuse*: The proxy signer cannot sign messages that have not been authorized by the original signer.

In typical digital signature schemes, signatures generated by any signer are publicly verifiable and transferable, and have the non-repudiation property. In other words, anyone who gets a signature and has the signer's public key can verify the authenticity of the signature, and transfer his conviction to anyone else by presenting the signature. Furthermore, no signer can deny the signature generated by him. However, the public verifiability and non-repudiation property of signatures is not always desirable in some scenarios where the privacy of the signer must be protected, such as e-tender, e-voting, e-auction, e-health and e-commerce applications. In 1996, Jakobsson et al. [28] suggested the notion of *designated verifier signature* (or *proof*) (DVS) as a solution for resolving the conflict between authenticity and privacy of signatures. In DVS, the authenticity of any signature can be verified by only one verifier who is designated by the signer when generating it, in a non-interactive way. That is, for any signature in DVS schemes, no one else than the designated verifier of the signature can get the conviction on the identity of the signer who generated it, while the designated verifier cannot transfer his conviction to anyone else since he can always simulate the signature indistinguishably. Anyone can verify the validity of any signature (i.e., the fact that the signature is generated by one of the signer and the designated verifier), but no one except the designated verifier can confirm the identity of the true signer due to the indistinguishability between the original signature and the fake signature (*transcript*) simulated by the designated verifier even if the private keys of the signer and the designated verifier are revealed. Jakobsson et al. [28] also proposed the idea of a *strong DVS* (SDVS), where the designated verifier's private key is required in the verification algorithm so that no one can even verify the validity of signatures except the designated verifier. Steinfeld et al. [54] extended the concept of DVS to the one of *universal* DVS (UDVS), in which anyone who holds the standard signature on a message can transform it to a DVS to protect the privacy of the signature holder from dissemination of signatures by verifiers. In 2004, Laguillaumie and Vergnaud [34] proposed the new primitive called *multi-designated verifiers signatures* (MDVS) where the signer can designate multiple verifiers for one signature and each verifier can verify the signature individually.

There are some security requirements for secure DVS schemes. The correctness and the unforgeability are the *de facto* standard requirements for digital signatures. The *correctness* requires that any signature properly produced by a legitimate signer must be accepted by the designated verifier. The *unforgeability* means that it is infeasible to produce a valid signature without the knowledge of the private key of either the signer or the designated verifier. The natural requirement derived from the essence of the DVS is the *non-transferability* which is identified by Steinfeld et al. [54]. The non-transferability means that a designated verifier for any signature cannot use the signature to produce the evidence which convinces any third party that the signature was generated by the signer. As the requirement for anonymity of signatures, Laguillaumie and Vergnaud [35] suggested a weaker notion for ordinary DVS, *source hiding* and a stronger notion for strong DVS, *privacy of signer's identity* (*PSI*). Source hiding means that given a message and a signature on it, it is infeasible to decide who from the signer or the designated verifier generated the signature, even if the private keys of the signer and designated verifier are known. Privacy of signer's identity means that given a signature and the public keys of two potential signers for the signature, it is infeasible to determine under which of the two corresponding private keys the signature was generated, without the knowledge of the private key of the designated verifier. Lipmaa et.al [40] identified a new security requirement for secure DVS schemes, called the *non-delegatability*. The non-delegatability requires that if a third party can produce a valid signature with respect to a designated verifier or identify the identity of the signer from the signature, it must 'know' the private key of either the original signer or the designated verifier. It is a critical property in applications where responsibilities of signers and designated verifiers are so important that they cannot be delegated to third parties.

In 2003, Dai et al. [8] proposed a primitive called a designated verifier proxy signature (DVPS) by combining the idea of DVS with the concept of proxy signature (PS) to capture the security requirements from e-commerce applications, such as sale of digital products (softwares, games, digital music, e-books, etc.) In the definition of DVPS proposed by Dai et al. [8], the verifiers of proxy signatures are designated by original signers in advance and this point makes their DVPS slightly inflexible. In 2004, Wang [61] suggested an improved DVPS scheme where verifiers of proxy signatures are designated by proxy signers instead of the original signers. Then vaious DVPS schemes have been discussed in many literatures [18,17,24,26,30,36,37,41,59,65,68,69].

1.1 Motivation

In 2005, Cao et al. [2] introduced the idea of DVPS into the identity-based setting to propose the first *identity-based DVPS* (IBDVPS) scheme. Along their research line, Lal and Verma [36, 37], Kang et al. [31], Yang [65], Islam and Biswas [24, 26], Hu et al. [18], etc., successively proposed various IBDVPS schemes. On the other hand, in 2006, the first DVS scheme in the certificateless setting, namely a *certificateless DVS* (CLDVS) scheme, proposed by Huang et al. [23]. Since then, Du and Wen [10], Yang et al. [64], He and Chen [14], Islam and Biswas [25], Chen et al. [5], Pakniat [46], Rastegari et al. [48] etc., also proposed CLDVS schemes. In 2005, Li et al. [38] combined the concept of proxy signature with CL-PKC to propose the notion of *certificateless proxy signature* (CLPS). Then the CLPS and its some extensions (e.g. *certificateless multi-proxy signature*, *certificateless proxy multi-signature*, etc) have been discussed in many literatures [3,4,11,29,43,45,58,60,62,63,70].

It is natural to combine the concept of DVPS with CL-PKC which is more flexible public-key setting than the identity-based setting. However, to the best of our knowledge, there are no researches in this direction up to now.

Generally, the security of any cryptographic primitive is ensured by a formal proof, neither by a heuristic proof nor by a claim without any proof, and formal proofs of the security are based on formal definitions and precise assumptions. The power of adversaries and the security of the primitive are defined within the security model of the primitive. A lot of previously proposed primitives without a formal security proof turned out to be insecure subsequently to their release, then fixed, but they would be proved to be insecure again. We can see this phenomenon on IBDVPS schemes and CLDVS schemes in the literature. For example, Kang et al. [31] showed that Lal and Verma's scheme in [36] is not secure. Later, the insecurity of Kang et al.'s IBDVPS scheme in [31] was proved. The first IBDVPS scheme proposed by Cao et al. in [2], and Yang's scheme in [65] have not a formal proof of their security. Islam and Biswas [24] proposed an IBDVPS scheme with only heuristic security proof. Islam and Biswas [26] again proposed an IBDVPS scheme and presented a formal security proof, but did not give a formal definition of the security model. Meanwhile, Huang et al.'s CLDVS scheme in [23] is insecure against malicious KGC attacks [46]. Lin et al. [39] showed that Islam and Biswas's CLDVS scheme in [25] is vulnerable to key-compromise and malicious KGC attacks. Pakniat [46] disproved Chen et al.'s claim that their certificateless strong DVS scheme in [5] satisfies all the security requirements, and showed that it is forgeable. Li et al. [38] proposed the first scheme of the CLPS without any formal security proof. Later, Yap et al. [66] and Lu et al. [42] found that Li et al.'s scheme is insecure. Furthermore, Lu et al. [42] improved Li et al.'s CLPS scheme, but they, just as Li et al. [38], did not present any formal security proof for the scheme. Padhye et al. [45] proposed an elliptic curve discrete log problem (ECDLP)-based CLPS scheme without pairings. They presented a formal security proof, but Shi et al. [53] showed their scheme is not secure for practical applications. This illustrates the necessity of well-defined security models and a rigorous security analysis.

Hu et al. [18] presented the formal definition of an IBDVPS scheme and a formal security model for it, focusing on only the unforgeability. Hu et al. [17] proposed a well-defined security model for DVPS schemes. Huang et al. [22], Huang et al. [20], Tian et al. [57], Yoneyama et al. [67], Sharma et al. [52], Islam and Biswas [27], Hu et al. [19], Steinfeld et al. [54], Huang et al. [21], Laguillaumie et al. [33], Rastegari et al. [49], Laguillaumie et al. [34], Damgård et al. [9], Rastegari and Berenjkoub [47], etc. proposed a formal security model for *identity-based DVS* (IBDVS), SDVS, UDVS, or MDVS schemes, respectively. A number of security models for the CLPS schemes were proposed by Wan et al. [60], Chen et al. [3], Chen et al. [4], Jin et al. [29], Zhang et al. [70], Xu et al. [63], Du et al. [11], Padhye et al. [45], etc. However, we cannot find a formal security model for CLDVS schemes in the literature.

Motivated by the above discussion, we propose the notion of certificateless designated verifier proxy signature (CLDVPS), the security model for CLDVPS schemes and the concrete CLDVPS scheme in this paper.

1.2 Our contribution

For the first time in the literature, we introduce the notion of certificateless (strong) designated verifier proxy signature (CLDVPS) as a kind of strong DVS, formalize the security model for CLDVPS schemes and propose the first CLDVPS scheme in this paper.

The main contributions of this paper can be summarized as follows:

(1) Firstly, we formalize the definition of a CLDVPS scheme and propose the formal security model for CLDVPS schemes. Referring to Hu et al.'s security model [18] for IBDVPS schemes, we propose a formal definition of a CLDVPS scheme, then improve the security models for CLPS schemes from [4, 63, 70, 11, 45] and combine them with Hu et al.'s security model [17] for DVPS schemes and other models for DVS to construct the first security model for CLDVPS schemes.

(2) Secondly, we propose a concrete CLDVPS scheme. This is the first CLDVPS scheme in the literature. Our scheme is inspired by Hu et al.'s DVPS scheme [17], Islam and Biswas's IBDVPS scheme [26], and CLPS schemes from [70, 45].

(3) Finally, based on our new security model for CLDVPS schemes, we prove that our scheme is secure in the random oracle model under the assumption that the computational Diffie–Hellman (CDH) problem and Gap Bilinear Diffie–Hellman (GBDH) problem are intractable.

1.3 Organization

The remainder of this paper is organized as follows. In Section 2, some preliminaries are given. In Section 3, we present the definition of a CLDVPS scheme and formalize the security model for CLDVPS schemes. Then a concrete CLDVPS scheme is proposed in Section 4. We prove the security of the proposed scheme in Section 5. Finally, we conclude the paper in Section 6.

2. Notations and Preliminaries

Throughout this paper, we will use following notations:

- $\lambda \in \mathrm{N}$: the security parameter;
- 1^{λ} : the string consisting of λ ones;
- $s \leftarrow_{\mathrm{R}} S$: the operation of picking an element s uniformly at random from a finite set S ;
- $r \hookleftarrow \mathrm{A}(i_1,\cdots,i_m)$: the operation of running a probabilistic (randomized) algorithm A with input $i_1,\cdots,i_m$ and assigning the result to r ;

- $r \longleftarrow \mathrm{A}(i_1,\cdots,i_m)$: the operation of running a deterministic algorithm A with input $i_1,\cdots,i_m$ and assigning the result to r ;
- $V_1 \ / \ V_2$: one of two values V_1 and V_2 ;
- $V_1 \| V_2$: the concatenation of the (binary) string representations of two values V_1 and V_2 ;
- $L := R$: the assignment of the value of R to L .
- $\perp$: the null symbol indicating a failure or a false value;
- ∅: the empty set or the empty string.
- Let Γ_1 be a cyclic additive group of prime order q with the addition "+". If $P \in \Gamma_1$ and $k \in \mathrm{Z}_q$, we denote

  $\underbrace{P+\cdots+P}_{k \text{ times}} \in \Gamma$ as $[k]P$.
- PPT stands for *probabilistic polynomial-time*.
- Pr[E] stands for the probability of the event E.

A function $f : \mathrm{N} \rightarrow \mathrm{P}$ is said to be *negligible* if, for all polynomial p , there exists an integer $k_p \in \mathrm{N}_1$ such that $|f(n)| < 1/|p(n)|$ for all $n > k_p$, or equivalently, if for all $c \in \mathrm{N}_1$ there exists an integer $k_c \in \mathrm{N}_1$ such that $|f(n)| < n^{-c}$ for all $n > k_c$.

Definition 1. Let $\Gamma_{\mathrm{A}1}$ be a cyclic additive group of prime order q and $\Gamma_{\mathrm{M}1}$ be a cyclic multiplicative group of the same order with the identity element $1_{\mathrm{M}1}$. A map $\hat{e} : \Gamma_{\mathrm{A}} \times \Gamma_{\mathrm{A}} \rightarrow \Gamma_{\mathrm{M}}$ is called a *bilinear pairing* if it has the following properties:

(1) Bilinearity: $\hat{e}(P, Q+R) = \hat{e}(P, Q) \cdot \hat{e}(P, R)$ and $\hat{e}(P+Q, R) = \hat{e}(P, R) \cdot \hat{e}(Q, R)$ for all P , Q , $R \in \Gamma_{\mathrm{A}}$.

(2) Non-degeneracy: There exists $P \in \Gamma_{\mathrm{A}}$ such that $\hat{e}(P, P) \neq 1_{\mathrm{M}}$.

(3) Efficient Computability: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for all P , $Q \in \Gamma_{\mathrm{A}}$.

$(\Gamma_{\mathrm{A}1}, \Gamma_{\mathrm{M}1})$ is called a *bilinear group* pair.

Numerous computational problems on bilinear pairings have been defined.

Definition 2. Let $\Gamma_{\mathrm{A}1}$ be a cyclic additive group of prime order q , G be a generator of $\Gamma_{\mathrm{A}1}$, $\Gamma_{\mathrm{M}1}$ be a cyclic multiplicative group of prime order q , $\hat{e} : \Gamma_{\mathrm{A}} \times \Gamma_{\mathrm{A}} \rightarrow \Gamma_{\mathrm{M}}$ be a bilinear pairing.

Computational Diffie–Hellman (**CDH**) Problem in Γ_{A}: Given G, and $[u]G$, $[v]G \in \Gamma_{\mathrm{A}}$ for some unknown $u, v \in Z_q^*$, compute $[uv]G \in \Gamma_{\mathrm{A}}$.

Decisional Bilinear Diffie–Hellman (**DBDH**) Problem in $(\Gamma_{\mathrm{A}}, \Gamma_{\mathrm{M}})$: Given $h \in \Gamma_{\mathrm{M}}$, G, and $[u]G$, $[v]G$, $[w]G \in \Gamma_{\mathrm{A}}$ for some unknown $u, v, w \in Z_q^*$, decide whether $h = \hat{e}(G, G)^{uvw}$ or not.

A DBDH oracle $\mathrm{O}_{\mathrm{DBDH}}$ is an algorithm that on input G, $[u]G$, $[v]G$, $[w]G \in \Gamma_{\mathrm{A}}$ and $h \in \Gamma_{\mathrm{M}}$, output 1 if $h = \hat{e}(G, G)^{uvw}$, or 0 otherwise.

Gap Bilinear Diffie–Hellman (**GBDH**) Problem in $(\Gamma_{\mathrm{A}}, \Gamma_{\mathrm{M}})$: Given G, $[u]G$, $[v]G$, $[w]G \in \Gamma_{\mathrm{A}}$, compute $\hat{e}(G, G)^{uvw} \in \Gamma_{\mathrm{M}}$ with the help of DBDH oracle $\mathrm{O}_{\mathrm{DBDH}}$.

In CL-PKC, two types of adversaries with different capabilities–type I adversaries and type II adversaries, are considered [1];

- A type I adversary A_{I} acts as a dishonest user, and can replace the public key of any user with his choice but cannot access to the master secret key.
- A type II adversary A_{II} acts as a malicious-but-passive KGC, and knows the master secret key and partia l private keys of all user but cannot replace any user's public key.

3. Certificateless designated verifier proxy signature: formal definitions and security model

In this section, we describe the formal definition of certificateless designated verifier proxy signature (CLDVPS) schemes within the category of the strong designated verifier signature (SDVS) and propose security model for CLDVPS schemes. In SDVS schemes, the generation of any signature involves the public key of the designated verifier and the verification of the signature needs the private key of the designated verifier. That is, a strong DVS (SDVS) scheme is more suitable to the main goal of the DVS, i.e., the guarantee of the restrictive verifiability or the anonymity of signatures, and practical applications than any ordinary DVS scheme where anyone can decide whether any signature has been generated by one of the signer and the designated verifier or not, but cannot determine who the true signer is, excepting the designated verifier. In fact, the notion of the ordinary DVS corresponds to the one of the ring signature [50] in the particular case where the set *U* of possible signers contains two users, *U* := {*S*, *D*}, i.e., the signer *S* and the designated verifier *D*. If *S* generates a ring signature for the set *U*, then anyone can verify that either *S* or *D* has created the signature. User *D*, of course, knows that he has not computed this signature, so he concludes that the actual author of the signature is *S* [15].

3.1 Definition of a CLDVPS scheme

We first define the CLDVPS scheme formally, and then emphasize some features of the new definition.

Definition 3. A *certificateless (strong) designated verifier proxy signature (CLDVPS)* (*with appendix*) *scheme* $\mathbf{\Sigma}(\lambda)$ comprises legitimate entities of four types: the KGC, the original signer, the proxy signer and the designated verifier. It consists of the following eight polynomial-time algorithms:

(1) $(Params, s) \hookleftarrow$ Setup(1^{λ}): This is a probabilistic (randomized) algorithm run by the KGC, takes as input 1^{λ}, and returns a list of public system parameters $Params$ and a *master secret key* s.

(2) $(\mathrm{P}_U, \mathrm{s}_U) \hookleftarrow$ GenerateUserKeys($Params, \mathrm{ID}_U$): This is a probabilistic algorithm run by any entity except the KGC. When a user U run, it takes as input $Params$ and his identity ID_U, and outputs a *public/secret key* pair $(\mathrm{P}_U, \mathrm{s}_U)$ for the user U.

(3) $(\mathrm{K}_U, \mathrm{c}_U) \hookleftarrow$ GeneratePartialKeys($Params, s, \mathrm{ID}_U, \mathrm{P}_U$): This is a probabilistic algorithm run by the KGC, takes as input $Params$, the master secret key s, the identity ID_U of a user U and his public key P_U, and outputs a pair of *partial public / private keys* $(\mathrm{K}_U, \mathrm{c}_U)$ for the user U. $(\mathrm{P}_U, \mathrm{K}_U)/(\mathrm{s}_U, \mathrm{c}_U)$ is the pair of *full public / private keys* for the user U.

(4) $\delta_O \hookleftarrow$ GenerateDelegation($Params, \mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \mathrm{s}_O, \mathrm{c}_O, \mathrm{w}_O$): This is a probabilistic algorithm run by original signers. When an original signer O run, it takes as input $Params$, the identity ID_O of O, his full public/private key pair $(\mathrm{P}_O, \mathrm{K}_O)/(\mathrm{s}_O, \mathrm{c}_O)$, and a warrant w_O issued by O, and then outputs the delegation δ_O certified by O. δ_O includes w_O.

(5) **True**/**False** $\longleftarrow$ VerifyDelegation($Params, \mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O$): This is a deterministic algorithm run by any user. It takes as input $Params$, the identity ID_O of an original signer O, his full public key $(\mathrm{P}_O, \mathrm{K}_O)$ and the delegation δ_O issued by O, and outputs **True** if the delegation δ_O is valid, or **False** otherwise.

(6) $\sigma_m \hookleftarrow$ GenerateDVPSign($Params, \mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O, \mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{s}_P, \mathrm{c}_P, \mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D, m$): This is a probabilistic algorithm run by proxy signers. It takes as input $Params$, the identity ID_O and the full public

key (P_O, K_O) of an original signer O, the delegation δ_O issued by O, the identity ID_P and the full public/private key pair $(P_P, K_P)/(s_P, c_P)$ of the proxy signer P specified by the delegation δ_O, the identity ID_D and the full public key (P_D, K_D) of the verifier D designated by the proxy signer P, and a message m. It outputs a signature σ_m of P on the message m.

