SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

January 2021, Vol. 64 112101:1–112101:15
https://doi.org/10.1007/s11432-019-2903-2

Certificateless designated verifier proxy signature scheme for unmanned aerial vehicle networks

Lei HE[1,2], Jianfeng MA[3*], Limin SHEN[4] & Dawei WEI[1]

[1]*School of Computer Science and Technology, Xidian University, Xi'an* 710071*, China;*
[2]*School of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou* 450000*, China;*
[3]*School of Cyber Engineering, Xidian University, Xi'an* 710071*, China;*
[4]*School of Computer Science and Technology, Nanjing Normal University, Nanjing* 210023*, China*

Received 8 August 2019/Revised 16 November 2019/Accepted 12 May 2020/Published online 17 December 2020

Abstract Unmanned aerial vehicle (UAV) technologies have a promising application prospect. It is necessary to use digital signature schemes to protect the integrity and authentication of messages in the UAV networks. According to the characteristics of UAV networks, digital signature schemes should solve the problems of digital certificate management and key escrow, guarantee the real-time performance of UAV executing commands, and ensure that only the designated verifier can verify the signature. Therefore, we proposed a certificateless designated verifier proxy signature (CLDVPS) scheme which meets the requirements. We proved our signature scheme is existentially unforgeable under an adaptive chosen message attack in the random oracle model and compared the efficiency of our signature scheme with other signature schemes. We implemented these signature schemes in jPBC to obtain the computing time and theoretically analyzed signature length of these signature schemes. The analysis results indicate that our CLDVPS scheme is more efficient than most of other related signature schemes on computation costs and also efficient on communication costs.

Keywords certificateless, designated verifier, proxy, digital signature, unmanned aerial vehicle

Citation He L, Ma J F, Shen L M, et al. Certificateless designated verifier proxy signature scheme for unmanned aerial vehicle networks. Sci China Inf Sci, 2021, 64(1): 112101, https://doi.org/10.1007/s11432-019-2903-2

1 Introduction

Unmanned aerial vehicles (UAVs) has been widely used in many applications with the development of UAV technology. A UAV is low cost and has good maneuverability and flexible use. We informally classify UAVs into three categories according to the degree of autonomy, namely, UAVs under the control of remote operator, UAVs under the supervision of remote supervisor, and UAVs without operator and supervisor. The third type of UAVs does not require real-time control or supervision. It is equipped with sensors and onboard computers. The sensors monitor changes in the internal and external conditions in real time and the onboard computers respond to these changes. When a commander issues missions, the UAV will automatically complete the missions. It can perform tasks, such as surveillance and reconnaissance, for a long time without considering the operator or supervisor fatigue. It has better autonomy and availability. Therefore, the UAVs in this paper mainly refer to the third type of UAVs.

A UAV network usually refers to a network with UAV applications as its core, which has some characteristics, such as high mobility, dynamic topology, intermittent link, power constraints, and changing link quality [1]. A UAV network may include UAVs, a command center, satellites, ground control stations, and mobile command vehicles. It is illustrated as Figure 1. A UAV belongs to a command center which has the highest command authority for UAVs. The entities with authority to command UAV are called command stations, such as command centers, satellites, and ground control stations. When a command station wants to send commands to direct UAVs to perform tasks, it needs to prove that the commands have not been tampered and are from an authorized command station. It usually uses a digital signature

[*] Corresponding author (email: jfma@mail.xidian.edu.cn)

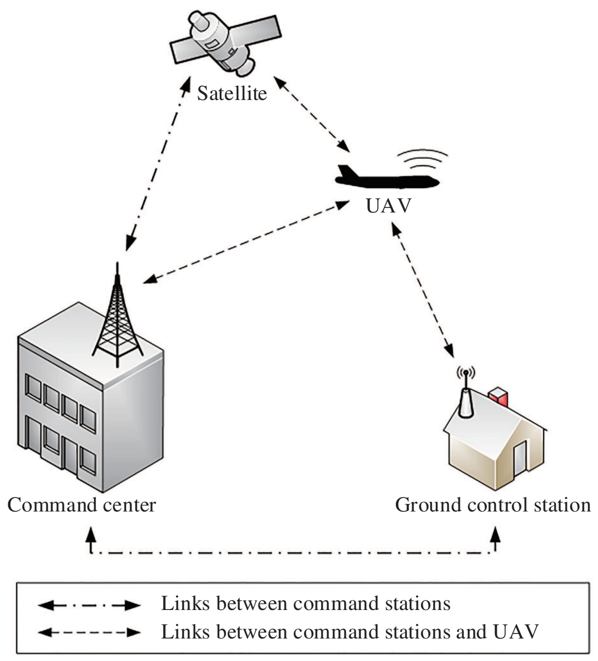
Satellite
UAV
Command center
Ground control station
Links between command stations
Links between command stations and UAV

Figure 1 (Color online) UAV network.

scheme to protect the integrity and authentication of commands. A command station computes the signature of command and sends {command, signature} to the UAV. After receiving the message, the UAV will verify the signature and confirm the integrity and authentication of the command. Afterwards, it decides whether to execute the command.

1.1 Problem statement

Most of digital signature schemes are based on digital certificates, while certificate authority (CA) ensures the correlation between public key and user's identity. A CA manages digital certificates and maintains a certificate revocation list (CRL). When there are a large number of users, the costs of managing certificates by CA and users will be huge. A CA needs to update CRL in time and send the latest CRL to UAVs. However, it is difficult for UAVs to receive the latest CRL in time because of intermittent link and changing link quality of UAV networks. Hence, a public key infrastructure (PKI) architecture based on CA is difficult to adapt to the characteristics of UAV networks. It is necessary to design a digital signature scheme without digital certificate to protect authentication and integrity in the UAV networks. Some researchers proposed identity-based digital signature schemes. However, a key generation center (KGC) knows the private keys of all users in an identity-based digital signature scheme. If an adversary obtains the user's private key from KGC, or if KGC uses the user's private key without authorization, the user will be impersonated to sign. It inevitably generates key escrow problem. To solve this problem, some researchers proposed certificateless digital signature schemes which do not have digital certificate management problems and key escrow problems.

A UAV has longer flight distance and executes remote missions in some cases, which results in that UAVs may be far away from command center and have longer communication delay, or even be completely incapable of communicating with the command center. If a UAV cannot receive and verify command in time, it may fly away from the target. When the UAV wants to execute this command, it will have to approach the target again, which wastes the power of UAV. Therefore, it is necessary to guarantee the real-time performance of executing commands by UAVs. In order to ensure a UAV executes commands in time, a command center may authorize other entities which are close to the UAV as agents. These agents can temporarily command the UAV. For example, the command center authorizes a ground control station (GCS) as an agent and the GCS directly issues commands to a UAV, which can be implemented by a proxy signature scheme.

An adversary can eavesdrop on the messages exchanged among UAVs and command stations and verify the signatures. It can confirm the next tasks of UAV and prepare in advance. Hence, it is necessary to ensure that the signature can only be verified by the designated verifier, namely, the UAV. It can satisfy this requirement by using a designated verifier signature scheme.

In summary, we have drawn the following conclusions through above analysis.

• It needs to use a certificateless signature scheme to solve the problems of digital certificate management and key escrow.

• It needs to use a proxy signature scheme to guarantee the real-time performance of UAV executing commands.

• It needs to use a designated verifier signature scheme to ensure that only the designated verifier can verify the signature.

Therefore, we propose a certificateless designated verifier proxy signature (CLDVPS) scheme for UAV networks to meet these requirements.

1.2 Our contributions

In this paper, we propose a CLDVPS scheme and analyze it. Our contributions mainly include:

(1) We analyze the requirements of digital signature scheme for UAV networks.

(2) We propose a CLDVPS scheme to meet the requirements of digital signature scheme for UAV networks. Our CLDVPS scheme can solve the problems of certificate management and key escrow. It can delegate a proxy signer to compute signatures of a message on behalf of original signer, which helps to guarantee the real-time performance of UAV executing commands. It also ensures that only the designated verifier can verify the signature.

(3) We analyze the security of our CLDVPS scheme in the random oracle model. The results indicate that our scheme has existential unforgeability under an adaptive chosen message attack (EUF-CMA).

(4) We implement our CLDVPS scheme and other related signature schemes by using jPBC and obtain experiment data. We also compare the signature length of these signature schemes. The analysis indicates that our scheme is more efficient than most of other related signature schemes on computation costs and also efficient on communication costs.

1.3 Organization of the remainder paper

The rest of this paper is organized as follows. In Section 2, we mainly review some certificateless digital signature schemes. In Section 3, we introduce some necessary preliminaries. We propose the CLDVPS scheme for UAV networks in Section 4. Afterwards, we analyze the security and efficiency of our CLDVPS scheme in Section 5. Finally, we conclude this paper in Section 6.

2 Related work

In a digital signature scheme, every user has a pair of public key and private key. A signer uses its private key to generate a digital signature, while a verifier uses the public key of signer to verify this signature. In such a digital signature scheme, it must ensure the authenticity and validity of user's public key. The traditional solution is to use a PKI. However, the management and maintenance of public key certificate library require a large number of computation, communication and storage costs. In 1984, Shamir proposed the identity-based cryptography, which avoids using and verifying certificate in the traditional PKI to solve the problem of user's public key authenticity [2]. However, there is inevitably an inherent flaw in such a system, namely, the problem of private key escrow. The concept of certificateless public key cryptography was proposed by Al-Riyami and Paterson [3]. A certificateless public key cryptography does not require public key certificate and solves the problem of private key escrow in an identity-based cryptography.

