Journal of King Saud University - Computer and Information Sciences 36 (2024) 102044

HOSTED BY

ELSEVIER

Contents lists available at ScienceDirect

Journal of King Saud University - Computer and Information Sciences

journal homepage: www.sciencedirect.com

Full Length Article

An efficient online/offline heterogeneous proxy signcryption for secure communication in UAV networks ☆

Negalign Wake Hundera [a,b], Wang Shumeng [a], Dagmawit Mesfin [c], Huiying Xu [a], Xinzhong Zhu [a,d,e,*]

[a] School of Computer Science and Technology, Zhejiang Normal University, Jinhua, 321004, China
[b] Zhejiang Institute of Optoelectronics, Jinhua, China
[c] Department of Electrical Engineering, University of Notre Dame, IN, 46556, USA
[d] AI Research Institute of Beijing Geekplus Technology Co., Ltd., Beijing, 100101, China
[e] Research Institute of Hangzhou Artificial Intelligence, Zhejiang Normal University, Hangzhou, 311231, China

ARTICLE INFO

Keywords:
Unmanned aerial vehicle
Certificateless cryptosystem
Internet of Things
Identity-based cryptosystem
Online/offline signcryption

ABSTRACT

The rapid growth of the Internet of Things (IoT) has led to an increased deployment of unmanned aerial vehicles (UAVs) across various sectors. However, efficiency and security issues are persistently among the primary challenges in UAV networks. In addition, significant communication delays can occur when UAVs perform remote tasks far from a command center (CC); in some cases, they may be unable to communicate with the CC. To address these challenges, in this paper, an efficient online/offline heterogeneous proxy signcryption scheme for secure communication in UAV networks (HOOPSC) is proposed. This scheme enables the CC in a certificateless cryptosystem (CLC) environment to delegate a nearby ground control station (GCS) to act as an agent, and directly send commands to the UAV within an identity-based cryptosystem (IBC) when the UAV undertakes remote tasks far from the CC. The UAV then decrypts and verifies commands for authenticity and confidentiality. In the proposed scheme, the signcryption process is split into offline and online phases, with most of the heavy computations conducted without the availability of the messages during the offline phase. Only light computations are performed in the online phase when a message is available. Moreover, a formal security proof is given in a random oracle model. Finally, a performance analysis reveals that HOOPSC outperforms existing relevant schemes, making it ideal for long-range operations in UAV networks.

1. Introduction

The rapid growth of wireless communication and the Internet of Things (IoT) has made unmanned aerial vehicle (UAV) technology increasingly popular in recent years. UAVs, or drone networks, are collaborative systems that use drones to accomplish tasks efficiently, and they can achieve specific objectives. UAVs are classified into three types based on their level of autonomy, those controlled by a remote operator, those supervised by a remote supervisor and those that operate without an operator or supervisor. There is no need for real-time control or monitoring with the third type of UAV because it is equipped with sensors and onboard computing. This enables it to operate independently, without the need for continuous control or supervision. The onboard computers respond to changes in the internal and environmental conditions tracked in real time by sensors. Once tasks are issued by the command center (CC), these unsupervised UAVs can autonomously perform them, ensuring the real-time processing of the gathered information. This paper focuses on the third category of UAVs, which are small, easy to operate, flexible and convenient for several

☆ This work was supported by the Zhejiang Normal University Postdoctoral Research Fund (Grant No. ZC304022985), the Natural Science Foundation of China (Project No. 61976196), and the Zhejiang Provincial Natural Science Foundation of China (Grant No. LZ22F030003).

* Corresponding author.
E-mail addresses: nigaccna21@zjnu.edu.cn (N.W. Hundera), wsmwsm@zjnu.edu.cn (W. Shumeng), dgeresu@nd.edu (D. Mesfin), xhy@zjnu.edu.cn (H. Xu), zxz@zjnu.edu.cn (X. Zhu).

Peer review under responsibility of King Saud University.

ELSEVIER Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2024.102044
Received 21 January 2024; Received in revised form 2 April 2024; Accepted 21 April 2024
Available online 4 May 2024

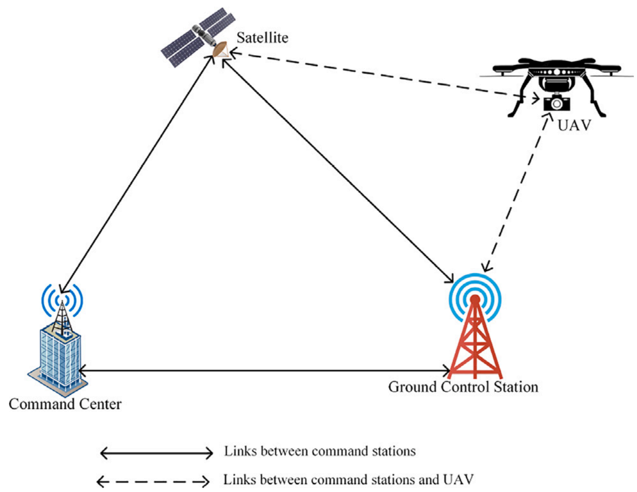

Fig. 1. Standard UAV scenario.

sectors, including military, agricultural, logistical and environmental monitoring (Zhou et al., 2023; Ge et al., 2020; Zhang et al., 2019; Faiçal et al., 2014; Khan et al., 2022a). However, UAV communication uses an open wireless network, which increases its vulnerability to various potential threats. Active attackers can intercept, manipulate and forge messages, whereas passive attackers can eavesdrop, making UAV communication security a critical issue and a hot topic among scholars (He et al., 2016; Mohsan et al., 2023; Pan et al., 2022; Khan et al., 2023).

The typical UAV scenario shown in Fig. 1, as described in Hua et al. (2021), consists of a UAV, CC, ground control station (GCS) and satellite that provides GPS navigation for the UAV and relays data transmissions. It is assumed that the UAV performs remote tasks far from the CC, leading to long delays or complete communication failures. If a UAV does not receive and verify commands in a timely manner, it may fly away from the target. In such cases, the UAV will have to approach the target again to execute the command, which will waste its power resources (Qi et al., 2019; Javed et al., 2022).

Therefore, it is critical to ensure that UAV commands are executed in real time. This is achieved by enabling the CC to delegate a nearby GCS to act as an agent and send commands directly to the UAV. The UAV then decrypts and verifies commands for authenticity and confidentiality. Advanced cryptographic methods, such as proxy signcryption, have been applied to enhance secure and efficient communications in UAV remote missions. Moreover, this study noted that UAVs, CCs and GCSs belong to different cryptographic infrastructures in a specific area. The three main public key cryptography infrastructures are public key infrastructure (PKI), identity-based cryptography (IBC) and certificateless cryptography (CLC). A PKI uses a certificate authority (CA) to link a user's public key with their identity, however, it can face challenges in certificate management, such as revocations and verifications (Spies, 2017). In IBC, where public keys are user identities such as email addresses or phone numbers, involves a private key generator (PKG) that generates secret keys, leading to key escrow issues (Barbosa and Farshim, 2008). CLC uses a key generation center (KGC) for master and partial private keys, allowing users to create secret keys while avoiding key escrow and certificate management problems (Li et al., 2022). Therefore, CLC is the best choice for control stations because it avoids key escrow and public key certificate management problems, whereas IBC, which is free from public key certificate management issues, is ideal for UAVs.