(7) **True**/**False** $\longleftarrow$ VerifyDVPSign($Params$, ID_O, P_O, K_O, δ_O, ID_P, P_P, K_P, ID_D, P_D, $K_D, s_D, c_D, m, \sigma_m$): This is a deterministic algorithm run by designated verifiers. It takes as input $Params$, the identity ID_O and the full public key (P_O, K_O) of an original signer O, and the delegation δ_O issued by O, the identity ID_P and the full public key (P_P, K_P) of a proxy signer P, the identity ID_D and the full public/private key pair $(P_D, K_D)/(s_D, c_D)$ of a designated verifier D, a message m, and a proxy signature σ_m. It outputs **True** if the delegation δ_O is valid, the proxy signer P is specified in the delegation δ_O, the message m conforms to the delegation δ_O and the proxy signature σ_m is valid, or **False** otherwise.

(8) $\sigma_m^S \longleftarrow$ SimulateDVPSign($Params$, ID_O, P_O, K_O, δ_O, ID_P, P_P, K_P, ID_D, P_D, K_D, s_D, c_D, m): This is a probabilistic algorithm run by any user (a potential designated verifier). It takes as input $Params$, the identity ID_O and the full public key (P_O, K_O) of an original signer O, the delegation δ_O issued by O, the identity ID_P and the full public key (P_P, K_P) of a proxy signer P, the identity ID_D and the full public/private key pair $(P_D, K_D)/(s_D, c_D)$ of a designated verifier D, and a message m. It outputs a simulated proxy signature σ_m^S on the message m in behalf of the proxy signer P.

Remark. In fact, the last algorithm SimulateDVPSign is neither an essential one, nor a component of the definition for a CLDVPS scheme. It is only used to describe and prove a security requirement for SDVS schemes, namely, non-transferability. We can get the definition of the certificateless ordinary designated verifier proxy signature scheme by eliminating the parameters s_D and c_D, i.e., the full private key of the designated verifier D from the input parameters of the algorithm VerifyDVPSign.

We obtained the above definition of a CLDVPS scheme by improving and modifying the definitions of a CLPS scheme in such a way that it can capture the functions of a DVS scheme. When comparing our definition with the previous definitions of a CLPS scheme (e.g., in [4, 11, 45, 60, 63, 70]), we can find some differences.

Main difference is in the 3rd algorithm, GeneratePartialKeys, generating partial keys of any user. In other definitions, it is usually named as Partial-Private-Key-Extract. It takes system parameters, the master secret key and the identity of a user as input, and outputs only a partial private key of the user. However, in our definition, fo llowing what Cheng and Chen [6] has done for the definition of CL-PKC, the public key P_U of the user U is added to its input parameters, and the public value K_U is also outputted from it. It is possible that the input parameter P_U are ignored within the algorithm and the public value K_U of the output is the empty string. Hence, any partial key generation algorithms following the other definitions can be covered by our definition. On the other hand, by adding the public key P_U selected by a user as input of GeneratePartialKeys, our definition can capture the schemes achieving Girault's trust level 3 [13]. This technique has been already discussed in [1]. By including the partial public key K_U into output of GeneratePartialKeys, our definition can explicitly depict the schemes in which a user's partial private key has a public part, e.g., Padhye and Tiwari's CLPS scheme [45].

Another difference is that some algorithms in the traditional definitions of CL-PKC are eliminated in our definition. Firstly, in the same approach as taken by Wan et al. [60], Xu et al. [63], Zhang et al. [70], Du and Wen [11], etc., we eliminate the algorithm Set-Secret-Value generating a user secret value, the algorithm Set-Private-Key generating a user private key, and the algorithm Set-Public-Key generating a user public key, by including the function of Set-Private-Key into other algorithms and combining the functions of Set-Secret-Value and Set-Public-Key with GenerateUserKeys. The detailed advantages of this approach can be referred to [16]. Secondly, we eliminate the algorithm generating a proxy signing key and include the function of the algorithm into the proxy signing algorithm GenerateDVPSign. This makes the our definition more versatile since it is possible that the partial private key c_P and the secret key s_P of the proxy signer, the delegation δ_O, and the others are 'mixed' together in some randomized way during proxy signing.

Final difference is in specifying the input/output of algorithms according to the above modifications on the old definitions. At least, in our definition, partial public keys and identities of users are explicitly added to input parameters of such algorithms as GenerateDelegation, VerifyDelegation, GenerateDVPSign, VerifyDVPSign, etc.

Definition 4. (Correctness) For a CLDVPS scheme $\mathbf{\Sigma}(\lambda)$, any $(\mathit{Params}, s) \longleftarrow$ Setup(1^{λ}), any $\mathrm{ID}_O, \mathrm{ID}_P, \mathrm{ID}_D \in \{0,1\}^*$ and any m, $\mathrm{w}_O \in \{0,1\}^*$, let

$(\mathrm{P}_O, \mathrm{s}_O) \leftarrow$ GenerateUserKeys($Params$, ID_O),

$(\mathrm{P}_P, \mathrm{s}_P) \leftarrow$ GenerateUserKeys($Params$, ID_P),

$(\mathrm{P}_D, \mathrm{s}_D) \leftarrow$ GenerateUserKeys($Params$, ID_D),

$(\mathrm{K}_O, \mathrm{c}_O) \leftarrow$ GeneratePartialKeys($Params$, s, ID_O, P_O),

$(\mathrm{K}_P, \mathrm{c}_P) \leftarrow$ GeneratePartialKeys($Params$, s, ID_P, P_P),

$(\mathrm{K}_D, \mathrm{c}_D) \leftarrow$ GeneratePartialKeys($Params$, s, ID_D, P_D),

$\delta_O \leftarrow$ GenerateDelegation($Params$, $\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \mathrm{s}_O, \mathrm{c}_O, \mathrm{w}_O$),

$\sigma_m \leftarrow$ GenerateDVPSign($Params$, $\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O, \mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{s}_P, \mathrm{c}_P, \mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D, m$),

$\sigma_m^{\mathrm{S}} \leftarrow$ SimulateDVPSign($Params$, $\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O, \mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D, \mathrm{s}_D, \mathrm{c}_D, m$).

If the differences of the following values

Pr[VerifyDelegation($Params$, $\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O$)=**True**]

Pr[VerifyDVPSign($Params$, $\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O, \mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D, \mathrm{s}_D, \mathrm{c}_D, m, \sigma_m$) = **True**],

Pr[VerifyDVPSign($Params$, $\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O, \mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D, \mathrm{s}_D, \mathrm{c}_D, m, \sigma_m^{\mathrm{S}}$) = **True**]

from 1 are all negligible, the CLDVPS scheme $\mathbf{\Sigma}(\lambda)$ is said to be *correct* or to satisfy the *correctness*(*completeness*).

3.2 Security Model

We first define adversaries against CLDVPS schemes and then specify security requirements for them.

As described in Section 2, there are two types of adversaries, types I and II, against a CLDVPS scheme as a cryptographic primitive of CL-PKC. The adversary against a CLDVPS scheme of Definition 3 is defined as follows:

Definition 5. An *adversary* $\mathrm{A}(\lambda)$ against a CLDVPS scheme $\mathbf{\Sigma}(\lambda)$ is a probabilistic algorithm which takes as input 1^{λ} and can access to some of the following oracles (as well as the random oracles if there exists):

(1) $(\mathrm{P}_U, \mathrm{K}_U) \leftarrow \mathrm{O}_{\mathrm{CreateUse}}(\mathrm{ID}_U)$: On input an identity ID_U, it creates a new user U whose identity is ID_U if the user has not yet been created. At the same time, a partial key pair and a pair of public/secret keys of the user U are generated or updated and the new full public key $(\mathrm{P}_U, \mathrm{K}_U)$ is returned.

(2) $\mathrm{s}_U/\perp \leftarrow \mathrm{O}_{\mathrm{GetSecretKey}}(\mathrm{ID}_U, \mathrm{P}_U)$: On input the identity ID_U and the public key P_U of a user U, it stores $(\mathrm{ID}_U, \mathrm{P}_U)$ in the query list **S** and outputs the user's secret key s_U pairing with P_U, if the user U has been created. It outputs $\perp$ otherwise.

(3) $\mathrm{c}_U/\perp \leftarrow \mathrm{O}_{\mathrm{GetPartialKeys}}(\mathrm{ID}_U, \mathrm{K}_U)$: On input the identity ID_U and the partial public key K_U of a user U, it stores $(\mathrm{ID}_U, \mathrm{K}_U)$ in the query list **P** and outputs the user's partial private key c_U pairing with K_U, if there exists the user U. It outputs $\perp$ otherwise. When K_U is the empty string, c_U is the current partial private key of the user U.

(4) $(\mathrm{P}_U, \mathrm{K}_U)/\perp \leftarrow \mathrm{O}_{\mathrm{GetPublicKey}}(\mathrm{ID}_U)$: On input the identity ID_U of a user U, it stores $(\mathrm{ID}_U, \mathrm{K}_U)$ in the query list **P** and outputs the full public key $(\mathrm{P}_U, \mathrm{K}_U)$ currently associated with ID_U, if there exists the user U. It outputs $\perp$ otherwise.

(5) "OK"$/\perp \leftarrow \mathrm{O}_{\mathrm{ReplaceUserKeys}}(\mathrm{ID}_U, \mathrm{P}'_U, \mathrm{s}'_U)$: On input the identity ID_U, a new public key P'_U and a new secret key s'_U of a user U, it stores $(\mathrm{ID}_U, \mathrm{P}'_U)$ in the query list **R**, raplaces the current pair of partial public / private keys of the user U with the new pair $(\mathrm{P}'_U, \mathrm{s}'_U)$ and outputs "OK", if there exists the user U. It outputs $\perp$ otherwise. s'_U may be the empty string.

(6) $\delta_O/\perp \leftarrow \mathrm{O}_{\mathrm{GetDelegation}}(\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \mathrm{w}_O)$: On input the identity ID_O of an original signer O, a full public key $(\mathrm{P}_O, \mathrm{K}_O)$ and a warrant w_O, it outputs the valid delegation δ_O (created newly if there does not exist) on the warrant w_O and stores the tuple $(\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \mathrm{w}_O, \delta_O)$ in the query-answer list **D**, if there exists the user O and $(\mathrm{P}_O, \mathrm{K}_O)$ is the (current or past) full public key of O. It outputs $\perp$ otherwise.

(7) $\sigma_m/\perp \leftarrow \mathrm{O}_{\mathrm{GetDVPSign}}(\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O, \mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D, m)$: On input the identity ID_O and a full public key $(\mathrm{P}_O, \mathrm{K}_O)$ of an original signer O, a delegation δ_O, the identity ID_P and a full public key $(\mathrm{P}_P, \mathrm{K}_P)$ of a proxy signer P, the identity ID_D and a full public key $(\mathrm{P}_D, \mathrm{K}_D)$ of a designed verifier D, and a message m, it outputs the valid proxy signature σ_m on the message m. If there does not exist one of O, P and D, δ_O is not valid, or ID_P and m do not conform to δ_O, it returns $\perp$.

(8) **True**/**False**$/\perp \leftarrow \mathrm{O}_{\mathrm{VerifyDVPSign}}(\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O, \mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D, m, \sigma_m)$: On input the

identity ID_O and a full public key ($\mathrm{P}_O, \mathrm{K}_O$) of an original signer O, a delegation δ_O, the identity ID_P and a full public key ($\mathrm{P}_P, \mathrm{K}_P$) of a proxy signer P, the identity ID_D and a full public key ($\mathrm{P}_D, \mathrm{K}_D$) of a designed veri fier D, and a message m, it returns $\perp$ if there does not exist one of O, P and D. Otherwise, it outputs **True** or **False** depending on whether the delegation δ_O is valid, the proxy signer P is specified in the delegation δ_O, the message m conforms to the delegation δ_O and the proxy signature σ_m is a valid signature on m for D, or not, and stores ($\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O, \mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D, m, \sigma_m$) in the query list $\mathbf{V}$.

An adversary is called a *type I adversary* and denoted by $\mathrm{A}_{\mathrm{I}}(\lambda)$ if it can access all the above oracles but cannot access to the master secret key. An adversary is called a *type II adversary* and denoted by $\mathrm{A}_{\mathrm{II}}(\lambda)$ if it knows the master secret key and partial private keys of all users but can access only 6 oracles of the above oracles except $\mathrm{O}_{\mathrm{GetPartialKeys}}$ and $\mathrm{O}_{\mathrm{ReplaceUserKeys}}$. An adversary is called a *trusted adversary* and denoted by $\mathrm{A}_t(\lambda)$ if it can access all the above oracles and all secret information including the master secret key.

Our definition of oracles for adversaries has some differences from other security models for CLPS, e.g., in [4, 11, 45, 63, 70]. Firstly, the oracle $\mathrm{O}_{\mathrm{CreateUser}}$ can not only create a new user but also update all keys for any existing user. By querying to this oracle, adversaries can trigger the legitimate renewal of all keys for any user. This makes our definition of adversary more practical. Secondly, a user's public key P_U and partial public key K_U are explicitly added to input of $\mathrm{O}_{\mathrm{GetSecretKey}}$ and $\mathrm{O}_{\mathrm{GetPartialKeys}}$, respectively. By this, our definition can capture the actions of adversaries trying to find something from the system's history in the certificateless settings with freely revocable keys. Thirdly, as a consequence of our definition of CLDVPS scheme excluding the algorithm generating a proxy signing key, we eliminate the oracle returning a proxy signing key unlike other models. Actually, it is not necessary for adversaries with having access to $\mathrm{O}_{\mathrm{GetSecretKey}}$ and $\mathrm{O}_{\mathrm{GetPartialKeys}}$. Finally, the specifications of the input/output of oracles are updated and the oracle $\mathrm{O}_{\mathrm{VerifyDVPSign}}$ is added to capture the requirement of DVS.

We consider that a secure CLDVPS scheme, as a strong DVS scheme and a proxy signature scheme, should satisfy the strong unforgeability, the non-transferability, the privacy of signer's identity, non-delegatability and the prevention of misuse property besides the correctness and the strongness. Since the verifiability, the strong identifiability, the strong undeniability and the distinguishability are conflict with the non-transferability, they are excluded from the security requirements for a secure CLDVPS scheme. Next, we present the formal definitions of strong unforgeability, the non-transferability, the privacy of signer's identity and non-delegatability in turn.

Definition 6 (Strong Unforgeability). Let us consider the following interactive game between an adversary $\mathrm{A}(\lambda) \in \{ \mathrm{A}_{\mathrm{I}}(\lambda), \mathrm{A}_{\mathrm{II}}(\lambda) \}$ against a CLDVPS scheme $\boldsymbol{\Sigma}(\lambda)$ and a challenger X.

sEUF Game:

Setup: X runs Setup(1^{λ}) to obtain the system parameter list $Params$ and the master secret key s. Then X sends $Params$ to the adversary $\mathrm{A}(\lambda)$. If $\mathrm{A}(\lambda)$ is $\mathrm{A}_{\mathrm{II}}(\lambda)$, X also passes s to the adversary $\mathrm{A}(\lambda)$. Otherwise, keeps the master secret key s secret. Finally, X initializes the lists $\mathbf{P}$, $\mathbf{S}$, $\mathbf{R}$ and $\mathbf{D}$ with empty list ($\emptyset$) respectively.

Attack: $\mathrm{A}(\lambda)$ gathers information by querying oracles allowed to him adaptively. X correctly simulates the oracles called by $\mathrm{A}(\lambda)$ and returns proper values.

Forgery: Eventually, $\mathrm{A}(\lambda)$ outputs a tuple ($\mathrm{ID}_O^*, \mathrm{P}_O^*, \mathrm{K}_O^*, \delta_O^*, \mathrm{ID}_P^*, \mathrm{P}_P^*, \mathrm{K}_P^*, \mathrm{ID}_D^*, \mathrm{P}_D^*, \mathrm{K}_D^*, m^*, \sigma_{m^*}$).

We say that $\mathrm{A}(\lambda)$ *wins* the game or *succeeds* in the game if the following conditions are all satisfied:

(1) VerifyDVPSign($Params, \mathrm{ID}_O^*, \mathrm{P}_O^*, \mathrm{K}_O^*, \delta_O^*, \mathrm{ID}_P^*, \mathrm{P}_P^*, \mathrm{K}_P^*, \mathrm{ID}_D^*, \mathrm{P}_D^*, \mathrm{K}_D^*, \mathrm{s}_D^*, \mathrm{c}_D^*, m^*, \sigma_{m^*}$)=**True**, where s_D^* and c_D^* are the secret key and the partial private key of the designated verifier with identity ID_D^*, respectively.

(2) The tuple ($\mathrm{ID}_O^*, \mathrm{P}_O^*, \mathrm{K}_O^*, \delta_O^*, \mathrm{ID}_P^*, \mathrm{P}_P^*, \mathrm{K}_P^*, \mathrm{ID}_D^*, \mathrm{P}_D^*, \mathrm{K}_D^*, m^*$) has not been submitted to the oracle $\mathrm{O}_{\mathrm{GetDVPSign}}$ in the *Attack* stage.

(3) When $\mathrm{A}(\lambda)$ is $\mathrm{A}_{\mathrm{I}}(\lambda)$, $(\mathrm{ID}_D^*, \mathrm{K}_D^*) \notin \mathbf{P}$ or $(\mathrm{ID}_D^*, \mathrm{P}_D^*) \notin \mathbf{S} \cup \mathbf{R}$. When $\mathrm{A}(\lambda)$ is $\mathrm{A}_{\mathrm{II}}(\lambda)$, $(\mathrm{ID}_D^*, \mathrm{P}_D^*) \notin \mathbf{S}$.

(4) When $\mathrm{A}(\lambda)$ is $\mathrm{A}_{\mathrm{I}}(\lambda)$, the following boolean expression is true:

$$(((\mathrm{ID}_O^*, \mathrm{K}_O^*) \notin \mathbf{P} \vee (\mathrm{ID}_O^*, \mathrm{P}_O^*) \notin \mathbf{S} \cup \mathbf{R}) \wedge (\mathrm{ID}_O^*, \mathrm{P}_O^*, \mathrm{K}_O^*, \mathrm{w}_O^*, *) \notin \mathbf{D}) \vee$$

$$((\mathrm{ID}_P^*, \mathrm{K}_P^*) \notin \mathbf{P} \vee (\mathrm{ID}_P^*, \mathrm{P}_P^*) \notin \mathbf{S} \cup \mathbf{R}),$$

where w_O^* is the warrant in the delegation δ_O^* and * stands for an arbitrary value.

When $\mathrm{A}(\lambda)$ is $\mathrm{A}_{\mathrm{II}}(\lambda)$, the following boolean expression is true:

$$((\mathrm{ID}_O^*, \mathrm{P}_O^*) \notin \mathbf{S} \wedge (\mathrm{ID}_O^*, \mathrm{P}_O^*, \mathrm{K}_O^*, \mathrm{w}_O^*, *) \notin \mathbf{D}) \vee ((\mathrm{ID}_P^*, \mathrm{P}_P^*) \notin \mathbf{S}).$$

A CLDVPS scheme $\boldsymbol{\Sigma}(\lambda)$ is said to be *strong existentially unforgeable against an adaptively chosen-message attack and an adaptively chosen-warrant attack* or to satisfy *Strong Unforgeability* if the probability of success of any PPT adversary $\mathrm{A}(\lambda)$ against the scheme in the above game is negligible in λ.

An adversary against a CLDVPS scheme can win the above game by forging a valid proxy signature or a valid delegation. A valid proxy signature cannot be generated by anyone including the original signer, except for the proxy signer and the designated verifier. According to the success condition (2) of an adversary in the above game, he can request a proxy signature on the challenge message m^* to the oracle $\mathrm{O}_{\mathrm{GetDVPSign}}$ as long as one of the original signer, the proxy signer and the designated signer of the signature is not a target user (i.e., ID_O^*, ID_P^*, or ID_D^*, correspondingly) or the delegation is not equal to δ_O^*. This is just the reason that the unforgeabililty of the above definition is strong.