Al-Riyami and Paterson proposed the first certificateless signature scheme but they did not provide a formal security analysis for the scheme. Huang et al. [4] analyzed this signature scheme and considered it cannot resist public key replacement attacks. He et al. [5], Wang et al. [6], and Karati et al. [7] respectively proposed certificateless signature schemes without bilinear pairing and analyzed their security in the random oracle model. Yeh et al. [8] proposed a certificateless signature scheme for smart objects in the Internet-of-Thing (IoT) environments. Jia et al. [9] analyzed the certificateless signature scheme proposed by Yeh et al. and pointed out there are two shortcomings. In this scheme, an adversary can impersonate the key generation center to issue partial private keys. Moreover, this scheme cannot resist public key replacement attacks. Afterwards, Jia et al. [9] proposed an improved certificateless signature scheme and proved it is secure in the random oracle model. Deng et al. [10] proposed an efficient certificateless signature scheme in the standard model. Du et al. [11] introduced outsourced computation into certificateless signature scheme and proposed an outsourced revocable certificateless signature scheme which outsources revocation functionality to a cloud server. It has been proved that the scheme is secure under the elliptic curve discrete logarithm problem (ECDLP). Karati et al. [12] proposed a lightweight certificateless signature scheme for industrial Internet of Things (IIoT).

Mambo et al. [13] proposed the proxy signature scheme. Jakobsson et al. [14] proposed the concept of designated verifier signature scheme in 1996. A designated verifier signature scheme provides authentication, but does not provide non-repudiation. The signature in the designated verifier signature scheme can only be verified by the designated verifier. Some researchers proposed certificateless proxy signature (CLPS) scheme, certificateless designated verifier signature (CLDVS) scheme, and CLDVPS scheme. Chen et al. [15] proposed a provably secure CLPS scheme. Zhang et al. [16, 17] proposed two CLPS schemes. Seo et al. [18] proposed an efficient CLPS scheme with provable security. He et al. [19] proposed a CLPS scheme without pairing and proved it is secure in the random oracle model. Padhye and Tiwari [20] proposed a CLPS scheme with message recovery based on ECDLP. It is proved secure in the random oracle model. However, Shi et al. [21] showed that it is not secure against Type I adversary. Lu and Li [22] proposed a provably secure CLPS scheme in the standard model. Huang et al. [23] and Chen et al. [24] proposed CLDVS schemes and proved the schemes are secure in the random oracle model, respectively. Du and Wen [25] proposed a CLDVS scheme and a CLDVPS scheme. Xiao et al. [26] proposed a certificateless strong designated verifier signature scheme and proved it is secure in the random oracle model. He and Chen [27] proposed an efficient CLDVS scheme without bilinear pairing and proved it is secure in the random oracle model. Lin [28] defined a formal security model of strong signer ambiguity against key-compromise attacks (SSA-KCA) for certificateless strong designated verifier signature scheme and proposed a signature scheme which meets the SSA-KCA and EUF-CMA security.

3 Preliminaries

3.1 Bilinear pairing

Let G_1 be an additive cyclic group with prime order q and G_2 be a multiplicative cyclic group with the same order q. A generator of G_1 is P. A map $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear pairing if it meets the following requirements:

• Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in Z_q^*$.

• Non-degeneracy: It exists $P, Q \in G_1$ such that $e(P, Q) \neq 1$.

• Computability: There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.

3.2 Complexity assumptions

Definition 1 (Computational bilinear Diffie-Hellman (CBDH) problem)**.** Given $P, aP, bP, cP \in G_1$ for unknown $a, b, c \in Z_q^*$, compute $e(P, P)^{abc}$.

3.3 Certificateless designated verifier proxy signature scheme

A CLDVPS scheme is mainly composed of the following algorithms.

• Setup: This algorithm is performed by KGC. It takes the security parameter as input and outputs a master key s and system parameters `params`.

• Partial-Private-Key-Extract: This algorithm is performed by KGC. It takes the `params`, s, and user's identity IDs as inputs and outputs the user's partial private key $D_{\rm ID}$.

• Set-Secret-Value: This algorithm is performed by the user. It takes the `params` and the security parameter as inputs and outputs the user's secret value $x_{\rm ID}$.

• Set-Private-Key: This algorithm is performed by the user. It takes the `params`, $x_{\rm ID}$ and $D_{\rm ID}$ as inputs and outputs the user's private key $\rm SK_{ID}$.

• Set-Public-Key: This algorithm is performed by the user. It takes the `params` and $x_{\rm ID}$ as inputs and outputs the user's public key $\rm PK_{ID}$.

• Generate-Delegation: This algorithm is performed by the original signer (OS). It takes the `params`, warrant w, $x_{\rm OS}$, identity of OS ${\rm ID_{OS}}$, identity of proxy signer (PS) ${\rm ID_{PS}}$, ${\rm PK_{OS}}$, and ${\rm PK_{PS}}$ as inputs and outputs the `delegation` from OS to PS.

• Verify-Delegation: This algorithm is performed by the PS. It takes the `params`, $x_{\rm PS}$, w, `delegation`, ${\rm ID_{OS}}$, ${\rm ID_{PS}}$, ${\rm PK_{OS}}$, and ${\rm PK_{PS}}$ as inputs and outputs private key for proxy signature, skp.

• Sign: This algorithm is performed by the PS. It takes the `params`, w, skp, ${\rm ID_{OS}}$, ${\rm ID_{PS}}$, ${\rm PK_{OS}}$, ${\rm PK_{PS}}$, $D_{\rm PS}$, public key of designated verifier (DV) ${\rm PK_{DV}}$, and message m as inputs and outputs the signature U.

• Verify: This algorithm is performed by the DV. It takes the `params`, w, public key for proxy signature PKp, $x_{\rm DV}$, ${\rm ID_{OS}}$, ${\rm ID_{PS}}$, ${\rm PK_{OS}}$, ${\rm PK_{PS}}$, U, and message m as inputs and outputs the result of verifying. It outputs `True` if the signature is correct, or `False` otherwise.

3.4 Security model of certificateless designated verifier proxy signature scheme

There are two kinds of adversaries in the certificateless signature scheme according to [29]. They are Type I adversary $\mathcal{A}_{\mathcal{I}}$ and Type II adversary $\mathcal{A}_{\mathcal{II}}$. An adversary $\mathcal{A}_{\mathcal{I}}$ simulates the attacks where $\mathcal{A}_{\mathcal{I}}$ can replace the user's public key ${\rm PK_{ID}}$ but is not given the user's partial private key $D_{\rm ID}$. An adversary $\mathcal{A}_{\mathcal{II}}$ simulates the attacks where $\mathcal{A}_{\mathcal{II}}$ can have the master key s but cannot replace the target user's public key. The security of CLDVPS scheme is formally defined through the following games played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A} \in \{\mathcal{A}_{\mathcal{I}}, \mathcal{A}_{\mathcal{II}}\}$.

• Setup: $\mathcal{C}$ runs this algorithm and takes a security parameter as input to obtain a master key s and system parameter `params`. $\mathcal{C}$ sends `params` to the adversary $\mathcal{A}$ who can access to the following oracles. If $\mathcal{A}$ is $\mathcal{A}_{\mathcal{II}}$, $\mathcal{C}$ also sends the master key s to $\mathcal{A}$.

• Create-User-Oracle: This oracle takes a user's identity ID as input. If this ID has existed in the User-list, it does nothing. Otherwise, it generates the $D_{\rm ID}$, $x_{\rm ID}$, and ${\rm PK_{ID}}$ and adds (ID, $x_{\rm ID}$, $D_{\rm ID}$, ${\rm PK_{ID}}$) to the User-list.

• Partial-Private-Key-Oracle: This oracle takes a user's identity ID as input and searches the User-list. It returns corresponding partial private key $D_{\rm ID}$. Note that $\mathcal{A}_{\mathcal{II}}$ does not need to access this oracle because it has the master key s.

• Secret-Value-Oracle: This oracle takes a user's identity ID as input and searches the User-list. It returns corresponding secret value $x_{\rm ID}$.

• Public-Key-Oracle: This oracle takes a user's identity ID as input and searches the User-list. It returns corresponding public key ${\rm PK_{ID}}$.

• Public-Key-Replacement-Oracle: This oracle takes a user's identity ID and a new public key ${\rm PK'_{ID}}$ as inputs. It replaces the public key ${\rm PK_{ID}}$ with the new one ${\rm PK'_{ID}}$ and updates corresponding information in the User-list.

• Proxy-Private-Key-Generation-Oracle: This oracle takes ${\rm ID_{OS}}$, ${\rm ID_{PS}}$, ${\rm PK_{OS}}$, and ${\rm PK_{PS}}$ as inputs and searches the proxy private key list ProPriK-list. It returns corresponding private key for proxy signature ${\rm skp_{ID}}$.

• Proxy-Public-Key-Generation-Oracle: This oracle takes ${\rm ID_{OS}}$, ${\rm ID_{PS}}$, ${\rm PK_{OS}}$, and ${\rm PK_{PS}}$ as inputs and searches the proxy public key list ProPubK-list. It returns corresponding public key for proxy signature ${\rm PKp_{ID}}$.

• Sign-Oracle: This oracle takes ${\rm ID_{OS}}$, ${\rm ID_{PS}}$, ${\rm PK_{OS}}$, ${\rm PK_{PS}}$, a warrant w, and a message m as inputs. It computes the signature, adds this signature to the Sign-list, and returns this signature.

• Verify-Oracle: This oracle takes ${\rm ID_{OS}}$, ${\rm ID_{PS}}$, ${\rm PK_{OS}}$, ${\rm PK_{PS}}$, a warrant w, a message m, and a signature U as inputs and returns the verification result. It outputs `True` if the signature is valid, or `False` otherwise.

Finally, $\mathcal{A}$ outputs a tuple (m^*, w^*, ${\rm ID^*_{PS}}$, U^*) as its forgery.

We say $\mathcal{A}$ wins this game if the forgery satisfies the following requirements.