1.1. Motivation and contribution

This study aims to ensure secure and efficient communication between UAVs, CCs and GCSs operating within different cryptographic environments, addressing the issues associated with long-range operations when UAVs perform remote tasks far from a CC through proxy delegation. Because UAVs have limited computational and storage capacity, the scheme employs both online and offline approaches to reduce the computational and communication burden on UAV networks. The HOOPSC method is used to ensure secure communication within UAV networks.

The contributions of this study are as follows:

1. First, an efficient online/offline heterogeneous proxy signcryption method for secure communication in UAV networks (HOOPSC) is proposed. In this scheme, the CC and GCS operate within the CLC environment, which avoids the certificate management issues of the PKI and the key escrow problems of the IBC, and the UAV operates within the IBC environment, thus avoiding certificate management issues.
2. The proposed HOOPSC scheme splits signcryption into offline and online phases. In the offline phase, most heavy computations are performed without knowledge of the message. During the online phase, only light computations are performed when a message becomes available.
3. The HOOPSC scheme achieves confidentiality, integrity, authentication and nonrepudiation. Its security has been proven in terms of indistinguishability against adaptive chosen ciphertext attacks (IND-CCA2) and existential unforgeability against adaptive chosen message attacks (EUF-CMA) under the DBDH and CDH problems in the random oracle model.
4. An extensive evaluation was performed to establish that the proposed scheme outperforms existing schemes in terms of computational cost and communication overhead.

1.2. Related work

Secure communication is a crucial aspect of UAV networks because it ensures the confidentiality, integrity and authentication of the data shared between the UAV, CC and GCS. One approach for achieving secure communication in UAV networks is the use of a cryptographic technique. The conventional method of signing and then encrypting can keep messages secure from both active and passive attackers, however, its substantial computational burden renders it impracticable for UAVs. Zheng (1997) combined an approach in which encryption and digital signatures were implemented together in a single logical step, thereby reducing the computational load. In 2007, Baek et al. (2007) officially proved the security of signcryption using a random oracle model, demonstrating that signcryption can accomplish both encryption and digital signature security. Following this research, many signcryption techniques have been proposed (Li et al., 2017b; Niu et al., 2023; Saraswat et al., 2017; Yu et al., 2022; Zhou et al., 2019; Khan et al., 2022b). However, all of the above schemes use PKI, which involves certificate management, storage and time.

Considering the certificate management overhead. Shamir (1985) proposed IBC in 1984 to avoid PKI certificate management issues. In IBC, the user's public key is obtained from their identity data, whereas PKG produces a secret key; thus, a key escrow issue is unavoidable. PKG attack compromises or destroys system security. Therefore, the CLSC has been advanced (Barbosa and Farshim, 2008), where the complete private key is split into two parts, the user's partial private key is generated by the KGC, and the user creates its secret value; thus, the PKI and IBC issues were resolved. Since then, numerous CLSC techniques have been implemented (Mandal et al., 2020; Khan et al., 2021a; Xu et al., 2022; Chen et al., 2021; Khan et al., 2021b).

In cryptography, proxy signcryption combines the features of both proxy signatures and signcryptions. Mambo et al. (1996) first proposed proxy signatures; in this approach, the original signer delegates the right to sign to a proxy signer. This proxy signer is then authorized to create a legal signature on behalf of the original signer. Li et al. (2005)

presented the first certificateless proxy signature technique, however, its security aspects were not validated in the study. Later, Lu et al. (2007) and Cho and Lee (2007) found that the method (Li et al., 2005) was vulnerable to forgery attacks and proposed improved schemes. Nonetheless, formal security proofs are not provided in either Lu et al. (2007) or Cho and Lee (2007). Subsequently, several certificateless proxy signature schemes were developed Yang et al. (2020), Lu and Li (2016), Deng et al. (2016). The concept of proxy signcryption was first proposed by Gamage et al. (1999), in which an original signcrypter delegates its role to a proxy. This proxy securely signifies the confidentiality and authenticity of messages and the ensurers. This concept has led to the development of various proxy signcryption schemes, particularly those that use bilinear pairings (Lo et al., 2014; Shin et al., 2023; Zhou et al., 2018; Hundera et al., 2022). However, existing schemes based on bilinear pairings often suffer from high computational and communication costs, as well as key escrow issues. To address these problems, Yanfeng et al. (2013) introduced a CLP-IBSC system without bilinear pairings, but they did not provide security proofs. MING (2014) and Lo et al. proposed IBPSC schemes in the standard model in 2014. Yu et al. (2018) proposed a universally composable IBPSC system. Hundera et al. (2020) presented an IBPSC approach to cloud data sharing. Yu and Wang (2019) developed a CLPSC scheme that employs CMGs. Ming and Wang (2015) proposed a signcryption approach that uses a proxy. Zhou (2016) also proposed a proxy signcryption scheme. Waheed et al. (2020) examined the (Ming and Wang, 2015) scheme and provided an improved ECC method using a standard computational model. Recently, Qu and Zeng (2022) proposed CLPSC for UAV networks. However, this scheme incurs extensive computational and communication overhead. Moreover, all the above schemes are homogeneous and cannot be used in heterogeneous communication. Owing to the dynamic nature and complexity of the communication environment of UAV systems, different communication terminals may have different security requirements in different cryptographic environments. This means that consideration must be given to signcryption schemes for heterogeneous systems. In this paper, an efficient HOOPSC scheme is proposed that protects the integrity, authentication, and confidentiality of communication across all channels and addresses the issues associated with long-range operations in UAV networks.

1.3. Organization

The remainder of this paper is organized as follows. The preliminary work is introduced in Section 2. The formal model of the HOOPSC is presented in Section 3. An efficient HOOPSC scheme is presented in Section 4. A security and performance analysis are provided in Sections 5 and 6, respectively. Finally, the conclusions are presented in Section 7.

2. Preliminary work

In this section, notations, bilinear maps, and security requirements are provided.

2.1. Notations

Table 1 lists the acronyms used in the study.

Table 1
Acronym and description.

Acronym	Description
x_i	Secret value of users
d_i	Partial private key of users
sk_i	Private key of users
pk_i	Public key of users
S_{pc}	Proxy delegation
k_p	Proxy key
d_{ID_i}	Private key for IBC users
Q_i	Public key for IBC users
m_ω	Warrant
ID_A	Identity of the command center
ID_B	Identity of the ground control station
ID_C	Identity of the UAV
m	Message
P_{pub}	Master public key
s	Master secret key
$\hat{e}$	A bilinear map
$\mathbb{G}_1$	Cyclic additive group
$\mathbb{G}_2$	Cyclic multiplicative group
λ	Security parameter
δ	Offline Ciphertext
σ	Online Ciphertext
UKG	Universal Key Generation
$params$	System parameters

2.2. Bilinear maps

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups with the same prime order q; with $\mathbb{G}_1$ as an additive group and $\mathbb{G}_2$ as a multiplicative group, respectively. Let P be the generator of $\mathbb{G}_1$. A bilinear pairing is a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ that satisfies the following requirements:

1. Bilinearity: For all $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q^*$, $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.
2. Nondegeneracy: There are $P, Q \in \mathbb{G}_1$ such that $\hat{e}(P, Q) \neq 1$, 1 is $\mathbb{G}_2$ identity element.
3. Computability: $\hat{e}(P, Q)$ is efficiently calculated for all $P, Q \in \mathbb{G}_1$.