Next, we define the non-transferability for CLDVPS, inspired from the works of Emura et al. [12], Huang et al. [20], Yoneyama et al. [67], Hu et al. [17], Rastegari et al. [49].

Definition 7 (Non-transferability). Let us consider the following interactive game between a trusted adversary (distinguisher) $\mathrm{A}_t(\lambda)$ against a CLDVPS scheme $\boldsymbol{\Sigma}(\lambda)$ and a challenger X.

NT Game:

Setup: X runs Setup(1^λ) to obtain the system parameter list $Params$ and the master secret key s. Then X sends $Params$ and s to the adversary $\mathrm{A}_t(\lambda)$.

Challenge: $\mathrm{A}_t(\lambda)$ gathers information by querying oracles adaptively. X correctly simulates the oracles called by $\mathrm{A}_t(\lambda)$ and returns proper values. Eventually, $\mathrm{A}_t(\lambda)$ submits to X a challenge message m^* and the identities $\{\mathrm{ID}_O^*, \mathrm{ID}_P^*, \mathrm{ID}_D^*\}$ of an original singer, a proxy signer and a designated verifier selected as targets along with a valid delegation δ_O^* and the public keys P_O^*, K_O^*, P_P^*, K_P^*, P_D^*, K_D^*. Then X flips a fair coin $b \leftarrow_{\mathrm{R}} \{0,1\}$, produces a signature σ^*=GenerateDVPSign($Params$, ID_O^*, P_O^*, K_O^*, δ_O^*, ID_P^*, P_P^*, K_P^*, s_P^*, c_P^*, ID_D^*, P_D^*, K_D^*, m^*) if b =0 or σ^* = SimulateDVPSign($Params$, ID_O^*, P_O^*, K_O^*, δ_O^*, ID_P^*, P_P^*, K_P^*, ID_D^*, P_D^*, K_D^*, s_D^*, c_D^*, m^*) if b =1 and replies to $\mathrm{A}_t(\lambda)$ with σ^*.

Guess: After receiving σ^*, $\mathrm{A}_t(\lambda)$ can continue to issue oracle queries. In the end, $\mathrm{A}_t(\lambda)$ returns his guess

$b' \in \{0,1\}$.

We say that $\mathrm{A}_t(\lambda)$ *wins* the game or *succeeds* in the game if $b' = b$, and define the *advantage* of $\mathrm{A}_t(\lambda)$ in the game as |Pr[$\mathrm{A}_t(\lambda)$ wins the NT game] – 1/2|.

A CLDVPS scheme $\mathbf{\Sigma}(\lambda)$ is said to be *non-transferable* or to satisfy *Non-transferability* if the advantage of any PPT trusted adversary $\mathrm{A}_t(\lambda)$ against the scheme in the above game is negligible in λ.

The following definition of the privacy of signer's identity (PSI) for CLDVPS are inspired from the works of Tian et al. [57] and Damgård et al. [9].

Definition 8 (Privacy of signer's identity, PSI). Let us consider the following interactive game between a trusted adversary (distinguisher) $\mathrm{A}_t(\lambda)$ against a CLDVPS scheme $\mathbf{\Sigma}(\lambda)$ and a challenger X.

PSI Game:

Setup: X runs Setup(1^{λ}) to obtain the system parameter list $Params$ and the master secret key s. Then X sends $Params$ and s to the adversary $\mathrm{A}_t(\lambda)$. X initializes the lists $\mathbf{P}$, $\mathbf{S}$, $\mathbf{R}$, $\mathbf{D}$ and $\mathbf{V}$ with empty list (∅) respectively.

Challenge: $\mathrm{A}_t(\lambda)$ gathers information by querying oracles adaptively. X correctly simulates the oracles called by $\mathrm{A}_t(\lambda)$ and returns proper values. Eventually, $\mathrm{A}_t(\lambda)$ submits to X a challenge message m^* and the identities { $\mathrm{ID}^*_{O_1}, \mathrm{ID}^*_{P_1}, \mathrm{ID}^*_{O_2}, \mathrm{ID}^*_{P_2}, \mathrm{ID}^*_{D}$ } of two original singer, two proxy signer and a designated verifier selecte d as targets along with the corresponding delegations $\delta^*_{O_1}$, $\delta^*_{O_2}$ and public keys $\mathrm{P}^*_{O_1}, \mathrm{K}^*_{O_1}, \mathrm{P}^*_{P_1}, \mathrm{K}^*_{P_1}, \mathrm{P}^*_{O_2}, \mathrm{K}^*_{O_2}$, $\mathrm{P}^*_{P_2}$, $\mathrm{K}^*_{P_2}$, P^*_{D}, K^*_{D}. Then X flips a fair coin $b \leftarrow_{\mathrm{R}}$ {0,1}, produces a signature σ^* = GenerateDVPSign($Params, \mathrm{ID}^*_{O_b}, \mathrm{P}^*_{O_b}, \mathrm{K}^*_{O_b}, \delta^*_{O_b}$, $\mathrm{ID}^*_{P_b}$, $\mathrm{P}^*_{P_b}$, $\mathrm{K}^*_{P_b}$, $\mathrm{s}^*_{P_b}$, $\mathrm{c}^*_{P_b}, \mathrm{ID}^*_{D}, \mathrm{P}^*_{D}, \mathrm{K}^*_{D}, m^*$) and replies to $\mathrm{A}_t(\lambda)$ with σ^*.

Guess: After receiving σ^*, $\mathrm{A}_t(\lambda)$ can continue to issue oracle queries. In the end, $\mathrm{A}_t(\lambda)$ returns his guess $b' \in \{0,1\}$.

We say that $\mathrm{A}_t(\lambda)$ *wins* the game or *succeeds* in the game if $b' = b$ and the following boolean expressions are all true:

(1) $((\mathrm{ID}^*_{D}, \mathrm{K}^*_{D}) \notin \mathbf{P} \vee (\mathrm{ID}^*_{D}, \mathrm{P}^*_{D}) \notin \mathbf{S} \cup \mathbf{R})$,

(2) $(\mathrm{ID}^*_{O_1}, \mathrm{P}^*_{O_1}, \mathrm{K}^*_{O_1}, \delta^*_{O_1}, \mathrm{ID}^*_{P_1}, \mathrm{P}^*_{P_1}, \mathrm{K}^*_{P_1}, \mathrm{ID}^*_D, \mathrm{P}^*_D, \mathrm{K}^*_D, m^*, \sigma^*) \notin \mathbf{V}$ $\wedge$

$(\mathrm{ID}^*_{O_2}, \mathrm{P}^*_{O_2}, \mathrm{K}^*_{O_2}, \delta^*_{O_2}, \mathrm{ID}^*_{P_2}, \mathrm{K}^*_{O_2}, \mathrm{P}^*_{P_2}, \mathrm{ID}^*_D, \mathrm{P}^*_D, \mathrm{K}^*_D, m^*, \sigma^*) \notin \mathbf{V}$.

We define the *advantage* of $\mathrm{A}_t(\lambda)$ in the game as |Pr[$\mathrm{A}_t(\lambda)$ wins the PSI game] – 1/2|.

A CLDVPS scheme $\mathbf{\Sigma}(\lambda)$ is said to satisfy *Privacy of signer's identity* if the advantage of any PPT trusted adversary $\mathrm{A}_t(\lambda)$ against the scheme in the above game is negligible in λ.

Inspired from the works of Huang et al. [20], Tian et al. [57] and Hu et al. [17] in which employ some similar definitions of the non-delegatability to one proposed by Lipmaa et.al [40], and the work of Rastegari et al. [49] using a relaxed version by Tian et al. [55], we define the non-delegatability for CLDVPS as follows.

Definition 9 (Non-delegatability). Let us consider the following interactive game between an adversary $\mathrm{A}(\lambda) \in \{ \mathrm{A}_{\mathrm{I}}(\lambda), \mathrm{A}_{\mathrm{II}}(\lambda) \}$ against a CLDVPS scheme $\mathbf{\Sigma}(\lambda)$ and a challenger X.

NDS Game:

Setup: X runs Setup(1^λ) to obtain the system parameter list $Params$ and the master secret key s. X sends $Params$ to the adversary $\mathrm{A}(\lambda)$. If $\mathrm{A}(\lambda)$ is $\mathrm{A}_{\mathrm{II}}(\lambda)$, X also passes s to the adversary $\mathrm{A}(\lambda)$. Finally, X initializes the lists $\mathbf{P}$, $\mathbf{S}$, $\mathbf{R}$, $\mathbf{D}$ and $\mathbf{V}$ with empty list (∅) respectively.

Select: $\mathrm{A}(\lambda)$ gathers information by querying oracles allowed to him adaptively. X correctly simulates the oracles called by $\mathrm{A}(\lambda)$ and returns proper values. Eventually, $\mathrm{A}(\lambda)$ submits to X the identity ID^*_P and full public key $(\mathrm{P}^*_P, \mathrm{K}^*_P)$ (or ID^*_D, $\mathrm{P}^*_D, \mathrm{K}^*_D$) of a proxy signer (or a designated verifier) selected as targets, which satisfy that $(\mathrm{ID}^*_P, \mathrm{K}^*_P) \notin \mathbf{P}$ or $(\mathrm{ID}^*_P, \mathrm{P}^*_P) \notin \mathbf{S} \cup \mathbf{R}$ (or, $(\mathrm{ID}^*_D, \mathrm{K}^*_D) \notin \mathbf{P}$ or $(\mathrm{ID}^*_D, \mathrm{P}^*_D) \notin \mathbf{S} \cup \mathbf{R}$) when $\mathrm{A}(\lambda)$ is $\mathrm{A}_{\mathrm{I}}(\lambda)$, and $(\mathrm{ID}^*_P, \mathrm{P}^*_P) \notin \mathbf{S}$ (or $(\mathrm{ID}^*_D, \mathrm{P}^*_D) \notin \mathbf{S}$) when $\mathrm{A}(\lambda)$ is $\mathrm{A}_{\mathrm{II}}(\lambda)$.

Challenge: X sends to $\mathrm{A}(\lambda)$ a challenge identity ID^*_D and full public key $(\mathrm{P}^*_D, \mathrm{K}^*_D)$ (or ID^*_P, $\mathrm{P}^*_P, \mathrm{K}^*_P$) of a designated verifier (or a proxy signer), an identity ID^*_O and full public key $(\mathrm{P}^*_O, \mathrm{K}^*_O)$ of an original signer, a valid delegation δ^*_O and a challenge message m^*, and asks $\mathrm{A}(\lambda)$ to submit a valid signature on $(\mathrm{ID}^*_O, \mathrm{P}^*_O, \mathrm{K}^*_O, \delta^*_O, \mathrm{ID}^*_P, \mathrm{P}^*_P, \mathrm{K}^*_P, \mathrm{ID}^*_D, \mathrm{P}^*_D, \mathrm{K}^*_D, m^*)$ to X. $\mathrm{A}(\lambda)$ can issue queries to the oracles allowed to him adaptively, but cannot query the tuple $(\mathrm{ID}^*_O, \mathrm{P}^*_O, \mathrm{K}^*_O, \delta^*_O, \mathrm{ID}^*_P, \mathrm{P}^*_P, \mathrm{K}^*_P, \mathrm{ID}^*_D, \mathrm{P}^*_D, \mathrm{K}^*_D, m^*)$ to the oracle

$\mathrm{O}_{\mathrm{GetDVPSign}}$ and cannot query to the oracle $\mathrm{O}_{\mathrm{VerifyDVPSign}}$ on input ID_D^*, $\mathrm{P}_D^*, \mathrm{K}_D^*$ if a designated verifier has been selected by $\mathrm{A}(\lambda)$ in the Select stage. X simulates the oracles called by $\mathrm{A}(\lambda)$. Eventually, $\mathrm{A}(\lambda)$ submits a valid signature (or a valid simulated signature) on $(\mathrm{ID}_O^*, \mathrm{P}_O^*, \mathrm{K}_O^*, \delta_O^*, \mathrm{ID}_P^*, \mathrm{P}_P^*, \mathrm{K}_P^*, \mathrm{ID}_D^*, \mathrm{P}_D^*, \mathrm{K}_D^*, m^*)$ to X if the following boolean expression is true:

$$((\mathrm{ID}_D^*, \mathrm{K}_D^*) \notin \mathbf{P} \vee (\mathrm{ID}_D^*, \mathrm{P}_D^*) \notin \mathbf{S} \cup \mathbf{R}) \wedge ((\mathrm{ID}_P^*, \mathrm{K}_P^*) \notin \mathbf{P} \vee (\mathrm{ID}_P^*, \mathrm{P}_P^*) \notin \mathbf{S} \cup \mathbf{R}),$$

when $\mathrm{A}(\lambda)$ is $\mathrm{A}_{\mathrm{I}}(\lambda)$, and the following boolean expression is true:

$$(\mathrm{ID}_D^*, \mathrm{P}_D^*) \notin \mathbf{S} \wedge (\mathrm{ID}_P^*, \mathrm{P}_P^*) \notin \mathbf{S},$$

when $\mathrm{A}(\lambda)$ is $\mathrm{A}_{\mathrm{II}}(\lambda)$.

X can repeat the steps in this stage and gather valid signatures enough for the next stage.

Extract: X tries to compute correctly the full private key $(\mathrm{s}_P^*, \mathrm{c}_P^*)$(or $\mathrm{s}_D^*, \mathrm{c}_D^*$) of the proxy signer (or the designated verifier) selected by $\mathrm{A}(\lambda)$ in the Select stage, using only his replies to the queries of $\mathrm{A}(\lambda)$ in the previous stages and the valid signatures returned by $\mathrm{A}(\lambda)$ in the Challenge stage.

A CLDVPS scheme $\mathbf{\Sigma}(\lambda)$ is said to satisfy *Non-delegatability* if there exists a PPT (in λ) algorithm X that can succeed in the above game with a non-negligible probability (in λ) against any PPT (in λ) adversary $\mathrm{A}(\lambda)$ which can generate a valid signature with a non-negligible probability (in λ) in the Challenge stage.

As Tian et al. [56] mentioned, although the non-delegatability property has been a subject of many literatures, it is a controversial concept as it may be undesirable in some applications, so that we will no longer discuss the non-delegatability in this work.

4. Our CLDVPS scheme

In this section, we propose a CLDVPS scheme using bilinear pairings in conformity with Definition 3, and consider the correctness and the efficiency of it.

Our scheme consists of the following eight algorithms:

(1) $(Params, s) \longleftarrow \mathsf{Setup}(1^\lambda)$: KGC takes the security parameter $\lambda \in \mathrm{N}$ as input and returns $Params$ and a master secret key s as follows:

a. Chooses a λ-bit prime p, and determines a finite field Φ_p of order p, an elliptic curve E/Φ_p defined by $E: Y^2 = X^3 + aX + b$ over Φ_p, a cyclic additive subgroup Γ_1 of prime order $q \geq 2^\lambda$ on E/Φ_p, a generator G of Γ_1, a cyclic multiplicative group Γ' of the same order q (e.g., Z_q^*) and an admissible bilinear map $\hat{e}: \Gamma_1 \times \Gamma_1 \to \Gamma'$.

b. $s \leftarrow_{\mathrm{R}} \mathrm{Z}_q^*$; $P_{\mathrm{KGC}} \leftarrow [s]G$.

c. Chooses three cryptographic hash functions $\mathrm{H}_1: \{0,1\}^* \to \mathrm{Z}_q^*$, $\mathrm{H}_2: \{0,1\}^* \to \Gamma_1$, $\mathrm{H}_3: \{0,1\}^* \to \Gamma_1$.

d. $B_{Params} := a \parallel b \parallel G \parallel P_{\mathrm{KGC}}$. (We assume that the binary representaions of a, b, G and P_{KGC} are obtained in a trivial way.)

e. Publishes $Params := \{a, b, p, q, G, P_{\mathrm{KGC}}, \mathrm{H}_1, \mathrm{H}_2, \mathrm{H}_3\}$ and keeps s secret

(2) $(\mathrm{P}_U, \mathrm{s}_U) \leftarrow$ GenerateUserKeys($Params$, ID_U): The user U with the identity ID_U sets his public key P_U and secret key s_U as follows:

a. $\mathrm{s}_U \leftarrow_{\mathrm{R}} \mathrm{Z}_q^*$; $\mathrm{P}_U \leftarrow [\mathrm{s}_U]G$; **return** $(\mathrm{P}_U, \mathrm{s}_U)$.

(3) $(\mathrm{K}_U, \mathrm{c}_U) \leftarrow$ GeneratePartialKeys($Params$, s, ID_U, P_U): On input $Params$, the master secret key s, a user's identity ID_U and public key P_U, KGC returns a pair of partial public / private keys $(\mathrm{K}_U, \mathrm{c}_U)$ for the user U with the identity ID_U as follows:

a. $h_U \leftarrow \mathrm{H}_1(B_{Params} \parallel \mathrm{ID}_U)$.

b. $\mathrm{k}_U \leftarrow_{\mathrm{R}} \mathrm{Z}_q^*$; $\mathrm{K}_U \leftarrow [\mathrm{k}_U]G$.

c. $\mathrm{c}_U \leftarrow (\mathrm{k}_U + s \cdot h_U) \bmod q$.

d. **return** $(\mathrm{K}_U, \mathrm{c}_U)$. (Publishes K_U and sends c_U to U via a secure channel.)

(U can validate its partial keys $(\mathrm{K}_U, \mathrm{c}_U)$ by checking the equation:

$$[\mathrm{c}_U]G \stackrel{?}{=} \mathrm{K}_U + [\mathrm{H}_1(B_{Params} \parallel \mathrm{ID}_U)]\, P_{\mathrm{KGC}}.)$$

(4) $\delta_O \leftarrow$ GenerateDelegation($Params$, ID_O, P_O, K_O, s_O, c_O, w_O): The original signer O outputs the delegation δ_O on the warrant w_O as follows:

a. $\mathrm{t}_O \leftarrow (\mathrm{s}_O + \mathrm{c}_O) \bmod q$.

b. $r_O \leftarrow_{\mathrm{R}} \mathrm{Z}_q^*$; $R_O \leftarrow [r_O]G$.

c. $H_2 \longleftarrow \mathrm{H}_2(\mathrm{w}_O \,\|\, \mathrm{ID}_O \,\|\, \mathrm{P}_O \,\|\, \mathrm{K}_O)$. (We assume that the binary representaions of P_O and K_O are obtained in a trivial way.)

d. $\Delta_O \longleftarrow [r_O]P_{\mathrm{KGC}} + [\mathrm{t}_O]H_2$.

e. $\delta_O := (\mathrm{w}_O, R_O, \Delta_O)$; **return** δ_O.

(5) **True**/**False** $\longleftarrow$ VerifyDelegation($Params, \mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O$): Any verifier, including the proxy signer specified by the delegation δ_O, can verify δ_O as follows:

a. $h_O \longleftarrow \mathrm{H}_1(B_{Params} \,\|\, \mathrm{ID}_O)$.

b. $\mathrm{T}_O \longleftarrow \mathrm{P}_O + \mathrm{K}_O + [h_O]P_{\mathrm{KGC}}$.

c. $(\mathrm{w}_O, R_O, \Delta_O) := \delta_O$. ($\delta_O$ is parsed as $(\mathrm{w}_O, R_O, \Delta_O)$.)

d. $H_2 \longleftarrow \mathrm{H}_2(\mathrm{w}_O \,\|\, \mathrm{ID}_O \,\|\, \mathrm{P}_O \,\|\, \mathrm{K}_O)$.

e. **return** $(\hat{e}(\Delta_O, G) \overset{?}{=} \hat{e}(P_{\mathrm{KGC}}, R_O) \cdot \hat{e}(H_2, \mathrm{T}_O))$.