(1) For the adversary $\mathcal{A}_{\mathcal{I}}$:

• $\mathcal{A}_{\mathcal{I}}$ has never submitted $({\rm ID^*_{OS}}, {\rm ID^*_{PS}}, m^*, w^*)$ to the Sign-Oracle and $({\rm ID^*_{OS}}, {\rm ID^*_{PS}}, m^*, w^*, U^*)$ to the Verify-Oracle;

• $\mathcal{A}_{\mathcal{I}}$ has never submitted ${\rm ID^*_{OS}}$ or ${\rm ID^*_{PS}}$ to Partial-Private-Key-Oracle and Secret-Value-Oracle;

• $\mathcal{A}_{\mathcal{I}}$ has never submitted $({\rm ID^*_{OS}}, {\rm ID^*_{PS}})$ with a warrant w^* to Proxy-Private-Key-Generation-Oracle, i.e., ${\rm ID^*_{PS}}$ was not designated by ${\rm ID^*_{OS}}$ as a proxy signer;

• The signature U^* of m^* is valid.

(2) For the adversary $\mathcal{A}_{\mathcal{II}}$:

• $\mathcal{A}_{\mathcal{II}}$ has never submitted $(\mathrm{ID}_{\mathrm{OS}}^*, \mathrm{ID}_{\mathrm{PS}}^*, m^*, w^*)$ to the Sign-Oracle and $(\mathrm{ID}_{\mathrm{OS}}^*, \mathrm{ID}_{\mathrm{PS}}^*, m^*, w^*, U^*)$ to the Verify-Oracle;

• $\mathcal{A}_{\mathcal{II}}$ has never submitted $\mathrm{ID}_{\mathrm{OS}}^*$ or $\mathrm{ID}_{\mathrm{PS}}^*$ to Secret-Value-Oracle and Public-Key-Replacement-Oracle;

• $\mathcal{A}_{\mathcal{II}}$ has never submitted $(\mathrm{ID}_{\mathrm{OS}}^*, \mathrm{ID}_{\mathrm{PS}}^*)$ with a warrant w^* to Proxy-Private-Key-Generation-Oracle, i.e., $\mathrm{ID}_{\mathrm{PS}}^*$ was not designated by $\mathrm{ID}_{\mathrm{OS}}^*$ as a proxy signer;

• The signature U^* of m^* is valid.

The success probability of $\mathcal{A}_{\mathcal{I}}$ to win the game is denoted by $\mathrm{Succ}_{\mathcal{A}_{\mathcal{I}}}^{\text{EUF-CMA}}$. Similarly, the success probability of $\mathcal{A}_{\mathcal{II}}$ to win the game is denoted by $\mathrm{Succ}_{\mathcal{A}_{\mathcal{II}}}^{\text{EUF-CMA}}$.

Definition 2. A CLDVPS scheme is EUF-CMA (or EUF-CMA secure) if the success probabilities of any polynomially bounded adversaries $\mathcal{A}_{\mathcal{I}}$ and $\mathcal{A}_{\mathcal{II}}$ in the above game is negligible, namely $\mathrm{Succ}_{\mathcal{A}_{\mathcal{I}}}^{\text{EUF-CMA}} \leqslant \varepsilon$ and $\mathrm{Succ}_{\mathcal{A}_{\mathcal{II}}}^{\text{EUF-CMA}} \leqslant \varepsilon$, where ε is negligible.

4 CLDVPS scheme for UAV networks

We describe our certificateless designated verifier proxy signature scheme for UAV networks in this section. A command center acts as original signer and sets security parameters. It delegates an entity, such as ground control station, to sign on behalf of the command center, that is, the ground control station is proxy signer. The UAV acts as designated verifier. For the convenience of presentation, we use A, B, and C to represent the original signer, proxy signer, and designated verifier, respectively. Our scheme consists of the following phases.

Setup: Given a security parameter l, it generates two groups G_1 and G_2 which have the same prime order q and a bilinear pairing $e: G_1 \times G_1 \rightarrow G_2$. $\boldsymbol{P}$ is a generator of group G_1. It selects three hash functions $H_1: \{0,1\}^* \rightarrow G_1$, $H_2: G_1 \times G_1 \times \{0,1\}^* \rightarrow Z_q^*$, and $H_3: \{0,1\}^* \times G_1 \times G_1 \rightarrow Z_q^*$. It selects a random secret $s \in Z_q^*$ as the master key and computes the system public key $P_0 = sP$. It lists the system parameters $\mathtt{params} = (G_1, G_2, e, q, P, P_0, H_1, H_2, H_3)$.

Partial-Private-Key-Extract: According to the identity $\mathrm{ID}_i \in \{0,1\}^* (i \in A, B, C)$, KGC computes $Q_i = H_1(\mathrm{ID}_i)$ and $D_i = sQ_i$. It sends ID_i, D_i to the users through a secure channel.

Set-Secret-Value: It takes `params` and user's identity ID_i as inputs and selects a random secret $x_i \in Z_q^*$ as user's secret value.

Set-Private-Key: It takes `params`, ID_i and secret value x_i as inputs and outputs the tuple (D_i, x_i) as user's private key.

Set-Public-Key: It takes `params` and secret value x_i as inputs and outputs the user's public key $P_i = x_i P$.

Generate-Delegation: The original signer $\boldsymbol{A}$ generates a random number $r \in Z_q^*$ and computes $R = rP$, $d = (x_A + r)H_2(\mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B, w)$. It sends $\{w, d, R\}$ to the proxy signer $\boldsymbol{B}$.

Verify-Delegation: The proxy signer $\boldsymbol{B}$ verifies the delegation received from $\boldsymbol{A}$ by checking whether the equality $dP = (P_A + R)H_2(\mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B, w)$ holds or not. If it does not hold, $\boldsymbol{B}$ will require delegation again. Otherwise, it will believe the delegation is valid and compute a proxy signing key as $\mathrm{skp} = d + x_B H_2(\mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B, w)$.

Sign: The proxy signer $\boldsymbol{B}$ computes

$$U = e(D_B, P_C)e(\mathrm{skp}H_3(m, w, \mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B)Q_B, P + P_C),$$

which is the CLDVPS. It sends $\{m, w, R, U\}$ to the designated verifier $\boldsymbol{C}$.

Verify: Firstly, the designated verifier $\boldsymbol{C}$ computes $\mathrm{PKp} = (P_A + P_B + R)H_2(\mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B, w)$ which meets the requirement $\mathrm{PKp} = \mathrm{skp}P$. Afterwards, it computes

$$V = e(Q_B, x_C P_0 + H_3(m, w, \mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B)(\mathrm{PKp} + x_C \mathrm{PKp})).$$

If $U = V$, it will believe the signature is valid. Otherwise, it will consider the signature is invalid.

Transcript-Simulate: The designated verifier $\boldsymbol{C}$ can produce the signature V' intended for itself by computing

$$V' = e(Q_B, x_C P_0 + H_3(m, w, \mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B)(\mathrm{PKp} + x_C \mathrm{PKp})).$$

5 Analysis of our CLDVPS scheme

5.1 Correctness

In this subsection, we provide the correctness proof of our CLDVPS scheme. Firstly, we prove the equation $\mathrm{PKp} = \mathrm{skp}P$.

$$\begin{aligned}\mathrm{PKp} &= (P_A + P_B + R)H_2(\mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B, w)\\ &= (x_AP + x_BP + rP)H_2(\mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B, w)\\ &= (x_A + r)H_2(\mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B, w)P + x_BH_2(\mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B, w)P\\ &= dP + x_BH_2(\mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B, w)P = \mathrm{skp}P.\end{aligned}$$

Secondly, we prove the equation $U = V$.

$$\begin{aligned}U &= e(D_B, P_C)e(\mathrm{skp}H_3(m, w, \mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B)Q_B, P + P_C)\\ &= e(sQ_B, x_CP)e(Q_B, \mathrm{skp}H_3(m, w, \mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B)(P + P_C))\\ &= e(Q_B, x_CP_0)e(Q_B, H_3(m, w, \mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B)(\mathrm{PKp} + x_C\mathrm{PKp}))\\ &= e(Q_B, x_CP_0 + H_3(m, w, \mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B)(\mathrm{PKp} + x_C\mathrm{PKp})) = V.\end{aligned}$$

Therefore, we have proved the correctness of our CLDVPS scheme.

5.2 Security analysis

Theorem 1. Our CLDVPS scheme is a designated verifier signature scheme.

Proof. We note that the CLDVPS verifying algorithm needs x_C which is a private key of designated verifier $\boldsymbol{C}$. The designated verifier can compute a valid CLDVPS signature by computing $V' = e(Q_B, x_CP_0 + H_3(m, w, \mathrm{ID}_A, \mathrm{ID}_B, P_A, P_B)(\mathrm{PKp} + x_C\mathrm{PKp}))$ which is indistinguishable from the one computed by proxy signer $\boldsymbol{B}$. If the designated verifier does not compute CLDVPS and the signature is verified by the verifying algorithm, it will believe that the signature is valid. No third party other than the designated verifier can determine the signature is valid or invalid. Therefore, this signature scheme is a designated verifier signature scheme.

Theorem 2. It is supposed that $\mathcal{A}_{\mathcal{I}}$ is a Type I adaptive chosen message adversary against our CLDVPS with success probability ε in time t. It makes q_{H_1} queries to the H_1-Oracle, q_{H_2} queries to the H_2-Oracle, q_{H_3} queries to the H_3-Oracle, q_{CU} queries to the Create-User-Oracle, q_{PPK} queries to the Partial-Private-Key-Oracle, q_{SV} queries to the Secret-Value-Oracle, q_{PK} queries to the Public-Key-Oracle, q_{PKR} queries to the Public-Key-Replacement-Oracle, q_{PPri} queries to the Proxy-Private-Key-Generation-Oracle, q_{PPub} queries to the Proxy-Public-Key-Generation-Oracle, q_S queries to the Sign-Oracle, and q_V queries to the Verify-Oracle within the polynomial time t. There is an algorithm $\mathcal{B}$ which uses $\mathcal{A}_{\mathcal{I}}$ to solve an instance of CBDH problem with the advantage

$$\varepsilon' \geqslant \frac{1}{q_{H_1}}\left(\frac{2}{q_{H_1}}\right)^{q_{\mathrm{PPK}}+q_{\mathrm{SV}}+q_{\mathrm{PPri}}}\left(1-\frac{1}{q_{H_1}}\right)^{q_{\mathrm{PPK}}+q_{\mathrm{SV}}+q_{\mathrm{PPri}}+q_S+q_V}\varepsilon$$

within the time

$$t + (q_{H_1} + 2q_{\mathrm{CU}} + q_{\mathrm{PPub}} + 2q_S + 3q_V + 2)t^M_{G1} + (2q_S + q_V + 1)t_P + q_St^M_{G2} + t^{\mathrm{Exp}}_{G2},$$

where t^M_{G1} denotes the time of executing multiplication operation in G_1, t_P denotes the time of executing pairing operation, t^M_{G2} denotes the time of executing multiplication operation in G_2, t^{Exp}_{G2} denotes the time of executing exponential operation in G_2.