The modified Weil and Tate pairings offer acceptable maps of this type (Boneh and Franklin, 2001). The security of a HOOPSC relies on the hardness of the following problems.

Given $\mathbb{G}_1$ and $\mathbb{G}_2$, q, P and $\hat{e}$, similar to the above definition.

Definition 1: *Decisional Bilinear Diffie–Hellman Problem (DBDHP)*. Given a tuple $(P, aP, bP, cP) \in \mathbb{G}_1$ for some $a, b, c \in \mathbb{Z}_q^*$ and $h \in \mathbb{G}_2$, the *DBDHP* is used to determine if $h = \hat{e}(P, P)^{abc}$.

Definition 2: *Computational Diffie–Hellman Problem (CDHP)*. Given $(P, aP, bP) \in \mathbb{G}_1$ for some $a, b \in \mathbb{Z}_q^*$, the *CDHP* in $\mathbb{G}_1$ is used to calculate abP.

2.3. Security requirements

The communication between the UAV, CC and GCS should adhere to security properties such as confidentiality, integrity, authentication and nonrepudiation. Confidentiality ensures that the query commands remain secret from anyone except the authorized users. Integrity guarantees that the commands transmitted from the GCS are not altered by unauthorized parties. Authentication verifies that only authorized GCSs can access the UAV. Nonrepudiation prevents a GCS from denying previously submitted inquiries. That is, if a GCS sends inquiry commands to a UAV, the action cannot be denied.

3. Formal model of HOOPSC

In this section, the network model, framework and security considerations for HOOPSC are described.

3.1. Network model

An overview of the network model is shown in Fig. 2. The model consists of four types of entities.

1. ***UKG:*** A trusted third party responsible for registering the UAV, CC and GCS. It also generates partial private keys for the CC and GCS and private keys for the UAV. In this scenario, UKG functions as a PKG in IBCs, and as a KGC in CLCs.

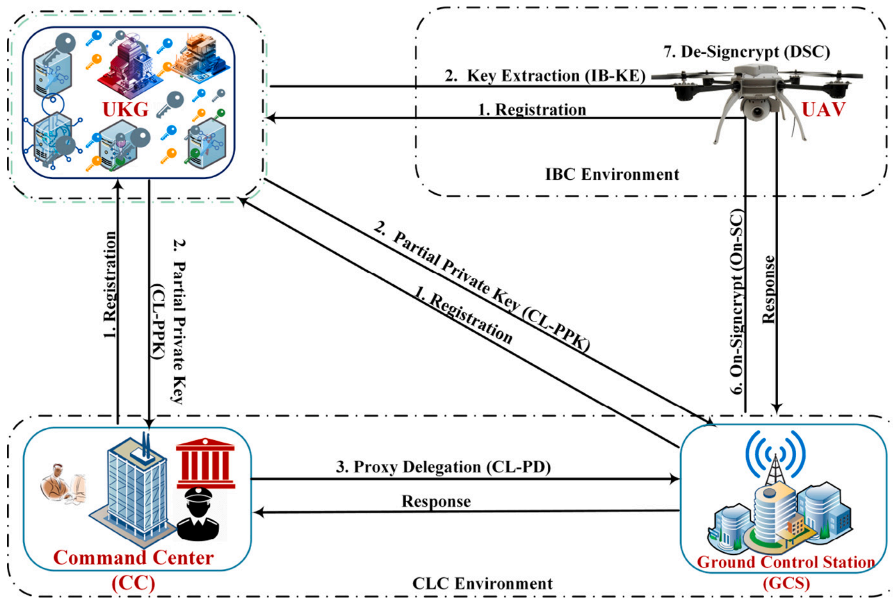

Fig. 2. The HOOPSC network model.

2. ***UAV:*** An entity that gathers data through onboard sensors and transmits them to the GCS and/or CC for further analysis, and follows orders from the CC or GCS.
3. ***Command Center:*** An entity responsible for the direct oversight and control of UAV operations. This entity delegates its controlling authority to GCS when a UAV performs a remote task far from the CC.
4. ***Ground Control Station:*** An entity that controls the overall operation of a UAV, including mission planning and real-time decision-making. The GCS acts on behalf of the CC using specialized information known as proxy delegation.

3.2. Framework

The generic HOOPSC scheme consists of the following twelve algorithms and involves the original delegator (CC), identified by ID_A, the proxy signcrypter (GCS), identified by ID_B and the UAV, identified by ID_C.

1. *Setup:* Executed by the *UKG*. A security parameter λ is taken as the input and outputs the master secret key s and the system parameters *params* that include the master public key P_{pub}. For simplicity, *params* is excluded from the descriptions of the other algorithms in the subsequent content.
2. *CL-PPK:* Run by the *UKG*, takes the master secret key s and a user's identity $ID_i \in \{0,1\}^*$ as inputs. It outputs partial private keys d_i.
3. *CL-SV:* It generates a secret value. User's identity ID_i is used as input and outputs a secret value x_i.
4. *CL-SK:* This is a full private key generation algorithm run by a user. It takes the partial private key d_i and a secret value x_i as input and outputs the full private key S_{k_i}
5. *CL-PK:* Users perform the algorithm. It takes a secret value x_i as input and outputs the public key P_{k_i}.
6. *IB-KE:* It is a key extraction algorithm executed by the *UKG*. It takes a master secret key s and an identity $ID_i \in \{0,1\}^*$ as inputs and outputs a private key d_{ID_i}.
7. *CL-PD:* Run by CC. The private key S_{k_A} and the public key P_{k_A} of the CC, along with the warrant m_ω, are taken as the inputs and outputs of the proxy delegation s_{pc}. The warrant m_ω includes details about the duration of a delegation and the identities of both the CC and GCS.
8. *CL-DV:* Executed by GCS. The warrant m_ω, proxy delegation s_{pc}, identity ID_B and public key pk_A are considered as inputs and verifies whether s_{pc} is from a legitimate user.
9. *CL-PRK:* Run by GCS. The warrant m_ω, the proxy delegation s_{pc} and private key S_{k_B} of the GCS are taken as the input and the proxy key k_p is the output.
10. *Off-SC:* Performed by GCS. It takes the identity ID_C of the UAV as the input and outputs an offline ciphertext δ.
11. *On-SC:* Run by GCS. The CC identity, GCS identity, UAV identity, proxy key k_p, warrant m_ω, the offline ciphertext δ and message m are used as input. The full ciphertext σ is the output.
12. *DSC:* Executed by the UAV, the private key d_{ID_C} of the UAV and the full ciphertext σ are taken as inputs. It outputs either m or $\perp$, indicating that σ is not a valid ciphertext.

The algorithms should meet the HOOPSC consistency constraint. If $\delta = Off\text{-}SC(Q_C, ID_C)$ and $\sigma = On\text{-}SC(\delta, k_p, m_\omega, m, ID_A, ID_B, ID_C)$, then $m = DSC(\sigma, ID_C, d_{ID_C})$. Note that *the CL-PPK, CL-SV, CL-SK, CL-PK, CL-PD, CL-DV* and *CL-PRK* algorithms are used for CLC users, whereas the *IB-KE* algorithm is used for IBC users.