(A verifier accepts δ_O if VerifyDelegation returns **True**, or rejects otherwise.)

(6) $\sigma_m \longleftarrow$ GenerateDVPSign($Params, \mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O, \mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{s}_P, \mathrm{c}_P, \mathrm{ID}_D, \mathrm{P}_D$, K_D, m): If the proxy signer P specified by the delegation δ_O accepts the delegation δ_O, he returns a proxy signature σ_m on the message m as follows:

a. $\mathrm{t}_P \longleftarrow (\mathrm{s}_P + \mathrm{c}_P) \bmod q$.

b: $h_D \longleftarrow \mathrm{H}_1(B_{Params} \,\|\, \mathrm{ID}_D)$.

c: $\mathrm{T}_D \longleftarrow \mathrm{P}_D + \mathrm{K}_D + [h_D]P_{\mathrm{KGC}}$.

d. $r_P \longleftarrow_{\mathrm{R}} \mathrm{Z}_q^*$; $R_P \longleftarrow [r_P]G$.

e. $H_3 \longleftarrow \mathrm{H}_3(m \,\|\, \delta_O \,\|\, \mathrm{ID}_P \,\|\, \mathrm{P}_P \,\|\, \mathrm{K}_P \,\|\, \mathrm{ID}_D \,\|\, \mathrm{P}_D \,\|\, \mathrm{K}_D)$. (We assume that the binary representaions of $\delta_O, \mathrm{P}_P, \mathrm{K}_P, \mathrm{P}_D$ and K_D are obtained in a trivial way.)

f. $v_m \longleftarrow \hat{e}([r_P]P_{\mathrm{KGC}} + [\mathrm{t}_P]H_3, \mathrm{T}_D)$.

g. $\sigma_m := (R_P, v_m)$; **return** σ_m.

(7) **True**/**False** $\longleftarrow$ VerifyDVPSign($Params, \mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O, \mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{ID}_D, \mathrm{P}_D$, $\mathrm{K}_D, \mathrm{s}_D, \mathrm{c}_D$, m, σ_m): Having all inputs, the designated verifier D verifies the proxy signature σ_m on the message m as

follows:

a. **if** VerifyDelegation($Params$, ID_O , P_O , K_O , δ_O)=**False then return False**.

b. **if** (the message m does not conform to the delegation δ_O) **then return False**.

c. $\mathrm{t}_D \longleftarrow (\mathrm{s}_D + \mathrm{c}_D) \bmod q$.

d: $h_P \longleftarrow \mathrm{H}_1 (B_{Params} \| \mathrm{ID}_P)$.

e: $\mathrm{T}_P \longleftarrow \mathrm{P}_P + \mathrm{K}_P + [h_P] P_{\mathrm{KGC}}$.

f. $(R_P , v_m) := \sigma_m$. (σ_m is parsed as (R_P , v_m).)

g. $H_3 \longleftarrow \mathrm{H}_3 (m \| \delta_O \| \mathrm{ID}_P \| \mathrm{P}_P \| \mathrm{K}_P \| \mathrm{ID}_D \| \mathrm{P}_D \| \mathrm{K}_D)$.

h. **return** ($v_m \overset{?}{=} \hat{e} (R_P , P_{\mathrm{KGC}})^{\mathrm{t}_D} \cdot \hat{e} (H_3 , \mathrm{T}_P)^{\mathrm{t}_D}$).

(The designated verifier D accepts σ_m if VerifyDVPSign returns True, or rejects otherwise.

(8) $\sigma_m^{\mathrm{S}} \hookleftarrow$ SimulateDVPSign($Params$, ID_O , P_O , K_O , δ_O , ID_P , P_P , K_P , ID_D , P_D , K_D , s_D , c_D , m):

Having all inputs, the designated verifier D returns a simulated signature σ_m^{S} on the message m as follows:

a. $\mathrm{t}_D \longleftarrow (\mathrm{s}_D + \mathrm{c}_D) \bmod q$.

b: $h_P \longleftarrow \mathrm{H}_1 (B_{Params} \| \mathrm{ID}_P)$.

c: $\mathrm{T}_P \longleftarrow \mathrm{P}_P + \mathrm{K}_P + [h_P] P_{\mathrm{KGC}}$.

d. $r_D \leftarrow_{\mathrm{R}} Z_q^*$; $R_D \longleftarrow [r_D] G$.

e. $H_3 \longleftarrow \mathrm{H}_3 (m \| \delta_O \| \mathrm{ID}_P \| \mathrm{P}_P \| \mathrm{K}_P \| \mathrm{ID}_D \| \mathrm{P}_D \| \mathrm{K}_D)$.

f. $v_m \longleftarrow \hat{e} (R_D , P_{\mathrm{KGC}})^{\mathrm{t}_D} \cdot \hat{e} (H_3 , \mathrm{T}_P)^{\mathrm{t}_D}$.

g. $\sigma_m^{\mathrm{S}} := (R_D , v_m)$; **return** σ_m^{S}.

Note that using the binding technique in [1], we can easily convert our scheme to the one which achieves Girault's trust level 3 [13], by concatenating P_U , P_O , P_D or P_P to the inputs of H_1 in the step (a) of GeneratePartialKeys, the step (a) of VerifyDelegation, the step (b) of GenerateDVPSign, the step (d) of VerifyDVPSign and the step (b) of SimulateDVPSign respectively.

Now, we consider the correctness of the proposed scheme.

The correctness of the algorithm VerifyDelegation verifying a delegation is proved as follows:

$$\begin{aligned}\hat{e}(\Delta_O, G) &= \hat{e}([r_O]P_{\mathrm{KGC}} + [\mathrm{t}_O]H_2, G)\\ &= \hat{e}([r_O]P_{\mathrm{KGC}}, G)\cdot\hat{e}([\mathrm{t}_O]H_2, G)\\ &= \hat{e}(P_{\mathrm{KGC}}, [r_O]G)\cdot\hat{e}(H_2, [\mathrm{t}_O]G)\\ &= \hat{e}(P_{\mathrm{KGC}}, R_O)\cdot\hat{e}(H_2, [(\mathrm{s}_O + \mathrm{c}_O)\bmod q]G)\\ &= \hat{e}(P_{\mathrm{KGC}}, R_O)\cdot\hat{e}(H_2, [\mathrm{s}_O]G + [\mathrm{c}_O]G)\\ &= \hat{e}(P_{\mathrm{KGC}}, R_O)\cdot\hat{e}(H_2, \mathrm{P}_O + [(\mathrm{k}_O + s\cdot h_O)\bmod q]G)\\ &= \hat{e}(P_{\mathrm{KGC}}, R_O)\cdot\hat{e}(H_2, \mathrm{P}_O + \mathrm{K}_O + [(s\cdot h_O)\bmod q]G)\\ &= \hat{e}(P_{\mathrm{KGC}}, R_O)\cdot\hat{e}(H_2, \mathrm{P}_O + \mathrm{K}_O + [h_O]P_{\mathrm{KGC}}).\end{aligned}$$

The correctness of the algorithm VerifyDVPSign verifying a proxy signature is proved as follows:

$$\begin{aligned}v_m &= \hat{e}([r_P]P_{\mathrm{KGC}} + [\mathrm{t}_P]H_3, \mathrm{T}_D)\\ &= \hat{e}([r_P]P_{\mathrm{KGC}}, [\mathrm{t}_D]G)\cdot\hat{e}([\mathrm{t}_P]H_3, [\mathrm{t}_D]G)\\ &= \hat{e}([r_P]P_{\mathrm{KGC}}, G)^{\mathrm{t}_D}\cdot\hat{e}([\mathrm{t}_P]H_3, G)^{\mathrm{t}_D}\\ &= \hat{e}(P_{\mathrm{KGC}}, [r_P]G)^{\mathrm{t}_D}\cdot\hat{e}(H_3, [\mathrm{t}_P]G)^{\mathrm{t}_D}\\ &= \hat{e}(P_{\mathrm{KGC}}, R_P)^{\mathrm{t}_D}\cdot\hat{e}(H_3, \mathrm{T}_P)^{\mathrm{t}_D}.\end{aligned}$$

Next, we compare the efficiency of our scheme with some existed DVS schemes in Table 1 where $H_{\mathbf{T}}$, $P_{\mathbf{T}}$, $S_{\mathbf{T}}$ and $A_{\mathbf{T}}$ stand for the time of a map-to-point hash operation, a bilinear pairing operation, an elliptic curve scalar multiplication and an elliptic curve point addition respectively, G_ℓ denotes the bit length of a point in the elliptic curve group Γ_1, and Z_ℓ means the length of the integer q, i.e., $\log q$. According to [7, 32], $H_{\mathbf{T}}$ is rather l ogner than $P_{\mathbf{T}}$ by a narrow margin, $P_{\mathbf{T}}$ is several times longer than $S_{\mathbf{T}}$, and $S_{\mathbf{T}}$ is several tens to hundreds times longer than $A_{\mathbf{T}}$. We ignore costless operations sush as one-way hash operation and operations on integers and the multiplicative group Γ', which are much cheaper than the point addition. G_ℓ is several times longer than Z_ℓ.

Table 1 shows that the efficiency of our scheme is not inferior to the schemes in [17, 5].

Table 1. Comparison of the efficiency

	[18] IBDVPS	[17] DVPS	[5] CLDVS	Our scheme CLDVPS
Generation of keys	$1H_{\mathbf{T}}+1S_{\mathbf{T}}$	$1S_{\mathbf{T}}$	$1H_{\mathbf{T}}+2S_{\mathbf{T}}$	$2S_{\mathbf{T}}$
Generation of delegation	$4H_{\mathbf{T}}+1S_{\mathbf{T}}+1A_{\mathbf{T}}$	$1H_{\mathbf{T}}+1S_{\mathbf{T}}$	$-$	$1H_{\mathbf{T}}+3S_{\mathbf{T}}+1A_{\mathbf{T}}$
Verification of delegation	$2H_{\mathbf{T}}+2P_{\mathbf{T}}+1A_{\mathbf{T}}$	$1H_{\mathbf{T}}+2P_{\mathbf{T}}$	$-$	$1H_{\mathbf{T}}+3P_{\mathbf{T}}+1S_{\mathbf{T}}+2A_{\mathbf{T}}$
Signing	$1P_{\mathbf{T}}$	$1H_{\mathbf{T}}+1P_{\mathbf{T}}+3S_{\mathbf{T}}+1A_{\mathbf{T}}$	$2H_{\mathbf{T}}+1P_{\mathbf{T}}+3S_{\mathbf{T}}+1A_{\mathbf{T}}$	$1H_{\mathbf{T}}+1P_{\mathbf{T}}+4S_{\mathbf{T}}+3A_{\mathbf{T}}$
Verification of signature	$1P_{\mathbf{T}}$	$2H_{\mathbf{T}}+2P_{\mathbf{T}}+2S_{\mathbf{T}}$	$3H_{\mathbf{T}}+1P_{\mathbf{T}}+3S_{\mathbf{T}}+1A_{\mathbf{T}}$	$1H_{\mathbf{T}}+2P_{\mathbf{T}}+3S_{\mathbf{T}}+2A_{\mathbf{T}}$
Key size	$1G_{\ell}+1Z_{\ell}$	$1G_{\ell}+1Z_{\ell}$	$2G_{\ell}+1Z_{\ell}$	$2G_{\ell}+2Z_{\ell}$
Delegation size	$1G_{\ell}$	$1G_{\ell}$	$-$	$2G_{\ell}$
Signature size	$1Z_{\ell}$	$2G_{\ell}$	$1G_{\ell}+2Z_{\ell}$	$1G_{\ell}+1G'_{\ell}$

5. Security Analysis of the proposed CLDVPS scheme

In this section, we discuss only the strong unforgeability and the non-transferability of our CLDVPS scheme. We use the notations and symbols in Section 4 as it is.

Theorem 1. In the random oracle model, if there exists a PPT (in λ) adversary $\mathrm{A}(\lambda)$ that can impersonates a original signer and forge a valid delegation to win sEUF Game against our CLDVPS scheme with a non-negligible probability of success (in λ), then the CDH problem in the group Γ_1 can be solved in polynomial time (in λ) with a non-negligible probability (in λ).

Proof. Suppose that $\mathrm{A}(\lambda)$ succeeds in sEUF Game with a non-negligible probability $\varepsilon(\lambda)$ in time $t(\lambda)$ which is polynomial in λ, and in sEUF Game, $\mathrm{A}(\lambda)$ makes N_{H_1}, N_{H_2}, N_{H_3} queries to H_1, H_2 and H_3 modeled as random oracles respectively. Let N_U and N_{UKey} be respectively the number of queries to $\mathrm{O}_{\mathrm{CreateUse}}$ that $\mathrm{A}(\lambda)$ makes in sEUF Game and the maximum number of same queries on a fixed user identity among them, i.e., the maximum number of key pairs of a fixed user. These are all functions of λ and do not exceed $t(\lambda)$.

Firstly, when $\mathrm{A}(\lambda)$ is a type I adversary $\mathrm{A}_{\mathrm{I}}(\lambda)$, we construct an algorithm X_{I} that uses $\mathrm{A}_{\mathrm{I}}(\lambda)$ to solve the CDH problem in the group Γ_1 as follows:

Algorithm X_{I}:

Input: An instance of the CDH problem, given by q, Γ_1, G, and $[u]G$, $[v]G \in \Gamma$ where $u, v \in \mathrm{Z}_q^*$ are unknown.

Output: The solution $[uv]G \in \Gamma$ of the above instance.

1. Select four indices $0 < \mathrm{I}_O \le N_U$, $0 \le \vartheta_O < N_{UKey}$, $0 < \mathrm{K} \le N_{\mathrm{H}_2}$, $0 < \Lambda \le N_{\mathrm{H}_3}$ and a boolean value $\mathrm{B} | \in \{0,1\}$ r andomly.

2. Simulate a challenger in sEUF Game as follows:

Setup: Let $s \leftarrow_{\mathrm{R}} \mathrm{Z}_q^*$, $P_{\mathrm{KGC}} \leftarrow [s]G$, $Params := \{a, b, p, q, G, P_{\mathrm{KGC}}\}$, $B_{Params} := a \parallel b \parallel G \parallel P_{\mathrm{KGC}}$ and send $Params$ to $\mathrm{A}_{\mathrm{I}}(\lambda)$. Initialize the lists $\mathbf{P}$, $\mathbf{S}$, $\mathbf{R}$ and $\mathbf{D}$ with empty list (∅) respectively, and prepare a list $\mathbf{H}_1$ of 2-tuples, two lists $\mathbf{H}_2$ and $\mathbf{H}_3$ of 3-tuples and a list $\mathbf{L}$ of tuples in form of $(\mathrm{ID}_U, \mathrm{P}_U, \mathrm{K}_U, \mathrm{s}_U, \mathrm{c}_U)$. Choose randomly $h_O^* \in \mathrm{Z}_q^*$ and $h_3^* \in \mathrm{Z}_q^*$. Let $\mathrm{Count}_O := 0$.

Attack: Answer $\mathrm{A}_{\mathrm{I}}(\lambda)$'s queries to oracles as follows:

- $\mathrm{H}_1(x)$: If there exists $h_U \in \mathrm{Z}_q^*$ such that $(x, h_U) \in \mathbf{H}_1$, then return h_U. If $|\mathbf{H}_1| = \mathrm{I}_O - 1$, then put (x, h_O^*) in $\mathbf{H_1}$ and return h_O^*. Otherwise, choose $h_U \in \mathrm{Z}_q^*$ randomly. If $h_U = h_O^*$, then abort. (This case is called *Event* E_1.) Otherwise, put $(x,\ h_U)$ in $\mathbf{H_1}$ and return h_U.
- $\mathrm{H}_2(y)$: If there exists $H_2 \in \Gamma_1$ such that $(y, H_2, *) \in \mathbf{H}_2$ (where * means for an arbitrary value), then return H_2. If $|\mathbf{H}_2| = \mathrm{K} - 1$, then put $(y, [v]G, \perp)$ in $\mathbf{H}_2$ and return $[v]G$. Otherwise, choose $h_2 \in \mathrm{Z}_q^*$ randomly, let $H_2 \leftarrow [h_2]G$, put (y, H_2, h_2) in $\mathbf{H}_2$ and return H_2.
- $\mathrm{H}_3(z)$: If there exists $H_3 \in \Gamma_1$ such that $(z, H_3, *) \in \mathbf{H}_3$ (where * means for an arbitrary value), then return H_3. If $|\mathbf{H}_3| = \Lambda - 1$ then put $(z, [h_3^*]G, h_3^*)$ in $\mathbf{H}_3$ and return $[h_3^*]G$. Otherwise, choose $h_3 \in \mathrm{Z}_q^*$ randomly. If $h_3 = h_3^*$, then abort. (This case is called *Event* E_3.) Otherwise, put $(z, [h_3]G, h_3)$ in $\mathbf{H}_3$ and return $[h_3]G$.
- $\mathrm{O}_{\mathrm{CreateUse}}(\mathrm{ID}_U)$: Let $h_U \leftarrow \mathrm{H}_1(B_{Params} \parallel \mathrm{ID}_U)$ and respond differently in the following cases:
  a. If there exists $(\mathrm{ID}_U, *, *, *, *) \in \mathbf{L}$ (where * means for an arbitrary value), then search $\mathbf{L}$ for the latest item $(\mathrm{ID}_U, *, \mathrm{K}_U, *, \mathrm{c}_U)$ indexed by ID_U to get K_U and c_U. Otherwise, consider the following cases:
  − If $\mathrm{B}| = 0$ and $h_U = h_O^*$, then let $\mathrm{K}_U \leftarrow [u]G$ and $\mathrm{c}_U := \perp$.

– Else, choose $k_U \in Z_q^*$ randomly, let $K_U \leftarrow [k_U]G$ and $c_U \leftarrow (k_U + s \cdot h_U) \bmod q$.

b. If $|B|=1$ and $h_U = h_O^*$, then let $\mathrm{Count}_O \leftarrow \mathrm{Count}_O + 1$. If $\mathrm{Count}_O = \vartheta_O$, then let $P_U \leftarrow [u]G$, put $(ID_U, P_U, K_U, \perp, c_U)$ in **L** and return (P_U, K_U). Otherwise, go forward.

c. Choose $s_U \in Z_q^*$ randomly, let $P_U \leftarrow [s_U]G$, put $(ID_U, P_U, K_U, s_U, c_U)$ in **L**, and return $(P_U$, $K_U)$.