Proof. It is given a random instance $(P, P_1 = aP, P_2 = bP, P_3 = cP)$ of CBDH problem where the variables a, b, and c are randomly chosen and $a, b, c \in Z^*_q$. We will construct an algorithm $\mathcal{B}$ which uses $\mathcal{A}_{\mathcal{I}}$ to solve the CBDH problem, namely, to obtain the value of $e(P, P)^{abc}$.

Setup: There are three parts in the proof, original signer AOS, proxy signer BPS and designated verifier CDV. The algorithm $\mathcal{B}$ sets $P_0 = P_1 = aP$ and $P_{\mathrm{CDV}} = P_3 = cP$ where (P_1, P_3) is the instance of CBDH problem.

H_1-Oracle: In order to respond to the queries of $\mathcal{A}_\mathcal{I}$, $\mathcal{B}$ maintains a list H_1-list which consists of tuples $(\mathrm{ID}_j, Q_j, l_j)$. It randomly chooses $f \in \{1, 2, \ldots, q_{H_1}\}$ and the f-th query corresponds to the target identity ID*. The H_1-list is initially empty. When $\mathcal{A}_\mathcal{I}$ queries the H_1-Oracle with ID_j, $\mathcal{B}$ will check the H_1-list. If the ID_j already appears in the H_1-list, $\mathcal{B}$ returns the corresponding $Q_j = l_j P$. If there is not such an ID_j in the H_1-list, $\mathcal{B}$ checks the value of j and simulates the oracle as follows.

(1) If $j = f$, $\mathcal{B}$ randomly chooses $l_j \in Z_q^*$ and sets $Q_j = l_j bP$. It adds $(\mathrm{ID}_j, Q_j, l_j)$ to H_1-list and returns Q_j as the response.

(2) If $j \neq f$, $\mathcal{B}$ randomly chooses $l_j \in Z_q^*$ and sets $Q_j = l_j P$. It adds $(\mathrm{ID}_j, Q_j, l_j)$ to H_1-list and returns Q_j as the response.

Create-User-Oracle: $\mathcal{B}$ maintains a list User-list which consists of tuples $(\mathrm{ID}_j, x_j, D_{\mathrm{ID}_j}, P_{\mathrm{ID}_j})$. At any time, $\mathcal{A}_\mathcal{I}$ can require to create the user with ID_j. If there is not such an ID_j in the H_1-list, $\mathcal{B}$ will first issue a query to H_1-Oracle to ensure that there is a tuple containing the ID_j in the H_1-list. Afterwards, if there is such an ID_j in the User-list, $\mathcal{B}$ will do nothing. Otherwise, $\mathcal{B}$ searches the ID_j in H_1-list.

(1) If $j = f$, $\mathcal{B}$ randomly chooses $x_j \in Z_q^*$ and adds $(\mathrm{ID}_j, x_j, \perp, x_j P)$ to User-list.

(2) If $j \neq f$, $\mathcal{B}$ randomly chooses $x_j \in Z_q^*$ and adds $(\mathrm{ID}_j, x_j, l_j P_0, x_j P)$ to User-list.

Partial-Private-Key-Oracle: At any time $\mathcal{A}_\mathcal{I}$ can query a partial private key of ID_j. It is assumed that the partial private key of ID_j has been created. If it has not been created, $\mathcal{A}_\mathcal{I}$ will require to create the user with ID_j. When it receives a query for partial private key of ID_j, $\mathcal{B}$ will search H_1-list for a tuple $(\mathrm{ID}_j, Q_j, l_j)$.

(1) If $j = f$ or $i = f$, $\mathcal{B}$ aborts.

(2) Otherwise, $\mathcal{B}$ searches the tuple $(\mathrm{ID}_j, x_j, D_{\mathrm{ID}_j}, P_{\mathrm{ID}_j})$ in the User-list and returns D_j as response.

Secret-Value-Oracle: At any time $\mathcal{A}_\mathcal{I}$ can query a secret value of ID_j. It is assumed that the secret value of ID_j has been created. If it has not been created, $\mathcal{A}_\mathcal{I}$ will require to create the user with ID_j. When it receives a query for secret value of ID_j, $\mathcal{B}$ will search H_1-list for a tuple $(\mathrm{ID}_j, Q_j, l_j)$.

(1) If $j = f$ or $i = f$, $\mathcal{B}$ aborts.

(2) Otherwise, $\mathcal{B}$ searches the tuple $(\mathrm{ID}_j, x_j, D_{\mathrm{ID}_j}, P_{\mathrm{ID}_j})$ in the User-list and returns x_j as response.

Public-Key-Oracle: When it receives a query for public key of ID_j, $\mathcal{B}$ will search the tuple $(\mathrm{ID}_j, x_j, D_{\mathrm{ID}_j}, P_{\mathrm{ID}_j})$ in the User-list and return P_{ID_j} as the response.

Public-Key-Replacement-Oracle: $\mathcal{A}_\mathcal{I}$ can require to replace the public key of user ID_j, P_{ID_j}, with P'_{ID_j} selected by $\mathcal{A}_\mathcal{I}$. $\mathcal{B}$ will update the corresponding information in the User-list as $(\mathrm{ID}_j, \perp, D_{\mathrm{ID}_j}, P'_{\mathrm{ID}_j})$.

H_2-Oracle: $\mathcal{B}$ keeps a list H_2-list which consists of tuples $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, w_j, h2_j)$. When $\mathcal{A}_\mathcal{I}$ issues a query $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, w_j)$ to H_2-Oracle, $\mathcal{B}$ will check whether $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, w_j)$ exists in the H_2-list.

(1) If it does, $\mathcal{B}$ returns $h2_j$ to $\mathcal{A}_\mathcal{I}$.

(2) If it does not, $\mathcal{B}$ randomly chooses $h2_j \in Z_q^*$, adds $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, w_j, h2_j)$ to the H_2-list and returns $h2_j$ to $\mathcal{A}_\mathcal{I}$.

Proxy-Private-Key-Generation-Oracle: $\mathcal{B}$ keeps a list PPri-list which consists of tuples $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, r_j, \mathrm{skp}_j)$. When $\mathcal{A}_\mathcal{I}$ issues a query $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j)$ to Proxy-Private-Key-Generation-Oracle, $\mathcal{B}$ will search H_1-list.

(1) If $i = f$ or $j = f$, $\mathcal{B}$ aborts.

(2) Otherwise, $\mathcal{B}$ checks whether $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j)$ exists in the PPri-list.

(i) If it does, $\mathcal{B}$ returns skp_j to $\mathcal{A}_\mathcal{I}$.

(ii) If it does not, $\mathcal{B}$ randomly chooses $r_j \in Z_q^*$ and computes $\mathrm{skp}_j = (x_i + r_j + x_j)h2_j$. It adds $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, r_j, \mathrm{skp}_j)$ to the PPri-list and returns skp_j to $\mathcal{A}_\mathcal{I}$.

Proxy-Public-Key-Generation-Oracle: $\mathcal{B}$ keeps a list PPub-list which consists of tuples $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, r_j, \mathrm{PKp}_j)$. When $\mathcal{A}_\mathcal{I}$ issues a query $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j)$ to Proxy-Public-Key-Generation-Oracle, $\mathcal{B}$ will search H_1-list.

(1) If $j = f$, $\mathcal{B}$ randomly chooses $r_j \in Z_q^*$ and $\mathrm{PKp}_j \in G_1$. It adds $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, r_j, \mathrm{PKp}_j)$ to the PPub-list.

(2) If $j \neq f$, $\mathcal{B}$ checks whether $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j)$ exists in the PPub-list.

(i) If it does, $\mathcal{B}$ returns corresponding PKp_j to $\mathcal{A}_\mathcal{I}$.

(ii) If there is not a tuple, $\mathcal{B}$ first makes a Proxy-Private-Key-Generation-Oracle query on $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j)$ and ensures that there is a tuple $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j)$ in the PPri-list. $\mathcal{B}$ obtains corresponding r_j and skp_j in the PPri-list. Afterwards, it computes $\mathrm{PKp}_j = \mathrm{skp}_j P$ and adds $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, r_j, \mathrm{PKp}_j)$ to the PPub-list. It returns PKp_j to $\mathcal{A}_\mathcal{I}$.

H_3-Oracle: $\mathcal{B}$ keeps a list H_3-list which consists of tuples $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, h3_j)$. $\mathcal{A}_\mathcal{I}$ issues a query on tuple $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j)$.

(1) If there is such a tuple $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j)$ in the H_3-list, $\mathcal{B}$ returns $h3_j$.

(2) Otherwise, $\mathcal{B}$ randomly chooses $h3_j \in Z_q^*$ and adds $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, h3_j)$ to the H_3-list. It returns $h3_j$ as response.

Sign-Oracle: $\mathcal{B}$ keeps a list Sign-list which consists of tuples $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, h3_j, U_j)$. $\mathcal{A_I}$ issues a query on tuple $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, h3_j)$.

(1) If $j = f$, it will abort.

(2) Otherwise, $j \neq f$.

(i) If there is a $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, h3_j, U_j)$ in the Sign-list, it returns U_j in the Sign-list.

(ii) If there is not a $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, h3_j, U_j)$ in the Sign-list, it sets $U_j = e(l_jP_0, P_3)e(\mathrm{skp}_j h3_j l_j P, (P + P_3))$ and adds $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, h3_j, U_j)$ to the Sign-list. It returns U_j as response.