3.3. Security notions

The proposed HOOPSC scheme ensures confidentiality *(IND-CCA2)* and unforgeability *(EUF-CMA)*. The concepts in Li et al. (2017a, 2016) were modified with minor adjustments to the HOOPSC.

3.3.1. Confidentiality

For confidentiality, the game between an adversary $\mathcal{A}$ and a challenger C is examined.

IND-CCA2: C interacts with $\mathcal{A}$.

Initial: C performs the *setup* with λ and sends *params* to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ makes polynomially limited requests.

1. *Partial private key inquiries:* $\mathcal{A}$ chooses $ID_i \in \{0,1\}^*$ and sends ID_i to C. C runs the *CL-PPK* algorithm and returns d_i to $\mathcal{A}$ as a partial private key.
2. *Private key inquiries:* $\mathcal{A}$ chooses $ID_i \in \{0,1\}^*$. C first computes the *CL-SV* and *CL-PPK*; then, it performs *CL-SK* and yields the full private key sk_i to $\mathcal{A}$.

3. *Public key inquiries:* $\mathcal{A}$ chooses $ID_i \in \{0,1\}^*$. $\mathcal{C}$ computes *CL-PK* and returns the public key pk_i to $\mathcal{A}$.
4. *Public replacement query:* $\mathcal{A}$ can replace pk_i with a value of its choice.
5. *Key extraction inquiries:* $\mathcal{A}$ chooses $ID_i \in \{0,1\}^*$. $\mathcal{C}$ computes *IB-KE* and returns private key d_{ID_i} to $\mathcal{A}$.
6. *Proxy delegation queries:* $\mathcal{A}$ selects $ID_i \in \{0,1\}^*$. $\mathcal{C}$ first computes the *CL-SK* and *CL-PK* algorithms, then performs the *CL-PD* and returns the proxy delegation S_{pc} to $\mathcal{A}$.
7. *Proxy key inquiries:* $\mathcal{A}$ selects two identities ID_i and ID_j. $\mathcal{C}$ first computes the *CL-PD* and *CL-SK* algorithms on ID_i and ID_j, respectively, to obtain s_{pc} and sk_j. Then, $\mathcal{C}$ runs *CL-PRK* and sends a proxy key K_p to $\mathcal{A}$.
8. *Designcrypt inquiries*: $\mathcal{A}$ provides a sender's identity ID_i, public key pk_i, receiver's identity ID_j, and ciphertext σ. $\mathcal{C}$ first performed the *IB-KE* process to extract d_{ID_j}. Then, $\mathcal{C}$ computes *Designcrypt* $(\sigma, ID_i, pk_i, ID_j, d_{ID_j})$ and returns the outcome to $\mathcal{A}$. The outcome is whether m or $\perp$.

Challenge: $\mathcal{A}$ determines when *Phase 1* is concluded. $\mathcal{A}$ chooses two messages of equal length, m_0 and m_1; sender ID_s; and receiver ID_r identities that it likes to challenge. $\mathcal{C}$ first runs *CL-PRK* to generate the proxy key k_p and runs the *IB-KE* to retrieve the public key of the receiver Q_r. Then, $\mathcal{C}$ selects a random bit $\eta \in \{0,1\}$ and determines $\delta = Off\text{-}SC(Q_r, ID_r)$ and $\sigma = On\text{-}SC(\delta, k_p, m_\eta, ID_s, ID_r)$. Finally, $\mathcal{C}$ sends σ to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ performs polynomially limited requests, as in *Phase 1*. This time, *Designcrypt* inquiry cannot be performed on (σ, ID_s, ID_r) to obtain m unless pk_s have been substituted after the challenge phase and key extract inquiries on the ID_r.

Guess: $\mathcal{A}$ creates ϑ^* and if $\vartheta^* = \vartheta$, then $\mathcal{A}$ wins the game.

$\mathcal{A}$'s advantage is defined as follows:

$\mathrm{Adv}(\mathcal{A}) = |2\Pr[\vartheta^* = \vartheta] - 1|$, where $\Pr[\vartheta^* = \vartheta]$ indicates the probability that $\vartheta^* = \vartheta$.

Definition 3: *HOOPSC* scheme is $(\varepsilon, t, q_{ppk}, q_{sk}, q_{pk}, q_{pkr}, q_{ke}, q_{pd}, q_{kp}, q_{dsc})$*–IND-CCA2* secure if no polynomial time adversaries $\mathcal{A}$ runs at a time of t and has an advantage of at least ε after at most q_{ppk} partial private key inquiries, q_{sk} private key inquiries, q_{pk} public key inquiries, q_{pkr} public key replacement inquiries, q_{ke} key extraction inquiries, q_{pd} proxy delegation inquiries, q_{kp} proxy key inquiries and q_{dsc} designcrypt inquiries in *IND-CCA2*. See *Section* 5 for a security proof. The definition of insider security incorporates signcryption confidentiality, assuming that the adversary knows all the sender secret keys (An et al., 2002).

3.3.2. Unforgeability

Here, because senders are in the CLC, consideration must be given to two types of adversaries for unforgeability, Type I and Type II (Li and Xiong, 2013; Li et al., 2013). A *Type I adversary* is an attacker who can forge or replace public keys but lacks access to the UKG master key. A *Type II adversary* is a UKG that knows the master secret key; however, such an adversary cannot alter the user's public keys. The security model of HOOPSC for unforgeability is established using two adversary games, EUF-CMA-I and EUF-CMA-II, involving Type I ($\mathcal{F}_1$) and type II ($\mathcal{F}_2$) adversaries that act against challengers ($\mathcal{C}$).

EUF-CMA-I: Here, $\mathcal{C}$ is played with $\mathcal{F}_1$.

Initial: $\mathcal{C}$ run the *setup* as in the *IND-CCA2* game.

Attack: $\mathcal{F}_1$ performs *partial private key inquiries*, *private key inquiries*, *public key inquiries*, *key extraction queries*, *proxy delegation inquiries*, and *proxy key inquiries* as in the *IND-CCA2* game. In a *signcrypt inquiry*, $\mathcal{F}_1$ sends ID_s, ID_r and σ to $\mathcal{C}$. $\mathcal{C}$ first runs *CL-PRK* to generate the proxy key k_p and runs the *IB-KE* to obtain the public key of the receiver Q_r. Then, $\mathcal{C}$ runs $\delta = Off\text{-}SC(Q_r, ID_s)$ and $\sigma = On\text{-}SC(\delta, k_p, m, ID_s, ID_r)$. Finally, $\mathcal{C}$ sends σ to $\mathcal{F}_1$.

Forgery: $\mathcal{F}_1$ generates a tuple (σ^*, ID_s, ID_r) and achieves success if the following conditions are met:

1. $\mathcal{F}_1$ is prohibited from extracting *private key inquiries* on ID_s.
2. *Designcrypt* $(\sigma^*, ID_s, pk_s, ID_r, d_{ID_r}) = m^*$
3. $\mathcal{F}_1$ cannot extract *proxy key inquiries* with (ID_s, ID_r), and *key extraction inquiries* cannot be obtained on ID_r.
4. $\mathcal{F}_1$ cannot request partial private key inquiries or public key replacement inquiries on ID_s.
5. $\mathcal{F}_1$ cannot ask for a signcrypt inquiry on (m^*, ID_s, ID_r).