- $O_{\mathrm{GetSecretKey}}(ID_U, P_U)$: Search **L** for the item $(ID_U, P_U, *, s_U, *)$ indexed by (ID_U, P_U) to get s_U. If the search succeeds, put (ID_U, P_U) in **S** and return s_U. Otherwise, return $\perp$.
- $O_{\mathrm{GetPartialKey}}(ID_U, K_U)$: Search **L** for the item $(ID_U, *, K_U, *, c_U)$ indexed by (ID_U, K_U) to get c_U. If the search succeeds, put (ID_U, K_U) in **P** and return c_U. Otherwise, return $\perp$.
- $O_{\mathrm{GetPublicKey}}(ID_U)$: Search **L** for the latest item $(ID_U, P_U, K_U, *, *)$ indexed by ID_U to get P_U and K_U. If the search succeeds, return (P_U, K_U). Otherwise, return $\perp$.
- $O_{\mathrm{ReplaceUserKeys}}(ID_U, P_U', s_U')$: Search **L** for the latest item $(ID_U, *, K_U, *, c_U)$ indexed by ID_U to get K_U and c_U. If the search succeeds, put $(ID_U, P_U', K_U, s_U', c_U)$ in **L**, put (ID_U, P_U') in **R** and return "OK". Otherwise, return $\perp$.
- $O_{\mathrm{GetDelegation}}(ID_O, P_O, K_O, w_O)$: Search **D** for the item $(ID_O, P_O, K_O, w_O, \delta_O)$ indexed by $(ID_O$, $P_O, K_O, w_O)$ to get δ_O. If the search succeeds, return δ_O. Otherwise, follow the next steps:

  a. let $H_2 \hookleftarrow H_2(w_O \,\|\, ID_O \,\|\, P_O \,\|\, K_O)$.

  b. Search $\mathbf{H_2}$ for the item $(w_O \,\|\, ID_O \,\|\, P_O \,\|\, K_O, H_2, h_2)$ indexed by $(w_O \,\|\, ID_O \,\|\, P_O \,\|\, K_O, H_2)$ to get h_2. If $h_2 = \perp$, then abort. (This case is called *Event* D.) Otherwise, go forward.

  c. Search $\mathbf{H_1}$ for the item $(B_{Params} \,\|\, ID_O, h_O)$ indexed by $B_{Params} \,\|\, ID_O$ to get h_O. If the search do es not succeed, return $\perp$.

  d. Search **L** for the latest item $(ID_O, P_O, K_O, *, *)$ indexed by ID_O to get $P_O, K_O \in \Gamma_1$. If the searc h does not succeed, return $\perp$.

  e. Choose $r_O \in Z_q^*$ randomly, let $R_O \leftarrow [r_O]G$.

f. Let $\Delta_O \longleftarrow [r_O]P_{\mathrm{KGC}}+[h_2](\mathrm{P}_O+\mathrm{K}_O+[h_O]P_{\mathrm{KGC}})$.

g. Let $\delta_O := (\mathrm{w}_O, R_O, \Delta_O)$, put $(\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \mathrm{w}_O, \delta_O)$ in $\mathbf{D}$ and return δ_O.

- $\mathrm{O}_{\mathrm{GetDVPSign}}(\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O, \mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D, m)$: Parse δ_O as $(\mathrm{w}_O, R_O, \Delta_O)$ and check whether all input parameters comform to w_O. If not, return $\perp$. If the check is passed, follow the next steps:

  a. Search $\mathbf{H_1}$ for the item $(B_{Params} \| \mathrm{ID}_O, h_O)$ indexed by $B_{Params} \| \mathrm{ID}_O$ to get h_O. If the search do es not succeed, return $\perp$.

  b. Let $H_2 \hookleftarrow \mathrm{H}_2(\mathrm{w}_O \| \mathrm{ID}_O \| \mathrm{P}_O \| \mathrm{K}_O)$ and $\mathrm{T}_O \longleftarrow \mathrm{P}_O + \mathrm{K}_O + [h_O]P_{\mathrm{KGC}}$. If $\hat{e}(\Delta_O, G) \neq \hat{e}(P_{\mathrm{KGC}}, R_O)\cdot \hat{e}(H_2, \mathrm{T}_O)$, then return $\perp$.

  c. Search $\mathbf{H_1}$ for the item $(B_{Params} \| \mathrm{ID}_P, h_P)$ indexed by $B_{Params} \| \mathrm{ID}_P$ to get h_P. If the search doe s not succeed, return $\perp$.

  d. If there does not exist $(\mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, *, *) \in \mathbf{L}$ (where * means for an arbitrary value) such that $\mathrm{P}_O \in \Gamma_1$ and $\mathrm{K}_O \in \Gamma_1$, then return $\perp$.

  e. Let $H_3 \hookleftarrow \mathrm{H}_3(m \| \delta_O \| \mathrm{ID}_P \| \mathrm{P}_P \| \mathrm{K}_P \| \mathrm{ID}_D \| \mathrm{P}_D \| \mathrm{K}_D)$.

  f. Search $\mathbf{H}_3$ for the item $(m \| \delta_O \| \mathrm{ID}_P \| \mathrm{P}_P \| \mathrm{K}_P \| \mathrm{ID}_D \| \mathrm{P}_D \| \mathrm{K}_D, H_3, h_3)$ indexed by $(m \| \delta_O \| \mathrm{ID}_P \| \mathrm{P}_P \| \mathrm{K}_P \| \mathrm{ID}_D \| \mathrm{P}_D \| \mathrm{K}_D, H_3)$ to get h_3.

  g. If $h_3 = h_3^*$, then abort. (This case is called *Event* S .)

  h. Search $\mathbf{H_1}$ for the item $(B_{Params} \| \mathrm{ID}_D, h_D)$ indexed by $B_{Params} \| \mathrm{ID}_D$ to get h_D. If the search do es not succeed, return $\perp$.

  i. Let $\mathrm{T}_P \longleftarrow \mathrm{P}_P + \mathrm{K}_P + [h_P]P_{\mathrm{KGC}}$ and $\mathrm{T}_D \longleftarrow \mathrm{P}_D + \mathrm{K}_D + [h_D]P_{\mathrm{KGC}}$.

  j. Choose $r_P \in \mathrm{Z}_q^*$ randomly, let $R_P \longleftarrow [r_P]G$ and $v_m \longleftarrow \hat{e}([r_P]P_{\mathrm{KGC}} + [h_3]\mathrm{T}_P, \mathrm{T}_D)$.

  k. Let $\sigma_m := (R_P, v_m)$ and return σ_m.

- $\mathrm{O}_{\mathrm{VerifyDVPSign}}(\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O, \mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D, m, \sigma_m)$: Parse δ_O as $(\mathrm{w}_O, R_O, \Delta_O)$ and check whether all input parameters comform to w_O. If not, return **False**. If the check is passed, follow the next steps:

a. Search $\mathbf{H_1}$ for the item ($B_{Params} \| \mathrm{ID}_O , h_O$) indexed by $B_{Params} \| \mathrm{ID}_O$ to get h_O. If the search do

es not succeed, return **False**.

b. Let $H_2 \hookleftarrow \mathrm{H}_2 (\mathrm{w}_O \| \mathrm{ID}_O \| \mathrm{P}_O \| \mathrm{K}_O)$ and $\mathrm{T}_O \longleftarrow \mathrm{P}_O + \mathrm{K}_O +[h_O] P_{\mathrm{KGC}}$. If $\hat{e} (\Delta_O , G) \neq \hat{e} (P_{\mathrm{KGC}}, R_O) \cdot \hat{e} (H_2 , \mathrm{T}_O)$ then return **False**.

c. Parse σ_m as (R_P , v_m) and let $H_3 \hookleftarrow \mathrm{H}_3 (m \| \delta_O \| \mathrm{ID}_P \| \mathrm{P}_P \| \mathrm{K}_P \| \mathrm{ID}_D \| \mathrm{P}_D \| \mathrm{K}_D)$.

d. Search $\mathbf{H}_3$ for the item ($m \| \delta_O \| \mathrm{ID}_P \| \mathrm{P}_P \| \mathrm{K}_P \| \mathrm{ID}_D \| \mathrm{P}_D \| \mathrm{K}_D , H_3 , h_3$) indexed by ($m \| \delta_O \| \mathrm{ID}_P \| \mathrm{P}_P \| \mathrm{K}_P \| \mathrm{ID}_D \| \mathrm{P}_D \| \mathrm{K}_D , H_3$) to get h_3.

e. Search $\mathbf{H_1}$ for the item ($B_{Params} \| \mathrm{ID}_P , h_P$) indexed by $B_{Params} \| \mathrm{ID}_P$ to get h_P. If the search doe

s not succeed, return **False**.

f. Search $\mathbf{H_1}$ for the item ($B_{Params} \| \mathrm{ID}_D , h_D$) indexed by $B_{Params} \| \mathrm{ID}_D$ to get h_D. If the search do

es not succeed, return **False**.

g. Let $\mathrm{T}_P \longleftarrow \mathrm{P}_P + \mathrm{K}_P +[h_P] P_{\mathrm{KGC}}$ and $\mathrm{T}_D \longleftarrow \mathrm{P}_D + \mathrm{K}_D +[h_D] P_{\mathrm{KGC}}$.

h. If $v_m = \hat{e} ([s]R_P +[h_3] \mathrm{T}_P , \mathrm{T}_D)$, then return **True**. Otherwise, return **False**.

Forgery: Wait until $\mathrm{A}_{\mathrm{I}}(\lambda)$ outputs a tuple $(\mathrm{ID}_O^* , \mathrm{P}_O^* , \mathrm{K}_O^* , \delta_O^* , \mathrm{ID}_P^* , \mathrm{P}_P^* , \mathrm{K}_P^* , \mathrm{ID}_D^* , \mathrm{P}_D^* , \mathrm{K}_D^* , m^* , \sigma_{m^*})$ as the final result of the game.

3. Check $\mathrm{A}_{\mathrm{I}}(\lambda)$'s final result and output the solution of the above instance (i.e., the input) as follows:

(1) If the following boolean expression is true, then abort. (This case is called *Event* R_1.).

$$h_O^* \neq \mathrm{H}_1 (B_{Params} \| \mathrm{ID}_O^*) \vee [v]G \neq \mathrm{H}_2 (\mathrm{w}_O^* \| \mathrm{ID}_O^* \| \mathrm{P}_O^* \| \mathrm{K}_O^*) \vee$$

$$[h_3^*] G \neq \mathrm{H}_3 (m^* \| \delta_O^* \| \mathrm{ID}_P^* \| \mathrm{P}_P^* \| \mathrm{K}_P^* \| \mathrm{ID}_D^* \| \mathrm{P}_D^* \| \mathrm{K}_D^*).$$

(2) Parse δ_O^* as $(\mathrm{w}_O^* , R_O^* , \Delta_O^*)$. If the following boolean expression is true, then abort since δ_O^* must be fo

rged by $\mathrm{A}_{\mathrm{I}}(\lambda)$ from the assumption of the theorem. (This case is called *Event* R_2 .)

$$(((\mathrm{ID}_O^* , \mathrm{K}_O^*) \in \mathbf{P} \wedge (\mathrm{ID}_O^* , \mathrm{P}_O^*) \in \mathbf{S} \cup \mathbf{R}) \vee (\mathrm{ID}_O^* , \mathrm{P}_O^* , \mathrm{K}_O^* , \mathrm{w}_O^* , \delta_O^*) \in \mathbf{D}).$$

(3) If $\mathrm{O}_{\mathrm{VerifyDVPSign}} (\mathrm{ID}_O^* , \mathrm{P}_O^* , \mathrm{K}_O^* , \delta_O^* , \mathrm{ID}_P^* , \mathrm{P}_P^* , \mathrm{K}_P^* , \mathrm{ID}_D^* , \mathrm{P}_D^* , \mathrm{K}_D^* , m^* , \sigma_{m^*})$=**False**, then then abort. (This case is called *Event* R_3 .)

(4) Search $\mathbf{L}$ for the item $(\mathrm{ID}_O^* , \mathrm{P}_O^* , \mathrm{K}_O^* , \mathrm{s}_O^* , \mathrm{c}_O^*)$ indexed by $(\mathrm{ID}_O^* , \mathrm{P}_O^* , \mathrm{K}_O^*)$ to get s_O^* and c_O^*. If the sear

ch does not succeed, then abort. (This case is called *Event* R_4.)

(5) If $B|=0$, then return ($\Delta_O^* - [s]R_O^* - [(s_O^* + s\cdot h_O^*) \bmod q][v]G$). If $B|=1$, then return ($\Delta_O^* - [s]R_O^* - [c_O^*][v]G$).

(The end of the algoritm X_I)

Now, we analyze the time complexity of X_I. The time complexity of X_I is dominated by the scalar multiplications, additions and pairing in Γ_1, operations in Z_q^* and searches on lists which are performed in the simulations of oracles. All operands for the above operations in Γ_1 and Z_q^* have at most a polynomial size in λ and we can assume that the computational cost of any operation in X_I is less than some polynomial $p(\lambda)$. On the other hand, the sizes of all lists in oracle simulations of X_I do not go beyond the number of all queries by $A_I(\lambda)$, which is not larger than $t(\lambda)$. It results that the time of a search on any list in X_I is also not longer than $t(\lambda)$. Therefore, the cost of answering an oracle query in X_I is also less than $O(p(\lambda)+t(\lambda))$, and we conclude that the total time complexity of X_I is $O(t(\lambda)p(\lambda)+t^2(\lambda))$ which is also polynomial in λ.

It is obvious that if X_I does not abort, then $A_I(\lambda)$'s view in the simulated game with X_I is indistinguishable from its view in the real attack.

Next, we show that X_I 's output is the solution for the instance of the CDH problem, given as input of X_I, if X_I does not abort.

Assume that X_I does not abort and outputs some result.

If $B|=0$, then $K_O^* = [u]G$, $c_O^* = \perp$, $s_O^* \in Z_q^*$ and $P_O^* = [s_O^*]G$ from the step (a) and (c) of $O_{\mathrm{CreateUse}}$. In this case, X_I 's output is ($\Delta_O^* - [s]R_O^* - [(s_O^* + s\cdot h_O^*) \bmod q][v]G$) from the step 3(5) of X_I. The validity of $A_I(\lambda)$'s final result (the forged signature) implies the validity of δ_O^*, and we obtain the following equation from the validity of δ_O^* and the equation $[v]G = H_2(w_O^* \| ID_O^* \| P_O^* \| K_O^*)$:

$$\hat{e}(\Delta_O^*, G) = \hat{e}(P_{\mathrm{KGC}}, R_O^*)\cdot \hat{e}([v]G, T_O^*),$$

where $T_O^* = P_O^* + K_O^* + [h_O^*]P_{\mathrm{KGC}}$. From the above equation, we can obtain

$$\hat{e}(\Delta_O^*, G) = \hat{e}(P_{\mathrm{KGC}}, R_O^*)\cdot \hat{e}([v]G, P_O^* + K_O^* + [h_O^*]P_{\mathrm{KGC}})$$

$$= \hat{e}([s]G, R_O^*) \cdot \hat{e}([v]G, [s_O^*]G + [u]G + [h_O^*][s]G)$$

$$= \hat{e}([s]G, R_O^*) \cdot \hat{e}([v]G, [(s_O^* + s \cdot h_O^*) \bmod q]G + [u]G)$$

$$= \hat{e}([s]R_O^*, G) \cdot \hat{e}([(s_O^* + s \cdot h_O^*) \bmod q][v]G, G) \cdot \hat{e}([v]G, [u]G),$$

and thus,

$$\hat{e}([v]G, [u]G) = \hat{e}(\Delta_O^* - [s]R_O^* - [(s_O^* + s \cdot h_O^*) \bmod q][v]G, G).$$

From the non-degeneracy of $\hat{e}$, we can obtain

$$[uv]G = \Delta_O^* - [s]R_O^* - [(s_O^* + s \cdot h_O^*) \bmod q][v]G .$$

Therefore, $\Delta_O^* - [s]R_O^* - [(s_O^* + s \cdot h_O^*) \bmod q][v]G$ is the solution for the instance given as input of X_I.

Similarly, If B|=1, then $P_O^* = [u]G, s_O^* = \perp, k_O^* \in Z_q^*, c_O^* = (k_O^* + s \cdot h_O^*) \bmod q$ and $K_O^* = [k_O^*]G$ from the step (b) of $O_{\text{CreateUse}}$. In this case, X_I's output is $(\Delta_O^* - [s]R_O^* - [c_O^*][v]G)$ from the step 3(5) of X_I. We obtain the following equation from the validity of δ_O^* and the equation $[v]G = H_2(w_O^* \| ID_O^* \| P_O^* \| K_O^*)$:

$$\hat{e}(\Delta_O^*, G) = \hat{e}(P_{\text{KGC}}, R_O^*) \cdot \hat{e}([v]G, T_O^*),$$

where $T_O^* = [u]G + [k_O^*]G + [h_O^*]P_{\text{KGC}}$. From the above equation, we can obtain

$$\hat{e}(\Delta_O^*, G) = \hat{e}(P_{\text{KGC}}, R_O^*) \cdot \hat{e}([v]G, [u]G + [k_O^*]G + [h_O^*]P_{\text{KGC}})$$

$$= \hat{e}([s]G, R_O^*) \cdot \hat{e}([v]G, [u]G + [k_O^*]G + [h_O^*][s]G)$$

$$= \hat{e}([s]G, R_O^*) \cdot \hat{e}([v]G, [(k_O^* + s \cdot h_O^*) \bmod q]G + [u]G)$$

$$= \hat{e}([s]R_O^*, G) \cdot \hat{e}([c_O^*][v]G, G) \cdot \hat{e}([v]G, [u]G),$$

and thus,

$$\hat{e}([v]G, [u]G) = \hat{e}(\Delta_O^* - [s]R_O^* - [c_O^*][v]G, G).$$

From the non-degeneracy of $\hat{e}$, we can obtain

$$[uv]G = \Delta_O^* - [s]R_O^* - [c_O^*][v]G .$$

Therefore, $\Delta_O^* - [s]R_O^* - [c_O^*][v]G$ is the solution for the instance given as input of X_I.

Now, we analyze the probability for X_I to output the solution of the given instance of the CDH problem.

We know that Event R_2 and R_3 never occur in the running of X_I if $A_I(\lambda)$ succeeds the simulated game with X_{I}, i.e., $\mathrm{A}_{\mathrm{I}}(\lambda)$ provides a valid delegation and a valid proxy signature. And if X_{I}'s selections of four indices I_{O}, ϑ_{O}, K, Λ and a Boolean value $\mathrm{B}|$ are correct for the final result $(\mathrm{ID}_{O}^{*}, \mathrm{P}_{O}^{*}, \mathrm{K}_{O}^{*}, \delta_{O}^{*}, \mathrm{ID}_{P}^{*}, \mathrm{P}_{P}^{*}, \mathrm{K}_{P}^{*}, \mathrm{ID}_{D}^{*}, \mathrm{P}_{D}^{*}, \mathrm{K}_{D}^{*}, m^{*}, \sigma_{m^{*}})$ of $\mathrm{A}_{\mathrm{I}}(\lambda)$, i.e., the target original signer with ID_{O}^{*} is the I_{O}-th user, $(\mathrm{P}_{O}^{*}, \mathrm{K}_{O}^{*})$ is the ϑ_{O}-th full public key of the target original signer, $\mathrm{H}_{2}(\mathrm{w}_{O}^{*} \| \mathrm{ID}_{O}^{*} \| \mathrm{P}_{O}^{*} \| \mathrm{K}_{O}^{*})$ (where w_{O}^{*} is the warrant in the delegation δ_{O}^{*}) is the K-th new query to H_{2}, $\mathrm{H}_{3}(m^{*} \| \delta_{O}^{*} \| \mathrm{ID}_{P}^{*} \| \mathrm{P}_{P}^{*} \| \mathrm{K}_{P}^{*} \| \mathrm{ID}_{D}^{*} \| \mathrm{P}_{D}^{*} \| \mathrm{K}_{D}^{*})$ is the Λ-th new query to H_{3}, $\mathrm{B}|=0$ in the case $(\mathrm{ID}_{O}^{*}, \mathrm{K}_{O}^{*}) \notin \mathbf{P}$ and $\mathrm{B}|=1$ in the case $(\mathrm{ID}_{O}^{*}, \mathrm{P}_{O}^{*}) \notin \mathbf{S} \cup \mathbf{R}$, Event D, S, R_{1} and R_{4} never occur in the running of X_{I}.

The probability that X_{I}'s selections of I_{O}, ϑ_{O}, K, Λ and $\mathrm{B}|$ are correct is at least $1/(N_{U} \cdot N_{UKey} \cdot N_{\mathrm{H}_{2}} \cdot N_{\mathrm{H}_{3}} \cdot 2)$. The probabilities of Events E_{1} and E_{3} in the running of X_{I} is at most $N_{\mathrm{H}_{1}}/q$ and $N_{\mathrm{H}_{3}}/q$, respectively.