Verify-Oracle: When $\mathcal{A_I}$ issues query on tuple $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, U_j)$, $\mathcal{B}$ will determine whether the U_j is a valid signature. $\mathcal{B}$ searches the Sign-list.

(1) If $j = f$, $\mathcal{B}$ will abort.

(2) If $j \neq f$ and there is such a tuple $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, U_j)$ in the Sign-list, $\mathcal{B}$ considers it is a valid signature and outputs `True`.

(3) If $j \neq f$ and there is not such a tuple $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, U_j)$, $\mathcal{B}$ determines whether $U_j = e(l_jP, x_kP_0 + h3_j((1 + x_k)\mathrm{skp}_j P))$.

(i) If it holds, $\mathcal{B}$ considers it is a valid signature and outputs `True`.

(ii) If it does not hold, $\mathcal{B}$ considers it is not a valid signature and outputs `False`.

Output: After all queries, $\mathcal{A_I}$ outputs a forgery $(m^*, w^*, \mathrm{ID}^*, U^*)$. $\mathcal{B}$ computes

$$(U^*/e(l_f bP, h3_f \mathrm{skp}_f(P + P_{\mathrm{CDV}})))^{(1/l_f)} = e(l_f bP_0, P_3)^{(1/l_f)} = e(abP, cP) = e(P, P)^{abc}.$$

It completes the description of $\mathcal{B}$. It is assumed that $\mathcal{B}$ solves the given instance of CBDH problem with advantage at least ε'. We analyze the events needed for $\mathcal{B}$ to succeed.

E_1: $\mathcal{B}$ does not abort as a result of any queries of Partial-Private-Key-Extract-Oracle, Secret-Value-Oracle, Proxy-Private-Key-Generation-Oracle, Sign-Oracle, and Verify-Oracle.

E_2: $\mathcal{A_I}$ generates a valid CLDVPS.

E_3: The forged signature fulfills $j = f$, namely, $\mathrm{ID}^* = \mathrm{ID}_f$.

$\mathcal{B}$ succeeds if all these events happen. It uses E_{11} to denote the event $\mathcal{B}$ does not abort as a result of any queries of Partial-Private-Key-Extract-Oracle, which happens with probability $\Pr[E_{11}] = (\frac{1}{q_{H_1}}(1 - \frac{1}{q_{H_1}}) + \frac{1}{q_{H_1}}(1 - \frac{1}{q_{H_1}}))^{q_{\mathrm{PPK}}} = (\frac{2}{q_{H_1}}(1 - \frac{1}{q_{H_1}}))^{q_{\mathrm{PPK}}}$. It needs to note that i and j are not simultaneously equal to f. Similarly, it uses E_{12}, E_{13}, E_{14}, and E_{15} to denote the events $\mathcal{B}$ does not abort as a result of any queries of Secret-Value-Oracle, Proxy-Private-Key-Generation-Oracle, Sign-Oracle, and Verify-Oracle, respectively. These events happen with probability $\Pr[E_{12}] = (\frac{2}{q_{H_1}}(1 - \frac{1}{q_{H_1}}))^{q_{\mathrm{SV}}}$, $\Pr[E_{13}] = (\frac{2}{q_{H_1}}(1 - \frac{1}{q_{H_1}}))^{q_{\mathrm{PPri}}}$, $\Pr[E_{14}] = (1 - \frac{1}{q_{H_1}})^{q_S}$, and $\Pr[E_{15}] = (1 - \frac{1}{q_{H_1}})^{q_V}$. Therefore, E_1 happens with the probability $\Pr[E_1] = (\frac{2}{q_{H_1}}(1 - \frac{1}{q_{H_1}}))^{q_{\mathrm{PPK}}+q_{\mathrm{SV}}+q_{\mathrm{PPri}}}(1 - \frac{1}{q_{H_1}})^{q_S+q_V}$.

The probability $\Pr[E_1E_2E_3]$ is decomposed as:

$$\begin{aligned}
\Pr[E_1E_2E_3] &= \Pr[E_1] \cdot \Pr[E_2|E_1] \cdot \Pr[E_3|E_1E_2] \\
&= \left(\frac{2}{q_{H_1}}\left(1 - \frac{1}{q_{H_1}}\right)\right)^{q_{\mathrm{PPK}}+q_{\mathrm{SV}}+q_{\mathrm{PPri}}} \left(1 - \frac{1}{q_{H_1}}\right)^{q_S+q_V} \frac{1}{q_{H_1}}\varepsilon \\
&= \frac{1}{q_{H_1}}\left(\frac{2}{q_{H_1}}\right)^{q_{\mathrm{PPK}}+q_{\mathrm{SV}}+q_{\mathrm{PPri}}} \left(1 - \frac{1}{q_{H_1}}\right)^{q_{\mathrm{PPK}}+q_{\mathrm{SV}}+q_{\mathrm{PPri}}+q_S+q_V} \varepsilon.
\end{aligned}$$

Moreover, the total running time of $\mathcal{B}$ is at most

$$t + (q_{H_1} + 2q_{\mathrm{CU}} + q_{\mathrm{PPub}} + 2q_S + 3q_V + 2)t_{G1}^M + (2q_S + q_V + 1)t_P + q_S t_{G2}^M + t_{G2}^{\mathrm{Exp}}.$$

Therefore, $\mathcal{B}$ solves the CBDH problem with the advantage

$$\varepsilon' \geqslant \frac{1}{q_{H_1}}\left(\frac{2}{q_{H_1}}\right)^{q_{\mathrm{PPK}}+q_{\mathrm{SV}}+q_{\mathrm{PPri}}} \left(1 - \frac{1}{q_{H_1}}\right)^{q_{\mathrm{PPK}}+q_{\mathrm{SV}}+q_{\mathrm{PPri}}+q_S+q_V} \varepsilon$$

within the time

$$t' \leqslant t + (q_{H_1} + 2q_{\mathrm{CU}} + q_{\mathrm{PPub}} + 2q_S + 3q_V + 2)t_{G1}^M + (2q_S + q_V + 1)t_P + q_S t_{G2}^M + t_{G2}^{\mathrm{Exp}}.$$

Theorem 3. It is supposed that $\mathcal{A}_{\mathcal{II}}$ is a Type II adaptive chosen message adversary against our CLDVPS with success probability ε in time t. It makes q_{H_1} queries to the H_1-Oracle, q_{H_2} queries to the H_2-Oracle, q_{H_3} queries to the H_3-Oracle, $q_{\rm CU}$ queries to the Create-User-Oracle, $q_{\rm PPK}$ queries to the Partial-Private-Key-Oracle, $q_{\rm SV}$ queries to the Secret-Value-Oracle, $q_{\rm PK}$ queries to the Public-Key-Oracle, $q_{\rm PKR}$ queries to the Public-Key-Replacement-Oracle, $q_{\rm PPri}$ queries to the Proxy-Private-Key-Generation-Oracle, $q_{\rm PPub}$ queries to the Proxy-Public-Key-Generation-Oracle, q_S queries to the Sign-Oracle, and q_V queries to the Verify-Oracle within the polynomial time t. There is an algorithm $\mathcal{B}$ which uses $\mathcal{A}_{\mathcal{II}}$ to solve an instance of CBDH problem with the advantage

$$\varepsilon' \geqslant \frac{1}{q_{H_1}}\left(\frac{2}{q_{H_1}}\right)^{q_{\rm SV}+q_{\rm PKR}+q_{\rm PPri}}\left(1-\frac{1}{q_{H_1}}\right)^{q_{\rm SV}+q_{\rm PKR}+q_{\rm PPri}+q_S+q_V}\varepsilon$$

within the time

$$t' \leqslant t+(q_{H_1}+2q_{\rm CU}+q_{\rm PPub}+2q_S+3q_V+2)t_{G1}^M+(2q_S+q_V+2)t_p+(q_S+1)t_{G2}^M+t_{G2}^{\rm Exp},$$

where t_{G1}^M denotes the time of executing multiplication operation in G_1, t_P denotes the time of executing pairing operation, t_{G2}^M denotes the time of executing multiplication operation in G_2, $t_{G2}^{\rm Exp}$ denotes the time of executing exponential operation in G_2.

Proof. It is given a random instance $(P, P_1 = aP, P_2 = bP, P_3 = cP)$ of CBDH problem where the variables a, b, and c are randomly chosen and $a, b, c \in Z_q^*$. We will construct an algorithm $\mathcal{B}$ which uses $\mathcal{A}_{\mathcal{II}}$ to solve the CBDH problem, namely, to obtain the value of $e(P, P)^{abc}$.

Setup: There are three parts in the proof, original signer AOS, proxy signer BPS and designated verifier CDV. The algorithm $\mathcal{B}$ sets $P_{\rm CDV} = P_3 = cP$ where P_3 is the instance of CBDH problem.

H_1**-Oracle:** In order to respond to the queries of $\mathcal{A}_{\mathcal{II}}$, $\mathcal{B}$ maintains a list H_1-list which consists of tuples $({\rm ID}_j, Q_j, l_j)$. It randomly chooses $f \in \{1, 2, \ldots, q_{H_1}\}$ and the f-th query corresponds to the target identity ${\rm ID}^*$. This list is initially empty. When $\mathcal{A}_{\mathcal{II}}$ queries the H_1-Oracle with ${\rm ID}_j$, $\mathcal{B}$ will check the H_1-list. If the ${\rm ID}_j$ already appears in the H_1-list, $\mathcal{B}$ will return the corresponding Q_j. If there is not such an ${\rm ID}_j$ in the H_1-list, $\mathcal{B}$ will simulate the oracle as follows.

(1) If $j = f$, $\mathcal{B}$ randomly chooses $l_j \in Z_q^*$ and sets $Q_j = l_j bP$. It adds $({\rm ID}_j, Q_j, l_j)$ to H_1-list and returns Q_j as response.