The advantages of $\mathcal{F}_1$ represent success probability.

Definition 4: *HOOPSC* scheme is $(\varepsilon, t, q_{ppk}, q_{sk}, q_{pk}, q_{pkr}, q_{ke}, q_{pd}, q_{kp}, q_{sc})$*–EUF-CMA-I* secure if no polynomial time adversaries $\mathcal{F}_1$ who runs at a time of t and has an advantage of at least ε after at most q_{ppk} partial private key inquiries, q_{sk} private key inquiries, q_{pk} public key inquiries, q_{pkr} public key replacement inquiries, q_{ke} key extraction inquiries, q_{pd} proxy delegation inquiries, q_{kp} proxy key inquiries and q_{sc} signcrypt inquiries in *EUF-CMA-I*. See *Section* 5 for a security proof.

EUF-CMA-II: Here, $\mathcal{C}$ plays with $\mathcal{F}_2$.

Initial: $\mathcal{C}$ runs the *setup* with λ and sends *params* and s to $\mathcal{F}_2$.

Attack: $\mathcal{F}_2$ executes the *private key*, *public key*, and *signcrypt inquiries* as in the *EUF-CMA-I* game. Here, we note that there are no *partial private keys* or *key extraction inquiries* in this game because $\mathcal{F}_2$ knows the master's private key s.

Forgery: $\mathcal{F}_2$ generates a tuple (σ^*, ID_s, ID_r) and achieves success if the following conditions are met:

1. $\mathcal{F}_2$ is prohibited from extracting *a private key query* on ID_s.
2. *Designcrypt* $(\sigma^*, ID_s, pk_s, ID_r, d_{ID_r}) = m^*$
3. $\mathcal{F}_2$ cannot extract the *proxy key query* with (ID_s, ID_r), and it cannot make the *key extraction query* on ID_r.
4. $\mathcal{F}_2$ cannot ask a signcrypt query on (m^*, ID_s, ID_r).

The advantage of $\mathcal{F}_2$ is the success probability.

Definition 5: *The HOOPSC* scheme is $(\varepsilon, t, q_{sk}, q_{pk}, q_{pkr}, q_{pd}, q_{kp}, q_{sc})$*–EUF-CMA-II* secures if no polynomial time adversaries $\mathcal{F}_2$ who run at most times t and has the advantage of at least ε after at most q_{sk} private key inquiries, q_{pk} public key inquiries, and q_{sc} signcrypt inquiries in *EUF-CMA-II*.

See Section 5 for a security proof. In Definitions 2 and 3, the adversary can obtain the secret key of the receiver. This definition encompasses insider security for unforgeable signcryption (An et al., 2002).

4. HOOPSC scheme

In this section, an efficient HOOPSC scheme is proposed. It is assumed that the UAV is tasked with a remote task that requires long flight distances from the command center (CC). In such scenarios, the CC identified by ID_A delegates its authority to the GCS, as identified by ID_B. The GCS then issues commands directly to the UAV on behalf of the CC. The UAV, identified by ID_C, decrypts and verifies the commands to ensure its authenticity and confidentiality. In this scheme, the CC and GCS operate in the CLC domain, and the UAV operates in the IBC domain. Additionally, the UKG serves as a trusted third party, generating a partial private key for users in the CLC environment and a private key for those in the IBC environment. The scheme consists of the following twelve algorithms.

1. *Setup* (λ): Given a security parameter λ, *UKG* chooses groups $\mathbb{G}_1$ (additive) and $\mathbb{G}_2$ (multiplicative) with prime order q, generator P of $\mathbb{G}_1$, bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, and hash functions: $H_1 : \{0,1\}^* \times \mathbb{G}_1 \to \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{Z}_q^*$, $H_3 : \mathbb{G}_1 \to \{0,1\}^n$ and $H_4 : \{0,1\}^* \to \mathbb{Z}_q^*$, where $\{0,1\}^n$ is the message space. The *UKG* selects a master secret key $s \in \mathbb{Z}_q^*$ at random and calculates the master public key $P_{pub} = sP$. Finally, the *UKG* publishes $params = \{\mathbb{G}_1, \mathbb{G}_2, \hat{e}, q, P, P_{pub}, n, H_1, H_2, H_3, H_4\}$ and keeps the master secret key s secret.

2. *CL-PPK*: Given an identity $ID_i \in \{0,1\}^*$, *UKG* selects $r_i \in \mathbb{Z}_q^*$, computes $k_i = r_i P$ and $d_i = r_i + y_i s$, where $y_i = H_1(ID_i, k_i)$ and sends (d_i, k_i) to the user.
3. *CL-SV*: A user with ID_i verifies whether $d_i P = k_i + y_i P_{pub}$ holds true. After successful verification, the user selects a secret value $x_i \in \mathbb{Z}_q^*$ and computes $p_i = x_i P$.
4. *CL-SK*: Given x_i and d_i, a user in the CLC sets its full private key $S_{k_i} = (x_i, d_i)$.
5. *CL-PK*: Given k_i and p_i, a user in the CLC sets $P_{k_i} = (k_i, p_i)$ is their public key.
6. *IB-KE*: Given a user's identity ID_C, the *UKG* randomly chooses $r_C \in \mathbb{Z}_q^*$ and computes $k_{ID_C} = r_C P$ and $d_{ID_C} = r_C + y_{ID_C} s$, where $y_{ID_C} = H_1(ID_C, k_{ID_C})$. The *UKG* then securely transmits (d_{ID_C}, k_{ID_C}) to the UAV. The UAV verifies the validity of the private key by checking whether $d_{ID_C} P = k_{ID_C} + y_{ID_C} P_{pub}$ holds true.
7. *CL-PD*: Given the private and public key pair (S_{k_A}, P_{k_A}) of the CC and the warrant m_ω, the CC in the CLC executes the delegation process as follows:
   (a) Randomly select $a \in \mathbb{Z}_q^*$.
   (b) Compute $D = aP$.
   (c) Compute $t = a + R_1(d_A + x_A z_A)$, where $R_1 = H_2\left(m_\omega || ID_B, P_{k_A}, D, P_{pub}\right)$ and $z_A = H_1\left(ID_A, p_A\right)$.
   (d) Finally, the proxy delegation $S_{pc} = (ID_A, ID_B, P_{k_A}, m_\omega, D, t)$ is sent to the GCS.
8. *CL-DV*: To verify a delegation, the proxy signcrypter (GCS) checks whether

$$tp = R_1(k_A + y_A P_{pub} + z_A p_A) + D,$$

   where

$$R_1 = H_2\left(m_\omega || ID_B, P_{k_A}, D, P_{pub}\right)$$
$$z_A = H_1\left(ID_A, p_A\right),$$
$$y_A = H_1(ID_A, k_A).$$

   Otherwise, the GCS rejects the delegation request.
9. *CL-PRK*: Upon successful verification, the GCS computes the proxy key

$$k_p = t + R_2(d_B + x_B z_B),$$

   where:

$$R_2 = H_2\left(m_\omega || ID_A, P_{k_B}, D, P_{pub}\right),$$
$$z_B = H_1\left(ID_B, p_B\right).$$

10. *Off-SC* : Given the identity ID_C of the UAV. The GCS then performs the *Off-SC* process as follows:
   (a) Chooses $x \in \mathbb{Z}_q^*$.
   (b) Compute $U = xP$.
   (c) Compute $Q_C = k_{ID_C} + y_{ID_C} P_{pub}$, where $y_{ID_C} = H_1(ID_C, k_{ID_C})$.
   (d) Compute $v = xQ_C$.
   (e) Compute $h_1 = H_3(v)$.
   (f) Output $\delta = (U, h_1)$.
11. *On-SC:* Given a message m, proxy key k_p, warrant m_ω, *Off-SC* δ, and the identities ID_A, ID_B, and ID_C, corresponding to the CC, GCS, and UAV, respectively. The algorithm is as follows:
   (a) Compute $h_2 = H_4\left(m \| ID_A \| ID_B || ID_C\right)$.
   (b) Compute $C = m \oplus h_1$.
   (c) Compute $S = (x + h_2 k_p)$.
   (d) Output $\sigma = (m_\omega, S, C, U)$.