Therefore, the probability that X_{I} outputs the solution of the given instance of CDH problem is at least the following expression:

$$\varepsilon(\lambda)/(N_{U} \cdot N_{UKey} \cdot N_{\mathrm{H}_{2}} \cdot N_{\mathrm{H}_{3}} \cdot 2)(1-N_{\mathrm{H}_{1}}/q)(1-N_{\mathrm{H}_{3}}/q).$$

Since $N_{\mathrm{H}_{1}}, N_{\mathrm{H}_{2}}, N_{\mathrm{H}_{3}}, N_{U}$ and N_{UKey} are polynomial in λ, q is exponential in λ, $\varepsilon(\lambda)$ is non-negligible, the a bove expression is also non-negligible.

Secondly, when $\mathrm{A}(\lambda)$ is a type II adversary $\mathrm{A}_{\mathrm{II}}(\lambda)$, we construct an algorithm X_{II} that uses $\mathrm{A}_{\mathrm{II}}(\lambda)$ to solve the CDH problem in the group Γ_{1} as follows:

Algorithm X_{II}:

Input/*Output*: (These are the same as those in Algorithm X_{I}.)

1. Select four indices $0<\mathrm{I}_{O} \le N_{U}$, $0 \le \vartheta_{O} < N_{UKey}$, $0<\mathrm{K} \le N_{\mathrm{H}_{2}}$ and $0<\Lambda \le N_{\mathrm{H}_{3}}$ randomly.

2. Simulate a challenger in EUF-CMA Game as follows:

Setup: Let $s \leftarrow_{\mathrm{R}} \mathrm{Z}_{q}^{*}, P_{\mathrm{KGC}} \leftarrow [s]G$, $Params := \{a, b, p, q, G, P_{\mathrm{KGC}}\}$, $B_{Params} := a \| b \| G \| P_{\mathrm{KGC}}$. Send $Params$ and s to $\mathrm{A}_{\mathrm{II}}(\lambda)$. Initialize the lists $\mathbf{S}$ and $\mathbf{D}$ with empty list ($\emptyset$) respectively, and prepare a list $\mathbf{H}_{1}$ of 2-tuples, two lists $\mathbf{H}_{2}$ and $\mathbf{H}_{3}$ of 3-tuples and a list $\mathbf{L}$ of tuples in form of $(\mathrm{ID}_{U}, \mathrm{P}_{U}, \mathrm{K}_{U}, \mathrm{s}_{U}, \mathrm{c}_{U})$.

Choose randomly $h_O^* \in Z_q^*$ and $h_3^* \in Z_q^*$. Let Count_O :=0.

Attack: Answer $\mathrm{A}_{\mathrm{II}}(\lambda)$'s queries to oracles as follows:

- $\mathrm{H}_1(x)$, $\mathrm{H}_2(y), \mathrm{H}_3(z)$: (These are the same as that in Algorithm X_{I}.)
- $\mathrm{O}_{\mathrm{CreateUser}}(\mathrm{ID}_U)$: Let $h_U \leftarrow\!\!\dashv \mathrm{H}_1(B_{Params} \| \mathrm{ID}_U)$ and respond differently in the following cases:
    a. If there exists $(\mathrm{ID}_U, *, *, *, *) \in \mathbf{L}$ (where * means for an arbitrary value), then search $\mathbf{L}$ for the latest item $(\mathrm{ID}_U, *, \mathrm{K}_U, *, \mathrm{c}_U)$ indexed by ID_U to get K_U and c_U. Otherwise, choose $\mathrm{k}_U \in Z_q^*$ randomly, let $\mathrm{K}_U \longleftarrow [\mathrm{k}_U]G$ and $\mathrm{c}_U \longleftarrow (\mathrm{k}_U + s \cdot h_U) \bmod q$.

    b. If $h_U = h_O^*$, then let $\mathrm{Count}_O \longleftarrow \mathrm{Count}_O + 1$. If $\mathrm{Count}_O = \vartheta_O$, then let $\mathrm{P}_U \longleftarrow [u]G$, put $(\mathrm{ID}_U, \mathrm{P}_U, \mathrm{K}_U, \perp, \mathrm{c}_U)$ in $\mathbf{L}$ and return $(\mathrm{P}_U, \mathrm{K}_U)$. Otherwise, go forward.

    c. (This is the same as that in Algorithm X_{I}.).
- $\mathrm{O}_{\mathrm{GetSecretKey}}, \mathrm{O}_{\mathrm{GetPublicKey}}, \mathrm{O}_{\mathrm{GetDelegation}}, \mathrm{O}_{\mathrm{GetDVPSign}}, \mathrm{O}_{\mathrm{VerifyDVPSign}}$: (These are the same as those in Algorithm X_{I}.)

Forgery: Wait until $\mathrm{A}_{\mathrm{II}}(\lambda)$ outputs a tuple $(\mathrm{ID}_O^*, \mathrm{P}_O^*, \mathrm{K}_O^*, \delta_O^*, \mathrm{ID}_P^*, \mathrm{P}_P^*, \mathrm{K}_P^*, \mathrm{ID}_D^*, \mathrm{P}_D^*, \mathrm{K}_D^*, m^*, \sigma_{m^*})$ as the final result of the game.

3. Check $\mathrm{A}_{\mathrm{II}}(\lambda)$'s final result and output the solution of the above instance (i.e., the input) as follows:

(1) (This is the same as that in Algorithm X_{I}.)

(2) Parse δ_O^* as $(\mathrm{w}_O^*, R_O^*, \Delta_O^*)$. If the following boolean expression is true, then abort. (This case is called *Event* R_2.).

$$((\mathrm{ID}_O^*, \mathrm{P}_O^*) \in \mathbf{S} \vee (\mathrm{ID}_O^*, \mathrm{P}_O^*, \mathrm{K}_O^*, \mathrm{w}_O^*, \delta_O^*) \in \mathbf{D}).$$

(3), (4), (5) (These are the same as those in Algorithm X_{I}.)

(The end of the algoritm X_{II})

Since the analysis of X_{II} is similar to the one of X_{I}, we omit it. This completes the proof.

Theorem 2. In the random oracle model, if there exists a PPT (in λ) adversary $\mathrm{A}'(\lambda)$ that can impersonates a proxy signer on the basis of a legitimate delegation to win sEUF Game against our CLDVPS scheme with a non-negligible probability of success (in λ), then the GBDH problem in the bilinear group pair (Γ_1, Γ') can be solved in polynomial time (in λ) with a non-negligible probability (in λ).

Proof. Suppose that $\mathrm{A}'(\lambda)$ as a impersonated proxy signer succeeds in sEUF Game with a non-negligible probability $\varepsilon(\lambda)$ in time $t(\lambda)$ which is polynomial in λ. Let N_{H_1}, N_{H_3}, N_U and N_{UKey} be the same as that in the proof of Theorem 1.

Firstly, when $\mathrm{A}'(\lambda)$ is a type I adversary $\mathrm{A}'_{\mathrm{I}}(\lambda)$, we construct an algorithm X'_{I} that uses $\mathrm{A}'_{\mathrm{I}}(\lambda)$ and the DBDH oracle $\mathrm{O}_{\mathrm{DBDH}}$ to solve the GBDH problem in the bilinear group pair (Γ_1, Γ') as follows:

Algorithm X'_{I}:

Input: An instance of the GBDH problem, given by q, Γ_1, G, and $[u]G$, $[v]G$, $[w]G \in \Gamma$ where $u, v, w \in \mathrm{Z}_q^*$ are unknown.

Output: The solution $\hat{e}(G,G)^{uvw} \in \Gamma'$ of the above instance.

1. Select five indices $0 < \mathrm{I}_P, \mathrm{I}_D \le N_U$, $0 \le \vartheta_P, \vartheta_D < N_{UKey}$, $0 < \Lambda \le N_{\mathrm{H}_3}$ and two boolean values $\mathrm{B}\beta_P, \mathrm{B}\beta_D \in \{0,1\}$ randomly.

2. Simulate a challenger in sEUF Game as follows:

Setup: Let $s \leftarrow_R \mathrm{Z}_q^*$, $P_{\mathrm{KGC}} \leftarrow [s]G$, $Params := \{a, b, p, q, G, P_{\mathrm{KGC}}\}$, $B_{Params} := a \,\|\, b \,\|\, G \,\|\, P_{\mathrm{KGC}}$ and send $Params$ to $\mathrm{A}'_{\mathrm{I}}(\lambda)$. Initialize the lists $\mathbf{P}$, $\mathbf{S}$, $\mathbf{R}$ and $\mathbf{D}$ with empty list (∅) respectively, and prepare a list $\mathbf{H}_1$ of 2-tuples, two lists $\mathbf{H}_2$ and $\mathbf{H}_3$ of 3-tuples and a list $\mathbf{L}$ of tuples in form of (ID_U, P_U, K_U, s_U, c_U). Choose randomly $h_P^* \in \mathrm{Z}_q^*$ and $h_D^* \in \mathrm{Z}_q^*$ such that $h_P^* \ne h_D^*$. Let $\mathrm{Count}_P := 0$ and $\mathrm{Count}_D := 0$.

Attack: Answer $\mathrm{A}'_{\mathrm{I}}(\lambda)$'s queries to oracles as follows:

- $\mathrm{H}_1(x)$: If there exists $h_U \in \mathrm{Z}_q^*$ such that $(x, h_U) \in \mathbf{H}_1$, then return h_U. If $|\mathbf{H}_1| = \mathrm{I}_P - 1$, then put (x, h_P^*) in $\mathbf{H_1}$ and return h_P^*. If $|\mathbf{H}_1| = \mathrm{I}_D - 1$, then put (x, h_D^*) in $\mathbf{H_1}$ and return h_D^*. Otherwise, choose $h_U \in \mathrm{Z}_q^*$ randomly. If $h_U = h_P^*$ or $h_U = h_D^*$, then abort. (This case is called *Event* E_1.) Otherwise, put (x, h_U) in $\mathbf{H_1}$ and return h_U.

- $\mathrm{H}_2(y)$: If there exists $H_2 \in \Gamma_1$ such that $(y, H_2, *) \in \mathbf{H}_2$ (where * means for an arbitrary value), then return H_2. Otherwise, choose $h_2 \in \mathrm{Z}_q^*$ randomly, put $(y, [h_2]G, h_2)$ in $\mathbf{H}_2$ and return $[h_2]G$.
- $\mathrm{H}_3(z)$: If there exists $H_3 \in \Gamma_1$ such that $(z, H_3, *) \in \mathbf{H}_3$ (where * means for an arbitrary value), then return H_3. If $|\mathbf{H}_3| = \Lambda - 1$ then put $(z, [w]G, \perp)$ in $\mathbf{H}_3$ and return $[w]G$. Otherwise, choose $h_3 \in \mathrm{Z}_q^*$ randomly, put $(z, [h_3]G, h_3)$ in $\mathbf{H}_3$ and return $[h_3]G$.
- $\mathrm{O}_{\mathrm{CreateUser}}(\mathrm{ID}_U)$: Let $h_U \leftarrow \mathrm{H}_1(B_{Params} \| \mathrm{ID}_U)$ and respond differently in the following cases:
  a. If there exists $(\mathrm{ID}_U, *, *, *, *) \in \mathbf{L}$ (where * means for an arbitrary value), then search $\mathbf{L}$ for the latest item $(\mathrm{ID}_U, *, \mathrm{K}_U, *, \mathrm{c}_U)$ indexed by ID_U to get K_U and c_U. Otherwise, consider the following cases:
  - If $\mathrm{B}\beta_P = 0$ and $h_U = h_P^*$, then let $\mathrm{K}_U \leftarrow [u]G$ and $\mathrm{c}_U := \perp$.
  - If $\mathrm{B}\beta_D = 0$ and $h_U = h_D^*$, then let $\mathrm{K}_U \leftarrow [v]G$ and $\mathrm{c}_U := \perp$.
  - Else, choose $\mathrm{k}_U \in \mathrm{Z}_q^*$ randomly, let $\mathrm{K}_U \leftarrow [\mathrm{k}_U]G$ and $\mathrm{c}_U \leftarrow (\mathrm{k}_U + s \cdot h_U) \bmod q$.

  b. If $\mathrm{B}\beta_P = 1$ and $h_U = h_P^*$, then let $\mathrm{Count}_P \leftarrow \mathrm{Count}_P + 1$. If $\mathrm{Count}_P = \vartheta_P$, then let $\mathrm{P}_U \leftarrow [u]G$, put $(\mathrm{ID}_U, \mathrm{P}_U, \mathrm{K}_U, \perp, \mathrm{c}_U)$ in $\mathbf{L}$ and return $(\mathrm{P}_U, \mathrm{K}_U)$. Otherwise, go forward.

  c. If $\mathrm{B}\beta_D = 1$ and $h_U = h_D^*$, then let $\mathrm{Count}_D \leftarrow \mathrm{Count}_D + 1$. If $\mathrm{Count}_D = \vartheta_D$, then let $\mathrm{P}_U \leftarrow [v]G$, put $(\mathrm{ID}_U, \mathrm{P}_U, \mathrm{K}_U, \perp, \mathrm{c}_U)$ in $\mathbf{L}$ and return $(\mathrm{P}_U, \mathrm{K}_U)$. Otherwise, go forward.

  d. Choose $\mathrm{s}_U \in \mathrm{Z}_q^*$ randomly, let $\mathrm{P}_U \leftarrow [\mathrm{s}_U]G$, put $(\mathrm{ID}_U, \mathrm{P}_U, \mathrm{K}_U, \mathrm{s}_U, \mathrm{c}_U)$ in $\mathbf{L}$, and return $(\mathrm{P}_U, \mathrm{K}_U)$.
- $\mathrm{O}_{\mathrm{GetSecretKey}}, \mathrm{O}_{\mathrm{GetPartialKey}}, \mathrm{O}_{\mathrm{GetPublicKey}}, \mathrm{O}_{\mathrm{ReplaceUserKeys}}$: (These are the same as those in Algorithm X_I in the proof of Theorem 1.)
- $\mathrm{O}_{\mathrm{GetDelegation}}(\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \mathrm{w}_O)$: (This is the same as that in Algorithm X_I in the proof of Theorem 1, except that the step (b) in Algorithm X_I is replaced by the following one:

  b. Search $\mathbf{H}_2$ for the item $(\mathrm{w}_O \| \mathrm{ID}_O \| \mathrm{P}_O \| \mathrm{K}_O, H_2, h_2)$ indexed by $(\mathrm{w}_O \| \mathrm{ID}_O \| \mathrm{P}_O \| \mathrm{K}_O, H_2)$ to get h_2.)

- $\mathrm{O}_{\mathrm{GetDVPSign}}(\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O, \mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D, m)$: (This is the same as that in Algorithm $\mathrm{X_I}$ in the proof of Theorem 1, except that the step (g) in Algorithm $\mathrm{X_I}$ is replaced by the following one:

  g. If $h_3 = \perp$, then abort. (This case is called *Event* S .))

- $\mathrm{O}_{\mathrm{VerifyDVPSign}}(\mathrm{ID}_O, \mathrm{P}_O, \mathrm{K}_O, \delta_O, \mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D, m, \sigma_m)$: Parse δ_O as $(\mathrm{w}_O, R_O, \Delta_O)$ and check whether all input parameters comform to w_O. If not, return **False**. If the check is passed, follow the next steps:

  a~g. (These steps are the same as those in Algorithm $\mathrm{X_I}$ in the proof of Theorem 1.)

  h. If $h_3 \neq \perp$, then return $v_m \overset{?}{=} \hat{e}([s]R_P + [h_3]\mathrm{T}_P, \mathrm{T}_D)$. Otherwise, go forward.

  i. Search $\mathbf{H_1}$ for the item $(B_{Params} \| \mathrm{ID}_P, h_P)$ indexed by $B_{Params} \| \mathrm{ID}_P$ to get h_P. If the search does not succeed, return $\perp$.

  j. Search $\mathbf{L}$ for the item $(\mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{s}_P, \mathrm{c}_P)$ indexed by $(\mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P)$ to get s_P and c_P. If the search does not succeed, return $\perp$. If $\mathrm{s}_P = \perp$ and $\mathrm{c}_P = \perp$, return $\perp$. If $\mathrm{s}_P \neq \perp$ and $\mathrm{c}_P \neq \perp$, return $v_m \overset{?}{=} \hat{e}([s]R_P + [(\mathrm{s}_P + \mathrm{c}_P) \bmod q]H_3, \mathrm{T}_D)$.

  k. Search $\mathbf{H_1}$ for the item $(B_{Params} \| \mathrm{ID}_D, h_D)$ indexed by $B_{Params} \| \mathrm{ID}_D$ to get h_D. If the search do es not succeed, return $\perp$.

  l. Search $\mathbf{L}$ for the latest item $(\mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D, \mathrm{s}_D, \mathrm{c}_D)$ indexed by $(\mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D)$ to get s_D and c_D. If the search does not succeed, return $\perp$. If $\mathrm{s}_D = \perp$ and $\mathrm{c}_D = \perp$, return $\perp$. If $\mathrm{s}_D \neq \perp$ and $\mathrm{c}_D \neq \perp$, return $v_m \overset{?}{=} \hat{e}(H_3, [(\mathrm{s}_D + \mathrm{c}_D) \bmod q]\mathrm{T}_P) \cdot \hat{e}(R_P, [s]\mathrm{T}_D)$.

  m. If $\mathrm{c}_P = \perp$ and $\mathrm{c}_D = \perp$, let $h \leftarrow \hat{e}(H_3, [(\mathrm{s}_D + s \cdot h_D) \bmod q]\mathrm{T}_P + [(\mathrm{s}_P + s \cdot h_P) \bmod q]\mathrm{K}_D) \cdot \hat{e}(R_P, [s]\mathrm{T}_D)$.

  n. If $\mathrm{c}_P = \perp$ and $\mathrm{s}_D = \perp$, let $h \leftarrow \hat{e}(H_3, [\mathrm{c}_D]\mathrm{T}_P + [(\mathrm{s}_P + s \cdot h_P) \bmod q]\mathrm{P}_D) \cdot \hat{e}(R_P, [s]\mathrm{T}_D)$.

  o. If $\mathrm{s}_P = \perp$ and $\mathrm{c}_D = \perp$, let $h \leftarrow \hat{e}(H_3, [(\mathrm{s}_D + s \cdot h_D) \bmod q]\mathrm{T}_P + [\mathrm{c}_P]\mathrm{K}_D) \cdot \hat{e}(R_P, [s]\mathrm{T}_D)$.

  p. If $\mathrm{s}_P = \perp$ and $\mathrm{s}_D = \perp$, let $h \leftarrow \hat{e}(H_3, [\mathrm{c}_D]\mathrm{T}_P + [\mathrm{c}_P]\mathrm{K}_D) \cdot \hat{e}(R_P, [s]\mathrm{T}_D)$.

  q. return $\mathrm{O}_{\mathrm{DBDH}}([u]G, [v]G, [w]G, v_m / h)$.

Forgery: Wait until $\mathrm{A}'_{\mathrm{I}}(\lambda)$ outputs a tuple $(\mathrm{ID}_O^*, \mathrm{P}_O^*, \mathrm{K}_O^*, \delta_O^*, \mathrm{ID}_P^*, \mathrm{P}_P^*, \mathrm{K}_P^*, \mathrm{ID}_D^*, \mathrm{P}_D^*, \mathrm{K}_D^*, m^*, \sigma_{m^*})$ as the final result of the game.