(2) If $j \neq f$, $\mathcal{B}$ randomly chooses $l_j \in Z_q^*$ and sets $Q_j = l_jP$. It adds $({\rm ID}_j, Q_j, l_j)$ to H_1-list and returns Q_j as response.

Create-User-Oracle: $\mathcal{B}$ maintains a list User-list which consists of tuples $({\rm ID}_j, x_j, D_{{\rm ID}_j}, P_{{\rm ID}_j})$. At any time, $\mathcal{A}_{\mathcal{II}}$ can require to create the user with ${\rm ID}_j$. If there is not such an ${\rm ID}_j$ in the H_1-list, $\mathcal{B}$ first issues a query to H_1-Oracle to ensure that there is a tuple in the H_1-list containing the ${\rm ID}_j$.

(1) If there is such an ${\rm ID}_j$ in the User-list, $\mathcal{B}$ does nothing.

(2) Otherwise, $\mathcal{B}$ randomly chooses $x_j \in Z_q^*$ and adds $({\rm ID}_j, x_j, sQ_j, x_jP)$ to User-list.

Secret-Value-Oracle: At any time $\mathcal{A}_{\mathcal{II}}$ can query a secret value of ${\rm ID}_j$. It is assumed that the secret value of ${\rm ID}_j$ has been created. If it has not been created, $\mathcal{A}_{\mathcal{II}}$ will require to create the user with ${\rm ID}_j$. When it receives a query for secret value of ${\rm ID}_j$, $\mathcal{B}$ will search H_1-list for a tuple $({\rm ID}_j, Q_j, l_j)$.

(1) If $j = f$ or $i = f$, $\mathcal{B}$ aborts.

(2) Otherwise, $\mathcal{B}$ searches the tuple $({\rm ID}_j, x_j, D_{{\rm ID}_j}, P_{{\rm ID}_j})$ in the User-list and returns x_j as the response.

Public-Key-Oracle: When it receives a query for public key of ${\rm ID}_j$, $\mathcal{B}$ searches the tuple $({\rm ID}_j, x_j, D_{{\rm ID}_j}, P_{{\rm ID}_j})$ in the User-list and returns $P_{{\rm ID}_j}$ as the response.

Public-Key-Replacement-Oracle: $\mathcal{A}_{\mathcal{II}}$ can require to replace the public key of user ${\rm ID}_j$, $P_{{\rm ID}_j}$, with $P'_{{\rm ID}_j}$ selected by $\mathcal{A}_{\mathcal{II}}$.

(1) If $j = f$ or $i = f$, $\mathcal{B}$ aborts.

(2) Otherwise, $\mathcal{B}$ will update the corresponding information in the User-list as $({\rm ID}_j, \perp, D_{{\rm ID}_j}, P'_{{\rm ID}_j})$.

H_2**-Oracle:** $\mathcal{B}$ keeps a list H_2-list which consists of tuples $({\rm ID}_i, {\rm ID}_j, P_i, P_j, w_j, h2_j)$. When $\mathcal{A}_{\mathcal{II}}$ issues a query $({\rm ID}_i, {\rm ID}_j, P_i, P_j, w_j)$ to H_2-Oracle, $\mathcal{B}$ checks whether $({\rm ID}_i, {\rm ID}_j, P_i, P_j, w_j)$ exists in the H_2-list.

(1) If it does, $\mathcal{B}$ returns $h2_j$ to $\mathcal{A}_{\mathcal{II}}$.

(2) Otherwise, $\mathcal{B}$ randomly chooses $h2_j \in Z_q^*$, adds $({\rm ID}_i, {\rm ID}_j, P_i, P_j, w_j, h2_j)$ to the H_2-list, and returns $h2_j$ to $\mathcal{A}_{\mathcal{II}}$.

Proxy-Private-Key-Generation-Oracle: $\mathcal{B}$ keeps a list PPri-list which consists of tuples $({\rm ID}_i, {\rm ID}_j, P_i, P_j, r_j, {\rm skp}_j)$. When $\mathcal{A}_{\mathcal{II}}$ issues a query $({\rm ID}_i, {\rm ID}_j, P_i, P_j)$ to Proxy-Private-Key-Generation-Oracle, $\mathcal{B}$ searches H_1-list.

(1) If $i = f$ or $j = f$, $\mathcal{B}$ aborts.

(2) Otherwise, $\mathcal{B}$ checks whether $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j)$ exists in the PPri-list.

(i) If it does, $\mathcal{B}$ return skp_j to $\mathcal{A_{II}}$.

(ii) Otherwise, $\mathcal{B}$ randomly chooses $r_j \in Z_q^*$ and computes $\mathrm{skp}_j = (x_i + r_j + x_j)h2_j$. It adds $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, r_j, \mathrm{skp}_j)$ to the PPri-list and returns skp_j to $\mathcal{A_{II}}$.

Proxy-Public-Key-Generation-Oracle: $\mathcal{B}$ keeps a list PPub-list which consists of tuples $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, r_j, \mathrm{PKp}_j)$. When $\mathcal{A_{II}}$ issues a query $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j)$ to Proxy-Public-Key-Generation-Oracle, $\mathcal{B}$ searches H_1-list.

(1) If $j = f$, $\mathcal{B}$ sets $\mathrm{PKp}_j = aP$ and randomly chooses $r_j \in Z_q^*$. It adds $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, r_j, \mathrm{PKp}_j)$ to the PPub-list.

(2) If $j \neq f$, $\mathcal{B}$ checks whether $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j)$ exists in the PPub-list.

(i) If it does, $\mathcal{B}$ returns corresponding PKp_j to $\mathcal{A_{II}}$.

(ii) If there is not the tuple, $\mathcal{B}$ first makes a Proxy-Private-Key-Generation-Oracle query on $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j)$ and obtains corresponding r_j and skp_j. Afterwards, it computes $\mathrm{PKp}_j = \mathrm{skp}_j P$ and adds $(\mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, r_j, \mathrm{PKp}_j)$ to the PPub-list. It returns PKp_j to $\mathcal{A_{II}}$.

H_3**-Oracle:** $\mathcal{B}$ keeps a list H_3-list which consists of tuples $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, h3_j)$. $\mathcal{A_{II}}$ issues a query on tuple $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j)$.

(1) If there is such a tuple $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j)$ in the H_3-list, $\mathcal{B}$ returns $h3_j$.

(2) Otherwise, $\mathcal{B}$ randomly chooses $h3_j \in Z_q^*$ and adds $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, h3_j)$ to the H_3-list. It returns $h3_j$ as response.

Sign-Oracle: $\mathcal{B}$ keeps a list Sign-list which consists of tuples $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, h3_j, U_j)$. $\mathcal{A_{II}}$ issues a query on tuple $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, h3_j)$.

(1) If $j = f$, it aborts.

(2) Otherwise, $j \neq f$.

(i) If there is a $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, h3_j, U_j)$ in the Sign-list, it returns U_j as response.

(ii) If there is not a $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, h3_j, U_j)$ in the Sign-list, it sets $U_j = e(l_j sP, P_3)e(\mathrm{skp}_j h3_j l_j P, P + P_3)$ and adds $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, h3_j, U_j)$ to the Sign-list. It returns U_j as response.

Verify-Oracle: When $\mathcal{A_{II}}$ issues queries on tuple $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, U_j)$, $\mathcal{B}$ will determine whether the U_j is a valid signature. $\mathcal{B}$ searches the Sign-list.

(1) If $j = f$, $\mathcal{B}$ aborts.

(2) If $j \neq f$ and there is such a tuple $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, U_j)$ in the Sign-list, $\mathcal{B}$ considers it is a valid signature and outputs `True`.

(3) If $j \neq f$ and there is not such a tuple $(m_j, w_j, \mathrm{ID}_i, \mathrm{ID}_j, P_i, P_j, U_j)$ in the Sign-list, $\mathcal{B}$ determines whether $U_j = e(l_j P, x_k sP + h3_j((1 + x_k)\mathrm{skp}_j P))$.

(i) If it holds, $\mathcal{B}$ considers it is a valid signature and outputs `True`.

(ii) If it does not hold, $\mathcal{B}$ considers it is not a valid signature and outputs `False`.

Output: After all queries, $\mathcal{A_{II}}$ will output a forgery $(m^*, w^*, \mathrm{ID}^*, U^*)$. $\mathcal{B}$ computes

$$\begin{aligned}(U^*/e(l_f sbP, P_3)e(\mathrm{skp}_f h3_f Q_f, P))^{(1/h3_f l_f)} &= e(\mathrm{skp}_f h3_f Q_f, P_{\mathrm{CDV}})^{(1/h3_f l_f)} \\ &= e(h3_f l_f b \mathrm{PKp}_f, cP)^{(1/h3_f l_f)} \\ &= e(abP, cP) = e(P, P)^{abc}.\end{aligned}$$

It completes the description of $\mathcal{B}$. It is assumed that $\mathcal{B}$ solves the given instance of CBDH problem with advantage at least ε'. First, we analyze the events needed for $\mathcal{B}$ to succeed.

E_1: $\mathcal{B}$ does not abort as a result of any queries of Secret-Value-Oracle, Public-Key-Replacement-Oracle, Proxy-Private-Key-Generation-Oracle, Sign-Oracle, and Verify-Oracle.

E_2: $\mathcal{A_{II}}$ generates a valid CLDVPS.

E_3: The forged signature fulfills $j = f$, namely, $\mathrm{ID}^* = \mathrm{ID}_f$.