   Then, the GCS sends σ to the UAV.
12. *DSC*: Upon receiving the signcrypted ciphertext $\sigma = (m_\omega, S, C, U)$, the UAV accepts the message only if the following holds:
   (a) Compute $v = d_{ID_C} U$.
   (b) Compute $h_1 = H_3(v)$.
   (c) Compute $m = C \oplus h_1$.
   (d) Accept the message if

$$SP = h_2\left(D + R_1(k_A + z_A p_A + y_A P_{pub}) + R_2(k_B + z_B p_B + y_B P_{pub})\right) + U$$

   return $\perp$ otherwise.
   The following describes how the message decryption process works:

$$\begin{aligned}
m &= C \oplus h_1 \\
&= C \oplus H_3(d_{ID_C} U) \\
&= m \oplus H_3(xQ_C) \oplus H_3(d_{ID_C} U) \\
&= m \oplus H_3(xQ_C) \oplus H_3((r_C + y_{ID_C} s)xP) \\
&= m \oplus H_3(xQ_C) \oplus H_3((k_{ID_C} + y_{ID_C} P_{pub})x) \\
&= m \oplus H_3(xQ_C) \oplus H_3(xQ_C) \\
m &= m
\end{aligned}$$

4.1. Correctness analysis

The HOOPSC scheme consists of four authentication steps:

1. Partial private key verification, where the users in the CLC environment check if

$$d_A P = k_A + y_A P_{\text{pub}}$$
$$= r_A P + y_A s P$$
$$= (r_A + y_A s)P$$
$$d_A P = k_A + y_A P_{\text{pub}}$$

2. Verification of delegation: The proxy signcrypter (GCS) in the CLC environment

$$tp = R_1(k_A + y_A P_{pub} + z_A p_A) + D$$
$$tp = (a + R_1(d_A + x_A z_A))P$$
$$= (aP + R_1(d_A P + x_A z_A P))$$
$$= D + R_1(d_A P + p_A z_A)$$
$$= D + R_1((r_A + y_A s)P + p_A z_A)$$
$$tp = D + R_1(k_A + y_A P_{pub} + p_A z_A)$$

3. The private key verification, where the UAV in the IBC environment checks if

$$d_{ID_C} P = k_{ID_C} + y_{ID_C} P_{pub}$$
$$= r_C P + y_{ID_C} s P$$
$$= (r_C + y_{ID_C} s)P$$
$$d_{ID_C} P = k_{ID_C} + y_{ID_C} P_{pub}$$

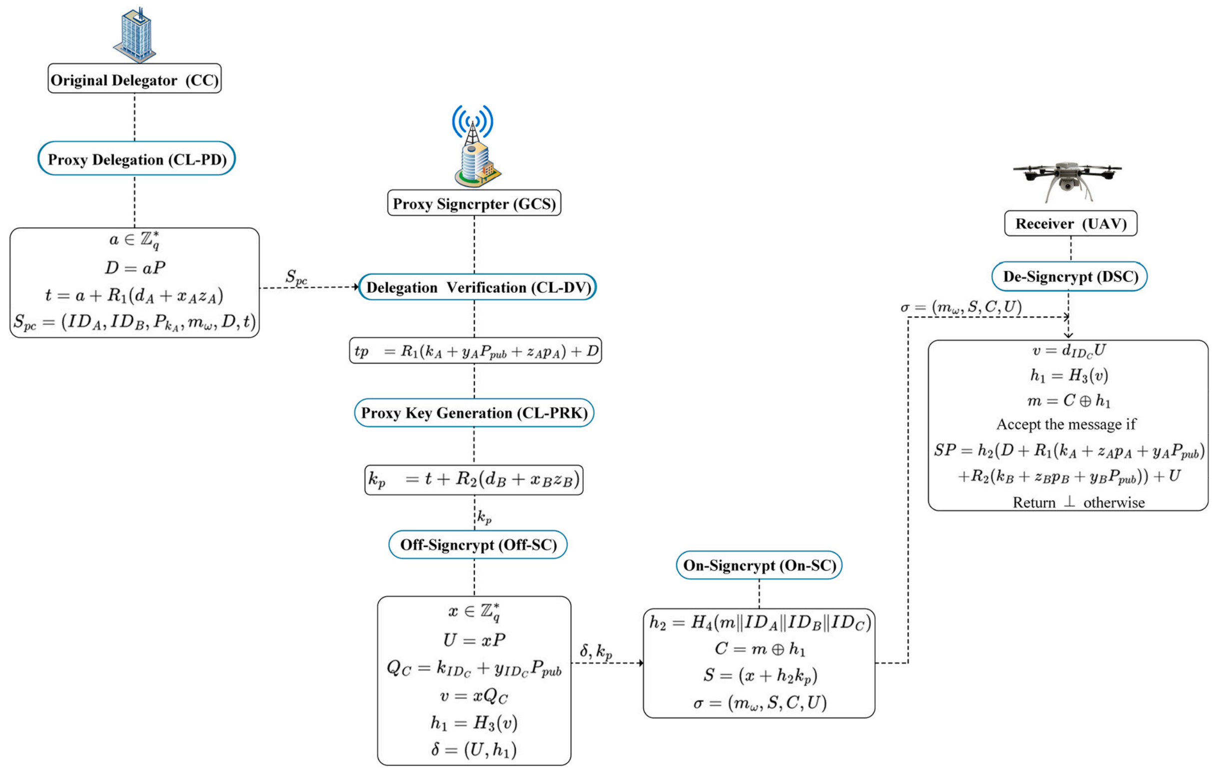

Fig. 3. Efficient HOOPSC communication.

4. In the *DSC* process, the message receiver checks if

$$SP = h_2\left(D + R_1(k_A + z_A p_A + y_A P_{pub}) + R_2(k_B + z_B p_B + y_B P_{pub})\right) + U$$

$$\begin{aligned} SP &= (x + h_2 k_p)P \\ SP &= xP + h_2(t + R_2(d_B + x_B z_B))P \\ &= U + h_2(tP + R_2(d_B P + x_B z_B P)) \\ &= U + h_2(tP + R_2((r_B + y_B s)P + p_B z_B)) \\ &= U + h_2(tP + R_2(k_B + y_B P_{pub} + p_B z_B)) \\ &= U + h_2((a + R_1(d_A + x_A z_A))P \\ &\quad + R_2(k_B + y_B P_{pub} + p_B z_B)) \\ &= U + h_2(aP + R_1(d_A + x_A z_A)P \\ &\quad + R_2(k_B + y_B P_{pub} + p_B z_B)) \\ &= h_2(D + R_1(d_A P + p_A z_A) \\ &\quad + R_2(k_B + y_B P_{pub} + p_B z_B)) + U \\ &= h_2(D + R_1((r_A + y_A s)P + p_A z_A) \\ &\quad + R_2(k_B + y_B P_{pub} + p_B z_B)) + U \\ SP &= h_2(D + R_1(k_A + y_A P_{pub} + p_A z_A) \\ &\quad + R_2(k_B + y_B P_{pub} + p_B z_B)) + U \end{aligned}$$

Here, Fig. 3 shows the efficient HOOPSC communication.