3. Check $\mathrm{A}'_{\mathrm{I}}(\lambda)$'s final result and output the solution of the above instance (i.e., the input) as follows:

(1) If the following boolean expression is true, then abort. (This case is called *Event* R_1.).

$$h_P^* \neq \mathrm{H}_1(B_{Params} \| \mathrm{ID}_P^*) \vee h_D^* \neq \mathrm{H}_1(B_{Params} \| \mathrm{ID}_D^*) \vee$$

$$[w]G \neq \mathrm{H}_3(m^* \| \delta_O^* \| \mathrm{ID}_P^* \| \mathrm{P}_P^* \| \mathrm{K}_P^* \| \mathrm{ID}_D^* \| \mathrm{P}_D^* \| \mathrm{K}_D^*).$$

(2) If the following boolean expression is true, then abort. (This case is called *Event* R_2 .)

$$(((\mathrm{ID}_P^*, \mathrm{K}_P^*) \in \mathbf{P} \wedge (\mathrm{ID}_P^*, \mathrm{P}_P^*) \in \mathbf{S} \cup \mathbf{R}) \vee ((\mathrm{ID}_D^*, \mathrm{K}_D^*) \in \mathbf{P} \wedge (\mathrm{ID}_D^*, \mathrm{P}_D^*) \in \mathbf{S} \cup \mathbf{R})).$$

(3) If $\mathrm{O}_{\mathrm{VerifyDVPSign}}(\mathrm{ID}_O^*, \mathrm{P}_O^*, \mathrm{K}_O^*, \delta_O^*, \mathrm{ID}_P^*, \mathrm{P}_P^*, \mathrm{K}_P^*, \mathrm{ID}_D^*, \mathrm{P}_D^*, \mathrm{K}_D^*, m^*, \sigma_{m^*})$=**False**, then then abort. (This case is called *Event* R_3.) Otherwise, parse σ_{m^*} as (R_P^*, v_{m^*}) and return v_{m^*}/h^*. (v_{m^*}/h^* is the last parameter of the query to $\mathrm{O}_{\mathrm{DBDH}}$ in the oracle $\mathrm{O}_{\mathrm{VerifyDVPSign}}$.)

(The end of the algoritm X'_{I})

Since the analysis of X'_{I} is similar to the one of X_{I} in the proof of Theorem 1, we only show that X'_{I} 's output is the solution for the given instance of the GBDH problem if X'_{I} does not abort.

X'_{I} can output the result v_{m^*}/h^* if and only if $\mathrm{A}'_{\mathrm{I}}(\lambda)$ succeeds the simulated game with X'_{I}, i.e., $\mathrm{A}'_{\mathrm{I}}(\lambda)$ provides a valid proxy signature, and X'_{I} 's selections of five indices $\mathrm{I}_P, \mathrm{I}_D, \vartheta_P, \vartheta_D, \Lambda$ and two Boolean values $\mathrm{B}\beta_P, \mathrm{B}\beta_D$ are correct for the final result $(\mathrm{ID}_O^*, \mathrm{P}_O^*, \mathrm{K}_O^*, \delta_O^*, \mathrm{ID}_P^*, \mathrm{P}_P^*, \mathrm{K}_P^*, \mathrm{ID}_D^*, \mathrm{P}_D^*, \mathrm{K}_D^*, m^*, \sigma_{m^*})$ of $\mathrm{A}'_{\mathrm{I}}(\lambda)$, i.e., the target proxy signer with ID_P^* is the I_P-th user, the target designated verifier with ID_D^* is the I_D-th user, $(\mathrm{P}_P^*, \mathrm{K}_P^*)$ is the ϑ_P-th full public key of the target proxy signer, $(\mathrm{P}_D^*, \mathrm{K}_D^*)$ is the ϑ_D-th full public key of the target designated verifier, $\mathrm{H}_3(m^* \| \delta_O^* \| \mathrm{ID}_P^* \| \mathrm{P}_P^* \| \mathrm{K}_P^* \| \mathrm{ID}_D^* \| \mathrm{P}_D^* \| \mathrm{K}_D^*)$ is the Λ-th new query to H_3, $\mathrm{B}\beta_P$=0 in the case $(\mathrm{ID}_P^*, \mathrm{K}_P^*) \notin \mathbf{P}$, $\mathrm{B}\beta_P$=1 in the case $(\mathrm{ID}_P^*, \mathrm{P}_P^*) \notin \mathbf{S} \cup \mathbf{R}$, $\mathrm{B}\beta_D$=0 in the case $(\mathrm{ID}_D^*, \mathrm{K}_D^*) \notin \mathbf{P}$, $\mathrm{B}\beta_D$=1 in the case $(\mathrm{ID}_D^*, \mathrm{P}_D^*) \notin \mathbf{S} \cup \mathbf{R}$.

Assume that X'_{I} does not abort and outputs the result.

If $B\beta_P$=0, then $K_P^* = [u]G$, c_P^* =⊥, $s_P^* \in Z_q^*$ and P_P^* =[s_P^*] G from the step (a) and (d) of $O_{\mathrm{CreateUse}}$. If $B\beta_P$=1, then $P_P^* = [u]G$, s_P^* =⊥, $k_P^* \in Z_q^*$, K_P^* =[k_P^*] G , c_P^* =($k_P^* + s \cdot h_P^*$) mod q from the step (a) and (b) of $O_{\mathrm{CreateUse}}$. If $B\beta_D$=0, then $K_D^* = [v]G$, c_D^* =⊥, $s_D^* \in Z_q^*$ and P_D^* =[s_D^*] G from the step (a) and (d) of $O_{\mathrm{CreateUse}}$. If $B\beta_D$=1, then $P_D^* = [v]G$, s_D^* =⊥, $k_D^* \in Z_q^*$, K_D^* =[k_D^*] G , c_D^* =($k_D^* + s \cdot h_D^*$) mod$q$ from the step (a) and (c) of $O_{\mathrm{CreateUse}}$. Hence, in running of $O_{\mathrm{VerifyProxySign}}$(ID_O^*, ID_P^*, ID_D^*, P_O^*, P_P^*, P_D^*, δ_O^*, m^*, σ_{m^*}), the step (m) is executed if $B\beta_P$=0 and $B\beta_D$=0, the step (n) is executed if $B\beta_P$=0 and $B\beta_D$=1, the step (o) is executed if $B\beta_P$=1 and $B\beta_D$=0, and the step (p) is executed if $B\beta_P$=1 and $B\beta_D$=1.

If X_I''s selections of Λ is correct for the final result provided by $A_I'(\lambda)$, the following equation holds.

$$[w]G = H_3(m^* \,\|\, \delta_O^* \,\|\, \mathrm{ID}_P^* \,\|\, P_P^* \,\|\, K_P^* \,\|\, \mathrm{ID}_D^* \,\|\, P_D^* \,\|\, K_D^*).$$

If $A_I'(\lambda)$ provides a valid proxy signature, the final result satisfies the following verification equation:

$$v_m^* = \hat{e}(R_P^*, P_{\mathrm{KGC}})^{t_D^*} \cdot \hat{e}([w]G, T_P^*)^{t_D^*}.$$

where σ_{m^*} =(R_P^*, v_m^*), $T_P^* = P_P^* + K_P^*$ +[h_P^*] P_{KGC} and t_D^* =($s_D^* + c_D^*$) mod$q$. From the above equation, we can obtain

$$\begin{aligned}
v_m^* &= \hat{e}(R_P^*, P_{\mathrm{KGC}})^{t_D^*} \cdot \hat{e}([w]G, T_P^*)^{t_D^*} \\
&= \hat{e}(R_P^*, [t_D^*][s]G) \cdot \hat{e}([w]G, [t_D^*][t_P^*]G) \\
&= \hat{e}(R_P^*, [s]T_D^*) \cdot \hat{e}([w]G, [(t_D^* \; t_P^*) \bmod q]G).
\end{aligned}$$

where t_P^* =($s_P^* + c_P^*$) mod$q$.

When $B\beta_{\Pi}$ =0 and $B\beta_{\Delta}$ =0, we can compute

$$\begin{aligned}
&\hat{e}([w]G, [(t_D^* \cdot t_P^*) \bmod q]G) \\
&\quad = \hat{e}([w]G, [((s_D^* + c_D^*) \cdot (s_P^* + c_P^*)) \bmod q]G) \\
&\quad = \hat{e}([w]G, [((s_D^* + v + s \cdot h_D^*) \cdot (s_P^* + c_P^*)) \bmod q]G) \\
&\quad = \hat{e}([w]G, [(s_D^* + s \cdot h_D^*) \bmod q][t_P^*]G + [(v \cdot (s_P^* + u + s \cdot h_P^*)) \bmod q]G) \\
&\quad = \hat{e}([w]G, [(s_D^* + s \cdot h_D^*) \bmod q]T_P^* + [(uv) \bmod q]G + [(s_P^* + s \cdot h_P^*) \bmod q][v]G)
\end{aligned}$$

$$= \hat{e}([w]G, [(s_D^* + s \cdot h_D^*) \bmod q]\, T_P^* + [(s_P^* + s \cdot h_P^*) \bmod q]\, K_D^*) \cdot \hat{e}([w]G, [(uv) \bmod q]\, G)$$

$$= \hat{e}([w]G, [(s_D^* + s \cdot h_D^*) \bmod q]\, T_P^* + [(s_P^* + s \cdot h_P^*) \bmod q]\, K_D^*) \cdot \hat{e}(G,G)^{uvw}.$$

Hence, from the following equation

$$h^* = \hat{e}([w]G, [(s_D^* + s \cdot h_D^*) \bmod q]\, T_P^* + [(s_P^* + s \cdot h_P^*) \bmod q]\, K_D^*) \cdot \hat{e}(R_P^*, [s]\, T_D^*)$$

and the above discussion, we can obtain

$$\hat{e}(G,G)^{uvw} = v_{m^*} / h^* .$$

Similarly, we can discuss other cases.

Secondly, when $A'(\lambda)$ is a type II adversary $A'_{II}(\lambda)$, we construct an algorithm X'_{II} that uses $A'_{II}(\lambda)$ and the DBDH oracle O_{DBDH} to solve the GBDH problem in the bilinear group pair (Γ_1, Γ') as follows:

Algorithm X'_{II}:

Input/*Output*: (These are the same as those in Algorithm X'_I.)

1. Select five indices $0 < I_P, I_D \le N_U, 0 \le \vartheta_P, \vartheta_D < N_{UKey}$ and $0 < \Lambda \le N_{H_3}$ randomly.

2. Simulate a challenger in EUF-CMA Game as follows:

Setup: Let $s \leftarrow_R Z_q^*, P_{KGC} \longleftarrow [s]G$, $Params := \{a, b, p, q, G, P_{KGC}\}$, $B_{Params} := a \,\|\, b \,\|\, G \,\|\, P_{KGC}$. Send $Params$ and s to $A'_{II}(\lambda)$. Initialize the lists $\mathbf{S}$ and $\mathbf{D}$ with empty list ($\emptyset$) respectively, and prepare a list $\mathbf{H}_1$ of 2-tuples, two lists $\mathbf{H}_2$ and $\mathbf{H}_3$ of 3-tuples and a list $\mathbf{L}$ of tuples in form of $(ID_U, P_U, K_U, s_U, c_U)$. Choose randomly $h_P^* \in Z_q^*$ and $h_D^* \in Z_q^*$ such that $h_P^* \ne h_D^*$. Let $Count_P := 0$ and $Count_D := 0$.

Attack: Answer $A'_{II}(\lambda)$'s queries to oracles as follows:

- $H_1(x)$, $H_2(y)$, $H_3(z)$: (These are the same as that in Algorithm X'_I.)
- $O_{CreateUse}(ID_U)$: Let $h_U \longleftarrow H_1(B_{Params} \| ID_U)$ and respond differently in the following cases:
  a. If there exists $(ID_U, *, *, *, *) \in \mathbf{L}$ (where * means for an arbitrary value), then search $\mathbf{L}$ for the latest item $(ID_U, *, K_U, *, c_U)$ indexed by ID_U to get K_U and c_U. Otherwise, choose $k_U \in Z_q^*$ randomly, let $K_U \longleftarrow [k_U]G$ and $c_U \longleftarrow (k_U + s \cdot h_U) \bmod q$.
  b. If $h_U = h_P^*$, then let $Count_P \longleftarrow Count_P + 1$. If $Count_P = \vartheta_P$, then let $P_U \longleftarrow [u]G$, put $(ID_U, P_U,$

$\mathrm{K}_U, \perp, \mathrm{c}_U$) in $\mathbf{L}$ and return (P_U, K_U). Otherwise, go forward.

c. If $h_U = h_D^*$, then let $\mathrm{Count}_D \leftarrow \mathrm{Count}_D + 1$. If $\mathrm{Count}_D = \vartheta_D$, then let $\mathrm{P}_U \leftarrow [v]G$, put ($\mathrm{ID}_U$, P_U, $\mathrm{K}_U, \perp, \mathrm{c}_U$) in $\mathbf{L}$ and return (P_U, K_U). Otherwise, go forward.

d. (This is the same as that in Algorithm X'_{I}.)

- $\mathrm{O}_{\mathrm{GetSecretKey}}, \mathrm{O}_{\mathrm{GetPublicKey}}, \mathrm{O}_{\mathrm{GetDelegation}}, \mathrm{O}_{\mathrm{GetDVPSign}}$: (These are the same as those in Algorithm X'_{I}.)
- $\mathrm{O}_{\mathrm{VerifyDVPSign}}$ (ID_O, P_O, K_O, δ_O, ID_P, P_P, K_P, ID_D, P_D, K_D, m, σ_m): Parse δ_O as (w_O, R_O, Δ_O) and check whether all input parameters comform to w_O. If not, return **False**. If the check is passed, follow the next steps:

  a–i. (These steps are the same as those in Algorithm X'_{I}.)

  j. Search $\mathbf{L}$ for the item ($\mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P, \mathrm{s}_P, \mathrm{c}_P$) indexed by ($\mathrm{ID}_P, \mathrm{P}_P, \mathrm{K}_P$) to get s_P and c_P. If the search does not succeed, return $\perp$. If $\mathrm{c}_P = \perp$, return $\perp$. If $\mathrm{s}_P \neq \perp$, return $v_m \overset{?}{=} \hat{e}([s]R_P + [(\mathrm{s}_P + \mathrm{c}_P) \bmod q] H_3, \mathrm{T}_D)$.

  k. Search $\mathbf{H_1}$ for the item ($B_{Params} \| \mathrm{ID}_D, h_D$) indexed by $B_{Params} \| \mathrm{ID}_D$ to get h_D. If the search do es not succeed, return $\perp$.

  l. Search $\mathbf{L}$ for the latest item ($\mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D, \mathrm{s}_D, \mathrm{c}_D$) indexed by ($\mathrm{ID}_D, \mathrm{P}_D, \mathrm{K}_D$) to get s_D and c_D. If the search does not succeed, return $\perp$. If $\mathrm{c}_D = \perp$, return $\perp$. If $\mathrm{s}_D \neq \perp$, return $v_m \overset{?}{=} \hat{e}(H_3, [(\mathrm{s}_D + \mathrm{c}_D) \bmod q] \mathrm{T}_P) \cdot \hat{e}(R_P, [s]\mathrm{T}_D)$.

  m. If $\mathrm{s}_P = \perp$ and $\mathrm{s}_D = \perp$, let $h \leftarrow \hat{e}(H_3, [\mathrm{c}_D] \mathrm{T}_P + [\mathrm{c}_P] \mathrm{K}_D) \cdot \hat{e}(R_P, [s]\mathrm{T}_D)$.

  n. return $\mathrm{O}_{\mathrm{DBDH}}([u]G, [v]G, [w]G, v_m / h)$.

Forgery: Wait until $A'_{\mathrm{II}}(\lambda)$ outputs a tuple (ID_O^*, P_O^*, K_O^*, δ_O^*, ID_P^*, P_P^*, K_P^*, ID_D^*, P_D^*, K_D^*, m^*, σ_{m^*}) as the final result of the game.

3. Check $A'_{\mathrm{II}}(\lambda)$'s final result and output the solution of the above instance (i.e., the input) as follows:

(1) (This is the same as that in Algorithm X'_{I}.)

(2) If the following boolean expression is true, then abort. (This case is called *Event* R_2.)

$$((\mathrm{ID}_P^*, \mathrm{P}_P^*) \in \mathbf{S} \vee (\mathrm{ID}_D^*, \mathrm{P}_D^*) \in \mathbf{S}).$$

(3) (This is the same as that in Algorithm X'_{I}.)

(The end of the algoritm $\mathrm{X}'_{\mathrm{II}}$)

Since the analysis of $\mathrm{X}'_{\mathrm{II}}$ is similar to the one of X'_{I}, we omit it. This completes the proof.

From Theorem 1 and 2, we can conclude that our CLDVPS scheme is strong existentially unforgeable against an adaptively chosen-message attack and an adaptively chosen-warrant attack in the random oracle model (ROM), under the assumption that the Computational Diffie–Hellman (CDH) Problem in an elliptic curve group and the Gap Bilinear Diffie–Hellman (GBDH) Problem in a bilinear group pair are hard.

Next, we consider the non-transferability of our CLDVPS scheme.

When we fix a challenge message m^*, the identities { ID_O^*, ID_P^*, ID_D^* } of an original singer, a proxy signer and a designated verifier, a valid delegation δ_O^*, the public keys P_O^*, K_O^*, P_P^*, K_P^*, P_D^*, K_D^*, and the corresponding private keys s_P^*, c_P^*, s_D^*, c_D^* for signing, based on submission of an adversary in NT Game (Definition 7), the following values are also fixed:

$$\mathrm{t}_P^* := (\mathrm{s}_P^* + \mathrm{c}_P^*) \bmod q,$$

$$\mathrm{t}_D^* := (\mathrm{s}_D^* + \mathrm{c}_D^*) \bmod q,$$

$$\mathrm{T}_P^* := \mathrm{P}_P^* + \mathrm{K}_P^* + [\mathrm{H}_1(B_{Params} \| \mathrm{ID}_P^*)]\, P_{\mathrm{KGC}},$$

$$\mathrm{T}_D^* := \mathrm{P}_D^* + \mathrm{K}_D^* + [\mathrm{H}_1(B_{Params} \| \mathrm{ID}_D^*)]\, P_{\mathrm{KGC}},$$

$$H_3 := \mathrm{H}_3(m^* \| \delta_O^* \| \mathrm{ID}_P^* \| \mathrm{P}_P^* \| \mathrm{K}_P^* \| \mathrm{ID}_D^* \| \mathrm{P}_D^* \| \mathrm{K}_D^*).$$

Then, the ensemble of valid signatures on m^* produced by the algorithm GenerateDVPSign is as follows:

$$\Sigma := \{(R_P, \hat{e}([r_P]P_{\mathrm{KGC}} + [\mathrm{t}_P^*]H_3, \mathrm{T}_D^*)) \mid r_P \leftarrow_{\mathrm{R}} \mathrm{Z}_q^*,\ R_P \longleftarrow [r_P]G\}.$$

Meanwhile, the the ensemble of valid simulated signatures on m^* produced by the simulation algorithm SimulateDVPSign is as follows:

$$\Sigma^{\mathrm{S}} := \{(R_D, \hat{e}(R_D, P_{\mathrm{KGC}})^{\mathrm{t}_D^*} \cdot \hat{e}(H_3, \mathrm{T}_P^*)^{\mathrm{t}_D^*}) \mid r_D \leftarrow_{\mathrm{R}} \mathrm{Z}_q^*,\ R_D \longleftarrow [r_D]G\}.$$

Considering the equations $\mathrm{T}_P^* = [\mathrm{t}_P^*]G$ and $\mathrm{T}_D^* = [\mathrm{t}_D^*]$, we can compute the following equations.

$$\hat{e}(R_D, P_{\mathrm{KGC}})^{\mathrm{t}_D^*} \cdot \hat{e}(H_3, \mathrm{T}_P^*)^{\mathrm{t}_D^*} = \hat{e}(R_D, [\mathrm{t}_D^*]P_{\mathrm{KGC}}) \cdot \hat{e}(H_3, [\mathrm{t}_D^*]\mathrm{T}_P^*)$$

$$= \hat{e}(R_D, [t_D^*][s]G) \cdot \hat{e}(H_3, [t_D^*][t_P^*]G)$$

$$= \hat{e}([s]R_D, [t_D^*]G) \cdot \hat{e}([t_P^*]H_3, [t_D^*]G)$$

$$= \hat{e}([r_D][s]G, \mathrm{T}_D^*) \cdot \hat{e}([t_P^*]H_3, \mathrm{T}_D^*)$$

$$= \hat{e}([r_D]P_{\mathrm{KGC}} + [t_P^*]H_3, \mathrm{T}_D^*).$$

That is, any element of Σ corresponds to only one element of Σ^{S} and vice versa. The randomness of any element in Σ is determined by $r_P \leftarrow_{\mathrm{R}} Z_q^*$ and the randomness of any element in Σ^{S} is also determined by $r_D \leftarrow_{\mathrm{R}} Z_q^*$. Therefore, the distributions of any element of Σ and any element of Σ^{S} are identical and any unconditional distinguisher A_t cannot determine whether the signature is created by the proxy signer or simulated by the designated verifier. Namely, we can obtain the following result.