$\mathcal{B}$ succeeds if all these events happen. It uses E_{11} to denote the event $\mathcal{B}$ does not abort as a result of any queries of Secret-Value-Oracle, which happens with probability $\Pr[E_{11}] = (\frac{1}{q_{H_1}}(1 - \frac{1}{q_{H_1}}) + \frac{1}{q_{H_1}}(1 - \frac{1}{q_{H_1}}))^{q_{\mathrm{SV}}} = (\frac{2}{q_{H_1}}(1 - \frac{1}{q_{H_1}}))^{q_{\mathrm{SV}}}$. It needs to note that i and j are not simultaneously equal to f. Similarly, it uses E_{12}, E_{13}, E_{14}, and E_{15} to denote the events $\mathcal{B}$ does not abort as a result of any queries of Public-Key-Replacement-Oracle, Proxy-Private-Key-Generation-Oracle, Sign-Oracle, and Verify-Oracle, respectively. These events happen with probability $\Pr[E_{12}] = (\frac{2}{q_{H_1}}(1 - \frac{1}{q_{H_1}}))^{q_{\mathrm{PKR}}}$, $\Pr[E_{13}] = (\frac{2}{q_{H_1}}(1 - \frac{1}{q_{H_1}}))^{q_{\mathrm{PPri}}}$,

$\Pr[E_{14}] = (1-\frac{1}{q_{H_1}})^{q_S}$, and $\Pr[E_{15}] = (1-\frac{1}{q_{H_1}})^{q_V}$. Therefore, E_1 happens with probability $\Pr[E_1] = (\frac{2}{q_{H_1}}(1-\frac{1}{q_{H_1}}))^{q_{\rm SV}+q_{\rm PKR}+q_{\rm PPri}}(1-\frac{1}{q_{H_1}})^{q_S+q_V}$.

The probability $\Pr[E_1E_2E_3]$ is decomposed as

$$\begin{aligned}\Pr[E_1E_2E_3] &= \Pr[E_1]\cdot\Pr[E_2|E_1]\cdot\Pr[E_3|E_1E_2]\\ &= \left(\frac{2}{q_{H_1}}\left(1-\frac{1}{q_{H_1}}\right)\right)^{q_{\rm SV}+q_{\rm PKR}+q_{\rm PPri}}\left(1-\frac{1}{q_{H_1}}\right)^{q_S+q_V}\frac{1}{q_{H_1}}\varepsilon\\ &= \frac{1}{q_{H_1}}\left(\frac{2}{q_{H_1}}\right)^{q_{\rm SV}+q_{\rm PKR}+q_{\rm PPri}}\left(1-\frac{1}{q_{H_1}}\right)^{q_{\rm SV}+q_{\rm PKR}+q_{\rm PPri}+q_S+q_V}\varepsilon.\end{aligned}$$

Moreover, the total running time of $\mathcal{B}$ is at most

$$t+(q_{H_1}+2q_{\rm CU}+q_{\rm PPub}+2q_S+3q_V+2)t^M_{G1}+(2q_S+q_V+2)t_p+(q_S+1)t^M_{G2}+t^{\rm Exp}_{G2}.$$

Therefore, $\mathcal{B}$ solves the CBDH problem with the advantage

$$\varepsilon' \geqslant \frac{1}{q_{H_1}}\left(\frac{2}{q_{H_1}}\right)^{q_{\rm SV}+q_{\rm PKR}+q_{\rm PPri}}\left(1-\frac{1}{q_{H_1}}\right)^{q_{\rm SV}+q_{\rm PKR}+q_{\rm PPri}+q_S+q_V}\varepsilon$$

within the time

$$t' \leqslant t+(q_{H_1}+2q_{\rm CU}+q_{\rm PPub}+2q_S+3q_V+2)t^M_{G1}+(2q_S+q_V+2)t_p+(q_S+1)t^M_{G2}+t^{\rm Exp}_{G2}.$$

5.3 Efficiency analysis

We compare the efficiency of our CLDVPS scheme with other related signature schemes on the computation costs and communication costs in this subsection. For the computation costs, we implement our CLDVPS scheme and other related signature schemes to obtain the computing time in different phases. For the communication costs, we theoretically analyze the signature length of different signature schemes.

We use a PC with Intel i5-4590 CPU and 4GB RAM and the Java pairing-based cryptography (jPBC) library [30] to implement our CLDVPS scheme and other related signature schemes. It selects the type A elliptic curve in the jPBC and implements various operations over such elliptic curve. We divide these signature schemes into four phases, delegation generation phase, delegation verification phase, signing phase, and verifying phase. We have obtained the computing time of our CLDVPS scheme, CLPS-CYC [15], CLPS-ZL [16], CLPS-ZhL [17], CLPS-SSH [18], and CLDVPS-DHZ [25] in the four phases. We have also obtained the computing time of CLDVS-HXY [23], CLDVS-CH [24], CLDVS-DHZ [25], CLDVS-XZB [26], and CLDVS-LHY [28] in the signing phase and verifying phase because there are not delegation generation phase and delegation verification phase in these CLDVS schemes. We divide these signature schemes into three groups and compare our CLDVPS with other signature schemes in different groups.

• CLPS schemes. They include CLPS-CYC, CLPS-ZL, CLPS-ZhL, and CLPS-SSH.

• CLDVS schemes. They include CLDVS-HXY, CLDVS-CH, CLDVS-DHZ, CLDVS-XZB, and CLDVS-LHY.

• CLDVPS scheme. It includes CLDVPS-DHZ.

(1) The computing time compared with other CLPS schemes.

• Delegation generation time. In the delegation generation phase, an original signer generates a delegation. The computing time in this phase is illustrated as Figure 2. From Figure 2 we find that the delegation generation time of our CLDVPS scheme is longer than CLPS-ZhL and shorter than other CLPS schemes.

• Delegation verification time. In the delegation verification phase, a proxy signer verifies the delegation and computes a private key for proxy signature if it is necessary. The computing time in this phase is illustrated as Figure 3. From Figure 3 we find that the delegation verification time of our CLDVPS scheme is shortest compared with other CLPS schemes.

• Signing time. In the signing phase, a proxy signer computes a signature. The computing time in this phase is illustrated as Figure 4. From Figure 4 we find that the signing time of our CLDVPS scheme is longer than CLPS-CYC and shorter than other CLPS schemes.

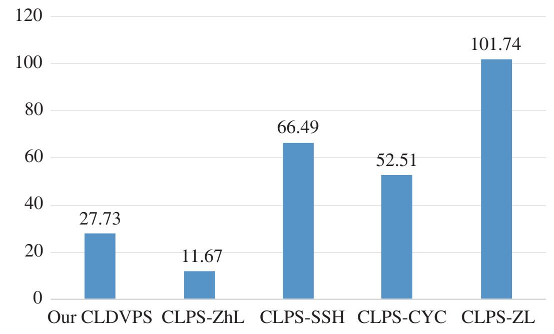

Figure 2 (Color online) Delegation generation time (ms) compared with other CLPS schemes.

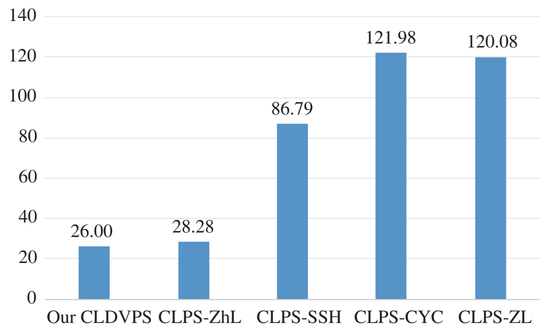

Figure 3 (Color online) Delegation verification time (ms) compared with other CLPS schemes.

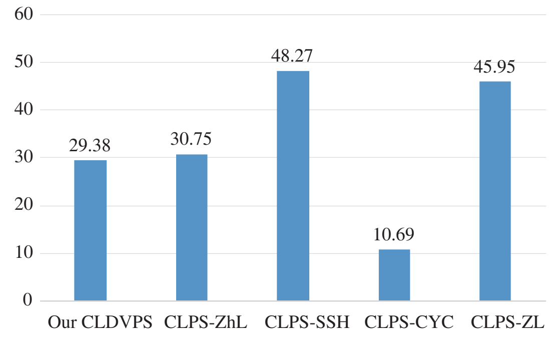

Figure 4 (Color online) Signing time (ms) compared with other CLPS schemes.

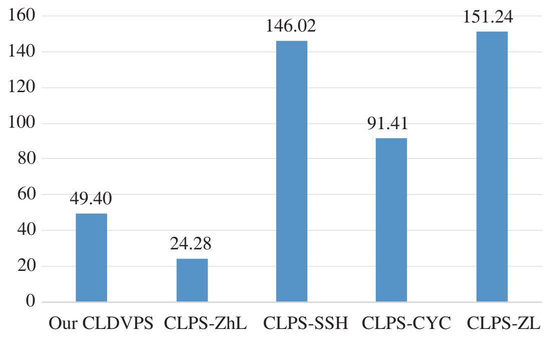

Figure 5 (Color online) Verifying time (ms) compared with other CLPS schemes.

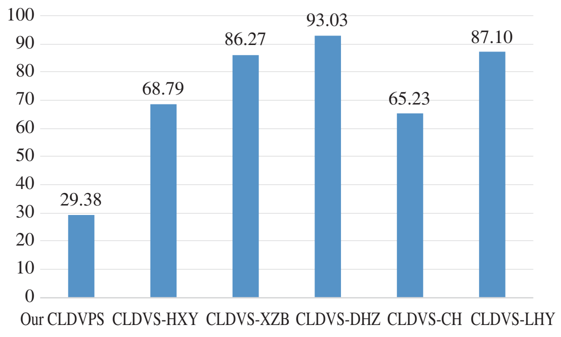

Figure 6 (Color online) Signing time (ms) compared with other CLDVS schemes.

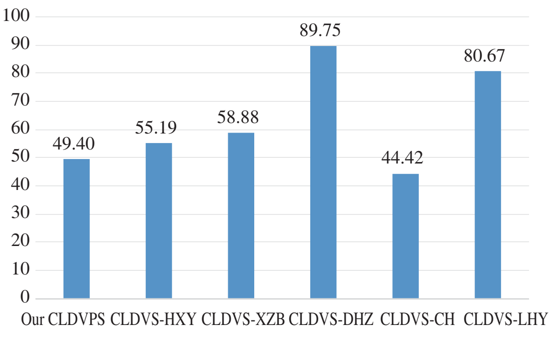

Figure 7 (Color online) Verifying time (ms) compared with other CLDVS schemes.