5. Security analysis

It is demonstrated that the HOOPSC meets the confidentiality and unforgeability requirements in Theorems 1 and 2.

5.1. Confidentiality

Theorem 1. *In the random oracle model, if the adversary $\mathcal{A}$ holds a nonnegligible advantage ϵ in compromising the IND-CCA2 security of the HOOPSC scheme within time frame t and performing q_{ppk} inquiries, q_{sk} inquiries, q_{pk} inquiries, q_{pkr} inquiries, q_{ke} inquiries, q_{pd} inquiries, q_{kp} inquiries, q_{dsc} inquiries, and q_{H_i} inquiries to oracles H_i $(i = 1, 2, 3, 4)$, then there is a C that can solve the DBDHP with an advantage*

$$\epsilon_{dbdh} \geq \left(\frac{\epsilon}{q_{H_1}}\right)\left(1 - \frac{q_{sc}\left(q_{H_2} + q_{H_3} + q_{H_4}\right)}{2^\lambda}\right)\left(1 - \frac{q_{dsc}}{2^\lambda}\right)$$

at time

$$t' \leq t + O\left(q_{kp} + q_{sc} + q_{dsc} q_{H_2}\right) t_p,$$

where t_p represents the time for a single pairing operation.

Proof. It is illustrated how C utilizes $\mathcal{A}$ as a function to resolve a given scenario (P, aP, bP, cP, h) of the *DBDHP*.

Initial: C randomly chooses $s \in \mathbb{Z}_q^*$ and sets $P_{pub} = sP$. C also establishes the receiver public key $Q_C = aP$. *params*, and Q_C are then sent to $\mathcal{A}$. Note that C is unaware of the values of $a \in \mathbb{Z}_q^*$.

Phase 1: C maintains a list L_i (where i ranges from 1 to 4) to simulate hash oracles H_1, H_2, H_3 and H_4, respectively. It also stores a list L_k to store the private and public key information, L_{pk} for the proxy key. The assumptions made are that the queries in H_1 are distinct and that $\mathcal{A}$ requests the queries in $H_1(ID_i)$ prior to the identity ID_i being utilized in the remaining queries. Furthermore, by employing the irreflexivity assumption (Boyen, 2003), it is assumed that the identities of the sender and receiver are distinct. Initially, all the lists are empty. When $\mathcal{A}$ queries, C picks a random ℓ from $(1, \ldots, q_{H_1})$ and answers $\mathcal{A}$'s queries as follows.

H_1 inquiries: For $H_1(ID_i, k_i)$ on the chosen identity ID_i. Initially, C verified whether H_1 was defined for the input (ID_i, k_i). If a query matches, then the previous value is returned. Otherwise, C chooses $y_i \in \mathbb{Z}_q^*$ and adds (ID_i, k_i, y_i) to L_1.

H_2 inquiries: For $H_2(m_\omega || ID_i, P_{k_j}, D, P_{pub})$ query, C first verifies whether the entry is in L_2. Return the previously set value if so.

Otherwise, C picks a random $h_{2i} \in \mathbb{Z}_q^*$ and appends the tuple $(m_\omega ||ID_i, P_{k_j}, Q, P_{pub}, h_{2i})$ into list L_2.

H_3 inquiries: For $H_3(v_i)$ inquiries, initially, C verified whether H_3 was defined for input v_i. Returns the previously defined value if so. Otherwise, C randomly selects h_{3i} from $\{0,1\}^n$, returns it as a response, and adds tuple (v_i, h_{3i}) to list L_3.

H_4 inquiries: For $H_4(m_i \| ID_i \| ID_j || ID_c)$ query, C first verifies whether the entry is in L_4. If true, C gives the current response; otherwise, it yields a random $h_{4i} \in \mathbb{Z}_q^*$ to $\mathcal{A}$. Furthermore, C performs simulations on the H_3 oracle to obtain $h_{3i} = H_3(v_i) \in \{0,1\}^n$, computes $C_i = m_i \oplus h_{3i}$ and sets $\xi = d_{ID_i}.h_{4i}$ to manage future designcryption queries. Finally, C inserts the tuple $(m_i \| ID_i \| ID_j || ID_c), C_i, \xi$ into list the L_4.

Partial private key inquiries: Partial private key inquiries on identity ID_i are made by $\mathcal{A}_1$. If $ID_i = ID_\ell$, the process terminates. Otherwise, C checks L_k and returns an existing value. Otherwise, C:

1. Selects $r_i,\ x_i,\ y_i \in \mathbb{Z}_q^*$ randomly.
2. Computes $p_i = x_i P$, $k_i = r_i P$, and $d_i = r_i + y_i s$.
3. Adds (ID_i, k_i, d_i) to L_k and $((ID_i, k_i), y_i)$ to L_1.

C then sends (d_i, k_i) to A_1.

Private key inquiries: $\mathcal{A}$ issues private key inquiry on identity ID_i. If $\mathrm{ID}_i = \mathrm{ID}_\ell$, the process fails. Otherwise, C randomly selects $x_i \in \mathbb{Z}_q^*$, returns $sk_i = (x_i, d_i)$ and adds (ID_i, k_i, x_i, d_i) to L_k. Here, d_i is obtained from a previous *partial private key inquiry* using ID_i.

Public key inquiries: $\mathcal{A}$ chooses ID_i and forwards it to C. If list L_k has a set $(ID_i, k_i, p_i, P_{k_i})$, then C returns P_{k_i} to $\mathcal{A}$. Otherwise, C selects a random $e_i, \alpha_i \in \mathbb{Z}_q^*$, calculates $p_i = e_i P$ and $k_i = \alpha_i P$, returns $P_{k_i} = (k_i, p_i)$ to $\mathcal{A}$ and adds $(ID_i, k_i, p_i, P_{k_i})$ to L_k.

Public key replacement inquiries: For q_{pkr} inquiry on $(ID_i, k_i, p_i, P_{k_i})$, C updates the list L_k with tuple $(ID_i, \perp, \perp, P_{k_i})$, where $\perp$ indicates an unknown number.

Key extraction inquiries: $\mathcal{A}_1$ query identity ID_i for key extraction inquiries. If $ID_i = ID_\ell$, the process terminates. Otherwise, C checks L_k and returns an existing value. Otherwise, C:

1. Selects r_i and $y_i \in \mathbb{Z}_q^*$ randomly.
2. Computes $k_{ID_i} = r_i P$, $d_{ID_i} = r_i + y_i s$
3. Adds $(ID_i, k_{ID_i}, d_{ID_i})$ to L_k and $((ID_i, k_{ID_i}), d_{ID_i})$ to L_1.