Theorem 3. The proposed CLDVPS scheme is undconditionally (perfectly) non-transferable.

6. Conclusion

Certificateless designated verifier proxy signature (CLDVPS) stands for designated verifier proxy signature (DVPS) in the certificateless setting which is intermediate between traditional PKI and Identity-based setting and has neither the certificate management issue nor private key escrow problem.

In this paper, we formalize the definition of a CLDVPS scheme and the security model for CLDVPS schemes. We then proposed the first CLDVPS scheme and analyze its efficiency. We also prove that in the random oracle model, our scheme is strong existentially unforgeable against an adaptively chosen-message attack and an adaptively chosen-warrant attack under the assumption that the Computational Diffie–Hellman (CDH) Problem in an elliptic curve group and the Gap Bilinear Diffie–Hellman (GBDH) Problem in a bilinear group pair are hard, and it is unconditionally non-transferable.

To the best of our knowledge, the concept of certificateless designated verifier proxy signature (CLDVPS) has not been appeared in the literature but Identity-based DVPS (IBDVPS) schemes have been discussed. Our scheme is the first provably secure CLDVPS scheme.

We believe that our scheme satisfies the privacy of signer’s identity.

We leave constructing a non-delegatable CLDVPS scheme as future work. It is also interesting to construct a CLDVPS schemes without pairings and prove the security in the standard model.

References

[1] S. S. Al-Riyami, K. G. Paterson: Certificateless Public Key Cryptography. In: C. S. Laih (Eds.) Proceedings of the 9th International Conference on the Theory and Application of Cryptology and Information Security (*ASIACRYPT* 2003), LNCS 2894, pp. 452–473. Springer-Verlag (2003)

[2] T. Cao, D. Lin, R. Xue: ID-based Designated-Verifier Proxy Signatures. *IEE Proceedings-Communications*, 152(6), pp. 989–994 (2005)

[3] H. Chen, F.-T. Zhang, R.-S. Song, Certificateless Proxy Signature Scheme with Provable Security, *Journal of Software*, 20(3), pp. 692–701 (2009)

[4] Y.-C. Chen, C.-L. Liu, G. Horng, K.-C. Chen: A Provably Secure Certificateless Proxy Signature Scheme. *International Journal of Innovative Computing, Information and Control*, 7(9), pp. 5557–5569 (2011)

[5] Y. Chen, et al.: A Certificateless Strong Designated Verifier Signature Scheme with Non-Delegatability. *International Journal of Network Security*, 19(4), pp. 573–582 (2017)

[6] Z. Cheng, L. Chen: Certificateless Public Key Signature Schemes from Standard Algorithms. In: C. Su, H. Kikuchi (Eds.) Proceedings of the 14th International Conference on Information Security Practice and Experience (*ISPEC* 2018), LNCS 11125, pp. 179–197. Springer-Verlag (2018)

[7] J. Cui, J. Zhang, H. Zhong, R. Shi, Y. Xu: An efficient certificateless aggregate signature without pairings for vehicular ad hoc networks. *Information Sciences*, 451–452, pp. 1–15 (2018)

[8] J. Dai, X. Yang, J. Dong: Designated-receiver Proxy Signature Scheme for Electronic Commerce. In: *Proceedings of IEEE International Conference on Systems, Man and Cybernetics-SMC* 2003, pp. 384–389. IEEE Press (2003)

[9] I. Damgård, H. Haagh, R. Mercer, A. Nitulescu, C. Orlandi, S. Yakoubov: Stronger Notions and Constructions for Multi-Designated Verifier Signatures. *IACR Cryptology ePrint Archive: Report* 2019/1153, pp. 1–44 (2019)

[10] H. Du, Q. Wen: Efficient Certificateless Designated Verifier Signatures and Proxy Signatures. *Chinese Journal of Electronics*, 8(1), pp. 95–100 (2009)

[11] H. Du, Q. Wen: Certificateless Proxy Multi-signature. *Information Sciences*, 276, pp. 21–30 (2014)

[12] K. Emura, A. Miyaji, K. Omote: An Anonymous Designated Verifier Signature. In: S.-H. Heng and K. Kurosawa (Eds.) Proceedings of the 4th International Conference on Provable Security (*ProvSec* 2010), LNCS 6402, pp. 184–198, 2010

[13] M. Girault: Self-certified public keys. In: D. W. Davies (Eds.) Proceedings of Workshop on the Theory and Application Workshop on the Theory and Application (*EUROCRYPT* 1991), LNCS 547, pp. 490–497. Springer-Verlag (1991)

[14] D. He, J. Chen: An Efficient Certificateless Designated Verifier Signature Scheme. *International Arab Journal of Information Technology*, 10(4), pp. 389–396 (2013)

[15] J. Herranz: On the transferability of private signatures. *Information Sciences* 179 (2009) 1647–1656. doi:10.1016/j.ins.2009.01.023

[16] B. C. Hu, D. S. Wong, X. Deng: Certificateless signature: a new security model and an improved generic construction. *Design, Codes and Cryptography*, 42(2), pp. 109–126 (2007)

[17] X. Hu, W. Tan, H. Xu, J. Wang: Short and Provably Secure Designated Verifier Proxy Signature Scheme. *IET Information Security*, 10(2), pp. 69–79 (2016). doi: 10.1049/iet-ifs.2014.0434

[18] X. Hu, J. Wang, H. Xu, W. Tan: A Short and Highly Efficient Identity-based Designated Verifier Proxy Signature Scheme. *Security and Communication Networks*, 8, pp. 907–916 (2015)

[19] X. Hu, H. Xu, Y. Liu, J. Wang, W. Tan, X. Zhang: An efficient designated verifier signature scheme with pairing-free and low cost. *Security and Communication Networks*, 9, pp. 5724–5732 (2016) DOI: 10.1002/sec.1731

[20] Q. Huang, G. Yang, D. S. Wong, W. Susilo: Identity-based strong designated verifier signature revisited. *The Journal of Systems and Software* 84(1), pp. 120–129 (2011). doi:10.1016/j.jss.2010.08.057.

[21] X. Huang, W. Susilo, Y. Mu, W. Wu: Universal designated verifier signature without delegatability. In: P. Ning, S. Qing, and N. Li (Eds.): Proceedings 8th International Conference on Information and Communications Security (*ICICS* 2006), LNCS 4307, pp. 479–498, (2006). doi:10.1007/11935308_34.

[22] X. Huang, W. Susilo, Y. Mu, F. Zhang, Short (identity-based) strong designated verifier signature schemes, In: K. Chen, R. Deng, X. Lai, J. Zhou (Eds.) Proceedings of the 2nd International Conference on Information Security Practice and Experience (*ISPEC* 2006), LNCS 3903, pp. 214–225 (2006). doi:10.1007/11689522_20.

[23] X. Huang, et al.: Certificateless Designated Verifier Signature Schemes. In: Proceedings of 20th International Conference on Advanced Information Networking and Applications (*AINA*'06), pp. 15–19. IEEE Press (2006)

[24] SK H. Islam, G. P. Biswas: Design of an efficient ID-based short designated verifier proxy signature scheme. In: Proceedings of Recent Advances in Information Technology (*RAIT* 2012), pp. 1–6. IEEE Press (2012)

[25] SK H. Islam, G. P. Biswas: Provably Secure Certificateless Strong Designated Verifier Signature Scheme based on Elliptic Curve Bilinear Pairings. *Journal of King Saud University – Computer and Information Sciences*, 25(1), pp. 51–61 (2013)

[26] SK H. Islam, G. P. Biswas: A provably secure identity-based strong designated verifier proxy signature scheme from bilinear pairings. *Journal of King Saud University – Computer and Information Sciences*, 26(1), pp. 55–67 (2014) doi:10.1016/j.jksuci.2013.03.004.

[27] SK H. Islam, G. P. Biswas: Provably Secure and Pairing-Based Strong Designated Verifier Signature Scheme with Message Recovery. *Arabian Journal for Science and Engineering* 2015; 40(4): 1069–1080. doi:10.1007/s13369-015-1568-2.

[28] M. Jakobsson, K. Sako, R. Impagliazzo: Designated Verifier Proofs and Their Applications. In: U. Maurer (Eds.) Proceedings of 15th International Conference on the Theory and Application of Cryptographic Techniques (*EUROCRYPT* '96), LNCS 1070, pp. 143–154. Springer-Verlag (1996). doi:10.1007/3-540-68339-9_13.

[29] Z. Jin, Q. Wen: Certificateless Multi-Proxy Signature. *Computer Communications*, 34(3), pp. 344–352 (2011)

[30] B. Kang: Attacks on one designated verifier proxy signature scheme. *Journal of Applied Mathematics*, 2012, 6, pp. 1–6 (2012)

[31] B. Kang, C. Boyd, E. Dawson: Identity-based strong designated verifier signature schemes: Attacks and new construction. *Computers and Electrical Engineering*, 35(1), pp. 49–53 (2009), doi:10.1016/j.compeleceng.2008.05.004.

[32] A. Karati, SK H. Islam, G. P. Biswas: A Pairing-free and Provably Secure Certificateless Signature Scheme. *Information Sciences*,450, pp. 378–391 (2018)

[33] F. Laguillaumie, B. Libert, J. Quisquater: Universal designated verifier signatures without random oracles or non-black box assumptions. In: R. De Prisco and M. Yung (Eds.) Proceedings of 5th International Conference on Security and Cryptography for Networks, *SCN* 2006, LNCS 4116, pp. 63–77 (2006) doi:10.1007/11832072_5.

[34] F. Laguillaumie, D. Vergnaud: Multi-designated verifiers signatures. In: J. López, S. Qing, E. Okamoto (Eds.) Proceedings 6th International Conference on Information and Communications Security, *ICICS* 2004 (LNCS 3269), 495–507, (2004). doi:10.1007/978-3-540-30191-2_38.

[35] F. Laguillaumie, D. Vergnaud: Designated verifier signatures: anonymity and efficient construction from any bilinear map. In: C. Blundo, S. Cimato (Eds.) Proceedings of the 4th Conference on Security and Cryptography for Networks (*SCN* 2004), LNCS 3352, pp. 107–121 (2005)

[36] S. Lal, V. Verma: Identity Based Strong Designated Verifier Proxy Signature Schemes. *IACR Cryptology ePrint Archive: Report* 2006/394, pp. 1–11 (2006)

[37] S. Lal, V. Verma: Identity Based Strong Bi-Designated Verifier Proxy Signature Schemes. *IACR Cryptology ePrint Archive: Report* 2008/024, pp. 1–12 (2008)

[38] X. Li, K. Chen, L. Sun: Certificateless Signature and Proxy Signature Schemes from Bilinear Pairings. *Lithuanian Mathematical Journal*, 45(1), pp. 76–83 (2005)

[39] H. Y. Lin, P. Y. Ting, L. F. Yang: On the Security of a Provably Secure Certificateless Strong Designated Verifier Signature Scheme based on Bilinear Pairings. In: *Proceedings of the International Conference on Telecommunications and Communication Engineering* (*ICTCE*'17), pp. 61–65. IEEE Computer Society (2017)

[40] H. Lipmaa, G. Wang, F. Bao: Designated verifier signature schemes: attacks, new security notions and a new construction, In: Proceedings of 32nd International Colloquium on Automata, Languages and Programming, *ICALP* 2005 (LNCS 3580), pp. 459–471 (2005), doi:10.1007/11523468_38.

[41] R. Lu, Z. Cao: Designated Verifier Proxy Signature Scheme with Message Recovery. *Applied Mathematics and Computation*, 169(2) pp. 1237–1246 (2005)

[42] R. Lu, D. He, C. Wang: Cryptanalysis and Improvement of a Certificateless Proxy Signature Scheme from Bilinear Pairings. In: Proceedings of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (*SNPD* '07), pp. 285–290. IEEE Computer Society (2007)

[43] Y. Lu, J. Li: Provably Secure Certificateless Proxy Signature Scheme in the Standard Model. *Theoretical Computer Science*, 639, pp. 42–59 (2016)

[44] M. Mambo, K. Usuda, and E. Okamoto: Proxy Signatures: Delegation of the Power to Sign Messages. *IEICE Transactions on Fundamentals of Electronic Communications and Computer Science*, E79-A (9), pp. 1338–1354 (1996)

[45] S. Padhye, N. Tiwari: ECDLP-based Certificateless Proxy Signature Scheme with Message Recovery. *Transactions on Emerging Telecommunications Technologies*, 26, pp. 346–354 (2015)

[46] N. Pakniat: On the Security of a Certificateless Strong Designated Verifier Signature Scheme. *IACR Cryptology ePrint Archive: Report* 2018/915, pp. 1–20 (2018)

[47] P. Rastegari, M. Berenjkoub: Multi-Designated Verifiers Signature Schemes with Threshold Verifiability. *IACR Cryptology ePrint Archive: Report* 2017/797, pp. 1–20 (2017).

[48] P. Rastegari, W. Susilo, M. Dakhilalian: Certificateless designated verifier signature revisited: achieving a concrete scheme in the standard model. *International Journal of Information Security*, 18 (5), pp. 619–635 (2019) doi:10.1007/s10207-019-00430-5.

[49] P. Rastegari, M. Berenjkoub, M. Dakhilalian, W. Susilo: Universal designated verifier signature scheme with non-delegatability in the standard model. *Information Sciences* 479 (2019) 321–334. doi: 10.1016/j.ins.2018.12.020.

[50] R. L. Rivest, A. Shamir, Y. Tauman: How to Leak a Secret. In: C. Boyd (Eds.): Proceedings of 7th International Conference on the Theory and Application of Cryptology and Information Security,

(*ASIACRYPT* 2001), LNCS 2248, pp. 552–565, 2001.

[51] A. Shamir: Identity-based Cryptosystems and Signature Schemes. In: *CRYPTO* '84 (LNCS 196), pp. 47–53. Springer-Verlag (1984)

[52] N. Sharma, R. A. Sahu, V. Saraswat, B. K. Sharma: Adaptively Secure Strong Designated Signature. In: O. Dunkelman and S.K. Sanadhya (Eds.): Proceedings of the 17th International Conference on Cryptology in India (*INDOCRYPT* 2016), LNCS 10095, pp. 43–60, 2016. doi: 10.1007/978-3-319-49890-4 3

[53] W. Shi, D. He, P. Gong: On the Security of a Certificateless Proxy Signature Scheme with Message Recovery. *Mathematical Problems in Engineering*, 2013, Article ID 761694, pp. 1–4 (2013)

[54] R. Steinfeld, L. Bull, H. Wang, J. Piperzyk: Universal designated-verifier signatures. In: C. S. Laih (Eds.) Proceedings of the 9th International Conference on the Theory and Application of Cryptology and Information Security (*ASIACRYPT* 2003), LNCS 2894, pp. 523–543. Springer-Verlag (2003), doi:10.1007/978-3-540-40061-5_33.

[55] H. Tian, X. Chen, Z. Jiang, Y. Du: Non-delegatable strong designated verifier signature on elliptic curves. In: H. Kim (Eds): Revised Selected Papers of the 14th International Conference on Information Security and Cryptology (*ICISC* 2011), LNCS 7259, pp. 219–234, doi:10.1007/978-3-642-31912-9_15.

[56] H. Tian, X. Chen, J. Li: A Short Non-delegatable Strong Designated Verifier Signature. In: W. Susilo, Y. Mu, and J. Seberry (Eds.): Proceedings of the 17th Australasian Conference on Information Security and Privacy (ACISP 2012), LNCS 7372, pp. 261–279, 2012.

[57] H. Tian, X. Chen, F. Zhang, B. Wei, Z. Jiang, Y. Liu: A non-delegatable strong designated verifier signature in ID-based setting for mobile environment. *Mathematical and Computer Modelling* 58 (2013) 1289–1300. doi:10.1016/j.mcm.2013.01.010.

[58] M. Tian, W. Yang, L. Huang: Cryptanalysis and Improvement of a Certificateless Multi-Proxy Signature Scheme. *IACR Cryptology ePrint Archive: Report* 2011/379, pp. 1–11 (2011)

[59] G. K. Verma, et al.: Bandwidth Efficient Designated Verifier Proxy Signature Scheme for Healthcare Wireless Sensor Networks. *Ad Hoc Networks*, 81, pp. 100–108 (2018)

[60] Z. Wan, X. Lai, J. Weng, X. Hong, Y. Long, W.-W. Jia: On Constructing Certificateless Proxy Signature from Certificateless Signature. *Journal of Shanghai Jiaotong University (Science)*, 13(6), pp. 692–694 (2008) doi: 10.1007/s12204-008-0692-5

[61] G. Wang: Designated-Verifier Proxy Signatures for E-Commerce. In: 2004 IEEE International Conference on Multimedia and Expo (*ICME* 2004), pp. 1731–1734. IEEE Computer Society (2004)

[62] H. Xiong, F. Li, Z. Qin: A Provably Secure Proxy Signature Scheme in Certificateless Cryptography. *Informatica*, 21(2), pp. 277–294 (2010)

[63] J. Xu, H. Sun, Q. Wen, H. Zhang: Improved Certificateless Multi-Proxy Signature. *The Journal of China Universities of Posts and Telecommunications* 19(4), pp. 94–105 (2012)

[64] B. Yang, Z. Hu, Z. Xiao: Efficient Certificateless Strong Designated Verifier Signature Scheme. In: Proceedings of 2009 5th International Conference on Computational Intelligence and Security (*CIS* '09), pp. 432–436. IEEE Press (2009) doi 10.1109/CIS.2009.191.

[65] Y. Yang: ID-based designated-verifier proxy signature scheme without a trusted party. In: Proceedings of the International Conference on Computer Application and System Modelling (*ICCASM* '10), vol. 7, pp. 191–193. IEEE Press (2010)

[66] W. Yap, S. Heng, B. Goi: Cryptanalysis of some proxy signature scheme without certificates. In: Proceedings of the 1st Workshop on Information Security Theory and Practices (*WISTP* '07) (LNCS 4462), pp. 115–126. Springer-Verlag (2007)

[67] K. Yoneyama, M. Ushida, K. Ohta: Rigorous security requirements for designated verifier signatures. In: X.

Lai, M. Yung, D. Lin (Eds.) Revised Selected Papers of the 6th International Conference 6th International Conference (*Inscrypt* 2010), LNCS 6584, pp. 318–335. Springer-Verlag (2011)

[68] Y. Yu, C. Xua, X. Zhang, Y. Liao: Designated Verifier Proxy Signature Scheme without Random Oracles. *Computers & Mathematics with Applications*, 57(8), pp. 1352–1364 (2009)

[69] J. Zhang: On the Security of a Designated-Verifier Proxy Signature Scheme and Its Improved Scheme (revisited). In: Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing, 2007 (*WiCom* 2007), pp. 2196–2199. IEEE Computer Society (2007)

[70] L. Zhang, F. Zhang, Q. Wu: Delegation of Signing Rights using Certificateless Proxy Signatures. *Information Sciences*, 184, pp. 298–309 (2012)