• Verifying time. In the verifying phase, a verifier verifies the signature received from the proxy signer. The computing time in this phase is illustrated as Figure 5. From Figure 5 we find that the verifying time of our CLDVPS scheme is longer than CLPS-ZhL and shorter than other CLPS schemes.

(2) The computing time compared with other CLDVS schemes.

• Signing time. The computing time in this phase is illustrated as Figure 6. From Figure 6 we find that the signing time of our CLDVPS scheme is shortest compared with other CLDVS schemes.

• Verifying time. The computing time in this phase is illustrated as Figure 7. From Figure 7 we find that the verifying time of our CLDVPS scheme is longer than CLDVS-CH and shorter than other CLDVS schemes.

(3) The computing time compared with other CLDVPS scheme. We compare our CLDVPS with CLDVPS-DHZ, which is illustrated as Figure 8. From Figure 8, we find that our CLDVPS has shorter computing time than CLDVPS-DHZ.

By comparing the computing time of different phases, our CLDVPS scheme has shorter computing

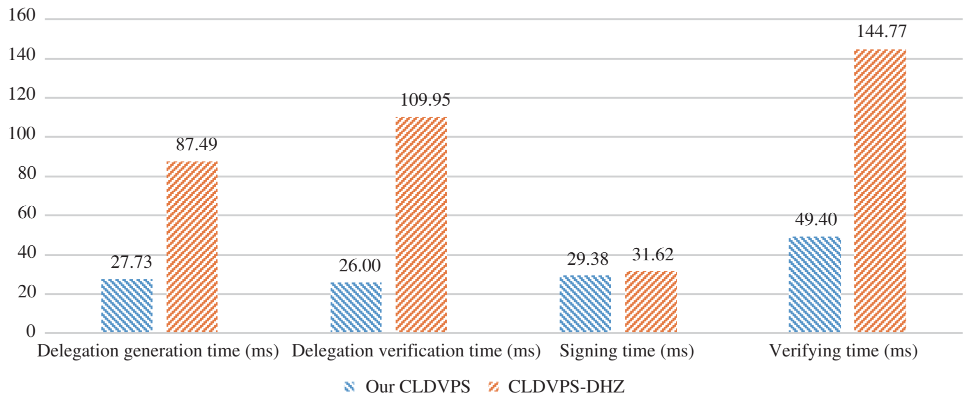

Figure 8 (Color online) Computation cost time (ms) compared with CLDVPS-DHZ.

Table 1 Length of different CLPS schemes

	Our CLDVPS	CLPS-CYC	CLPS-ZL	CLPS-ZhL	CLPS-SSH
Length	$\|G_1\| + \|G_2\|$	$\|G_1\|$	$3\|G_1\|$	$\|G_1\| + 2\|Z_q^*\|$	$3\|G_1\|$

Table 2 Length of different CLDVS schemes

	Our CLDVPS	CLDVS-HXY	CLDVS-CH	CLDVS-DHZ	CLDVS-XZB	CLDVS-LHY	CLDVPS-DHZ
Length	$\|G_1\| + \|G_2\|$	$\|Z_q^*\|$	$\|Z_q^*\|$	$\|G_1\| + 2\|Z_q^*\|$	$2\|G_1\|$	$\|G_1\| + 2\|Z_q^*\|$	$3\|G_1\| + 2\|Z_q^*\|$

time than most of other signature schemes. Therefore, our CLDVPS scheme is more efficient on the computation costs than most of other related signature schemes.

Next, we theoretically analyze the communication costs, namely, the signature length, of different signature schemes in the Tables 1 and 2. Due to the limited communication bandwidth of UAVs, a short digital signature facilitates the transmission and reception of signature. The result indicates that our CLDVPS scheme is also efficient on the the communication costs.

6 Conclusion

A UAV network has some characteristics, such as high mobility, dynamic topology, intermittent link, power constraints, and changing link quality. The digital signature scheme in a UAV network should solve the problems of digital certificate management and key escrow, guarantee the real-time performance of UAV executing commands, and ensure only the designated verifier can verify the signature. Therefore, we proposed a certificateless designated verifier proxy signature scheme for UAV networks to meet these requirements. We proved our CLDVPS scheme is existentially unforgeable under an adaptive chosen message attack in the random oracle model and compared the efficiency of our CLDVPS scheme with other signature schemes through computation costs and communication costs. We implemented these signature schemes in jPBC and obtained the computing time in different phases, which is computation cost. We theoretically analyzed the length of signature which is communication cost. The analysis results indicate that our CLDVPS scheme is more efficient than most of other related signature schemes on computation costs and also efficient on communication costs.

Acknowledgements This work was supported in part by Key Program of National Natural Science Foundation of China (NSFC) (Grant No. U1405255), in part by Shaanxi Science and Technology Coordination and Innovation Project (Grant No. 2016TZC-G-6-3), in part by Fundamental Research Funds for the Central Universities (Grant No. SA-ZD161504), and in part by National Natural Science Foundation of China (Grant No. 61802195).

References

1 Gupta L, Jain R, Vaszkun G. Survey of important issues in UAV communication networks. IEEE Commun Surv Tut, 2016, 18: 1123–1152
2 Shamir A. Identity-based cryptosystems and signature schemes. In: Proceedings of Workshop on the Theory and Application of Cryptographic Techniques, Santa Barbara, 1984. 47–53
3 Al-Riyami S S, Paterson K G. Certificateless public key cryptography. Lect Notes Comput Sci, 2003, 2894: 452–473
4 Huang X Y, Susilo W, Mu Y, et al. On the security of certificateless signature schemes from Asiacrypt 2003. In: Proceedings of International Conference on Cryptology and Network Security, Xiamen, 2005. 13–25

5 He D B, Chen J H, Zhang R. An efficient and provably-secure certificateless signature scheme without bilinear pairings. Int J Commun Syst, 2012, 25: 1432–1442

6 Wang L L, Chen K F, Long Y, et al. An efficient pairing-free certificateless signature scheme for resource-limited systems. Sci China Inf Sci, 2017, 60: 119102

7 Karati A, Islam S H, Biswas G P. A pairing-free and provably secure certificateless signature scheme. Inf Sci, 2018, 450: 378–391

8 Yeh K H, Su C H, Choo K K R, et al. A novel certificateless signature scheme for smart objects in the Internet-of-Things. Sensors, 2017, 17: 1001

9 Jia X Y, He D B, Liu Q, et al. An efficient provably-secure certificateless signature scheme for Internet-of-Things deployment. Ad Hoc Netw, 2018, 71: 78–87

10 Deng L Z, Yang Y X, Gao R H, et al. Certificateless short signature scheme from pairing in the standard model. Int J Commun Syst, 2018, 31: 3796

11 Du H Z, Wen Q Y, Zhang S S. A provably-secure outsourced revocable certificateless signature scheme without bilinear pairings. IEEE Access, 2018, 6: 73846–73855

12 Karati A, Islam S H, Karuppiah M. Provably secure and lightweight certificateless signature scheme for IIoT environments. IEEE Trans Ind Inf, 2018, 14: 3701–3711

13 Mambo M, Usuda K, Okamoto E. Proxy signatures for delegating signing operation. In: Proceedings of the 3rd ACM Conference on Computer and Communications Security, New Delhi, 1996. 48–57

14 Jakobsson M, Sako K, Impagliazzo R. Designated verifier proofs and their applications. In: Proceedings of International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, 1996. 143–154

15 Chen Y C, Liu C L, Horng G, et al. A provably secure certificateless proxy signature scheme. Int J Innov Comput Inf Control, 2011, 7: 5557–5569

16 Zhang L, Zhang F T, Wu Q H. Delegation of signing rights using certificateless proxy signatures. Inf Sci, 2012, 184: 298–309

17 Zhang L, Wu Q H, Qin B, et al. A generic construction of proxy signatures from certificateless signatures. In: Proceedings of the 27th International Conference on Advanced Information Networking and Applications, Barcelona, 2013. 259–266

18 Seo S H, Choi K Y, Hwang J Y, et al. Efficient certificateless proxy signature scheme with provable security. Inf Sci, 2012, 188: 322–337

19 He D B, Chen Y T, Chen J H. An efficient certificateless proxy signature scheme without pairing. Math Comput Model, 2013, 57: 2510–2518

20 Padhye S, Tiwari N. ECDLP-based certificateless proxy signature scheme with message recovery. Trans Emerg Tel Tech, 2015, 26: 346–354

21 Shi W B, He D B, Gong P. On the security of a certificateless proxy signature scheme with message recovery. Math Probl Eng, 2013, 2013: 761694

22 Lu Y, Li J G. Provably secure certificateless proxy signature scheme in the standard model. Theor Comput Sci, 2016, 639: 42–59

23 Huang X Y, Susilo W, Mu Y, et al. Certificateless designated verifier signature schemes. In: Proceedings of the 20th International Conference on Advanced Information Networking and Applications, Vienna, 2006. 15–19

24 Chen H, Song R S, Zhang F T, et al. An efficient certificateless short designated verifier signature scheme. In: Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing, Dalian, 2008

25 Du H Z, Wen Q Y. Effcient certificateless designated verifier signatures and proxy signatures. Chinese J Electron, 2009, 18: 95–100

26 Xiao Z B, Yang B, Li S G. Certificateless strong designated verifier signature scheme. In: Proceedings of the 2nd International Conference on E-Business and Information System Security, Wuhan, 2010

27 He D B, Chen J H. An efficient certificateless designated verifier signature scheme. Int Arab J Inf Technol, 2013, 10: 389–396

28 Lin H Y. A new certificateless strong designated verifier signature scheme: non-delegatable and SSA-KCA secure. IEEE Access, 2018, 6: 50765–50775

29 Huang X Y, Mu Y, Susilo W, et al. Certificateless signatures: new schemes and security models. Comput J, 2012, 55: 457–474

30 Caro A D, Iovino V. jPBC: Java pairing based cryptography. In: Proceedings of IEEE Symposium on Computers and Communications, Kerkyra, 2011. 850–855