C then sends the private key (d_{ID_i}, k_{ID_i}) to $\mathcal{A}_1$.

Proxy delegation queries: Upon receiving a proxy delegation query from $\mathcal{A}_1$ on $\left(ID_i, ID_j, m_\omega\right)$, C execute the proxy delegation query. It then sends the result $S_{pc} = (ID_i, ID_j, P_{k_i}, m_\omega, D, t)$ to $\mathcal{A}_1$ and adds $(ID_i, ID_j, P_{k_i}, m_\omega, D, t)$ to L_{pk}.

Proxy key inquiries: When $\mathcal{A}_1$ asks a proxy key query, C checks for tuple $(ID_i, ID_j, P_{k_i}, m_\omega, D, t)$ in L_{pk}. If this is found, then the proxy key K_P is returned. Otherwise, C :

1. A proxy delegation query is used to obtain S_{pc}.
2. Search L_k for ID_j to obtain the secret key sk_j.
3. Compute $k_p = t + R_2(d_j + x_j z_j)$, add the tuple to L_{pk}, and send k_p to $\mathcal{A}_1$.

Designcrypt queries: $\mathcal{A}$ chooses a ciphertext $\sigma = (m_\omega, S, C, U)$, and C operates as follows.

1. If $ID_s \neq ID_r$, then C first runs the inquiry *public key inquiry* for ID_s and *key extraction inquiry* for ID_r to obtain P_{k_s} and d_{ID_r}; then, C computes $v = d_{ID_r} U$ and runs H_3 queries on (v) to obtain h_1 and returns $m = C \oplus h_1$.
2. If $ID_s = ID_r$, C cannot obtain d_{ID_r} via the *key extraction query*. Here, v cannot be calculated. To ensure consistency, C searches for a tuple (v, h_1) in L_3 for various v values, such that *DBDH* $(aP, bP, v) = v$. If this item is present, then the correct v and h_1 values are determined. C then obtains h_1 by calling an H_4 query on $H_4\left(m \| ID_A \| ID_B || ID_C\right)$ and checks if
$$SP = h_2\left(D + R_1(k_{\mathrm{A}} + z_{\mathrm{A}} p_{\mathrm{A}} + y_{\mathrm{A}} P_{\mathrm{pub}}) + R_2(k_{\mathrm{s}} + z_{\mathrm{s}} p_{\mathrm{s}} + y_{\mathrm{s}} P_{\mathrm{pub}})\right) + U,$$
If this is true, C returns $m = C \oplus h_1$. Otherwise, the ciphertext is rejected, and $\perp$ returns.
3. When C reaches this point in its process, it puts a random $h_1 \in \mathbb{Z}_q^*$ in L_3, that is, $(U, *, h_1)$ for an unknown value of v and a random $h_2 \in \mathbb{Z}_q^*$ in list L_4 $\left(m \| ID_A \| ID_B || ID_C\right)$. Finally, C determines whether or not.
$$SP = h_2\left(D + R_1(k_{\mathrm{A}} + z_{\mathrm{A}} p_{\mathrm{A}} + y_{\mathrm{A}} P_{\mathrm{pub}}) + R_2(k_{\mathrm{s}} + z_{\mathrm{s}} p_{\mathrm{s}} + y_{\mathrm{s}} P_{\mathrm{pub}})\right) + U,$$
If this is true, C returns $m = C \oplus h_1$ to $\mathcal{A}$. Otherwise, the symbol $\perp$ is returned and the ciphertext is rejected. The symbol $*$ is linked to the identity ID_r. In scenarios (1) and (2), a failure occurs for the challenger if either the hash value h_1 or h_2 has been previously established in the list

Challenge: $\mathcal{A}$ generates two plaintexts of identical lengths, m_0 and m_1. To challenge a sender, ID_s and a receiver's identity ID_r must be used. If $ID_s \neq ID_r$, C fails. Otherwise, C uses a random bit $b \in \{0,1\}^n$ to signcrypt m_b. A random hash value $S^*, h_1, h_2 \in \mathbb{Z}_q^*$ is chosen, and $U^* = aP$ and $S^* = (x + h_2 k_p) = (taP + h_2(sk_s, pk_s))$ are set. Finally, C computes $C^* = m_b \oplus h_1$ and returns $\sigma_p^* = (S^*, C^*, U^*)$ to $\mathcal{A}$.

Guess: $\mathcal{A}_1$ produces a guess bit δ^* and wins if $\delta^* = \delta$. If $h = \hat{e}(P,P)^{abc}$, C returns 1; otherwise, it returns 0, illustrating $h \neq \hat{e}(P,P)^{abc}$.

$\mathcal{A}_1$'s advantage is defined as

$$\mathrm{Adv}_{\mathrm{HOOPSC}}^{\mathrm{IND-CCA2}}(\mathcal{A}) = |2\Pr[\delta^* = \delta] - 1|$$
$$P_1 = |\Pr[\delta^* = \delta] - \frac{1}{2}|$$
$$P_1 = \Pr[\delta^* = \delta|$$
$$\sigma_p = (m_b, ID_s, pk_s, ID_r, Pk_r S^*, C^*, U^*)]$$
$$= \frac{\varepsilon + 1}{2} - \frac{q_{sc}(q_{sc} + q_{H_2})}{2^\lambda}$$

and $P_0 = \Pr[\delta^* = i | h \in \mathbb{G}_2] = \frac{1}{2}$ for $i = 0, 1$.

Thus, we have

$$\mathrm{Adv}(C) =|\ P_{a,b,c,\in\mathbb{Z}_p^*,\theta\in\mathbb{G}_2}[1 \leftarrow C(P, aP, bP, cP, \theta)]$$
$$-P_{a,b,c,\in\mathbb{Z}_p^*}[1 \leftarrow C(P, aP, bP, cP, \hat{e}(P,P)^{abc})]\ |$$
$$= \frac{|P_1 - P_0|}{(2^{qH1})^2},$$
$$\epsilon_{gbdh} \geq (\frac{\epsilon}{q_{H_1}})(1 - \frac{q_s(q_{H_2} + q_{H_3} + q_{H_4})}{2^\lambda})(1 - \frac{q_u}{2^\lambda}).$$

5.2. Unforgeability

Theorem 2. *The HOOPSC scheme fulfills EUF-CMA security under the CDHP against the $\mathcal{F}_1$ and $\mathcal{F}_2$ adversaries.*

Proof. The *EUF-CMA-I* and *EUF-CMA-II* games described below demonstrate the security of Theorem 2.

EUF-CMA-I: *In the random oracle model, if an adversary $\mathcal{F}_1$ has a nonnegligible advantage ε in compromising the EUF-CMA-I security of the HOOPSC scheme within a time frame t and performing q_{ppk} inquiries, q_{sk} inquiries, q_{pk} inquiries, q_{pkr} inquiries, q_{ke} inquiries, q_{pd} inquiries, q_{kp} inquiries, q_{sc} inquiries, and q_{H_i} inquiries to oracles H_i $(i = 1,2,3,4)$, then there is a C that can resolve the CDHP with an advantage*

$$\varepsilon_{cdh} \geq \frac{10(q_{sc}+1)(q_{sc}+q_{H_3})q_{H_1}}{(2^\lambda - 1)}$$

In a time

$$t' \leq 120686 q_{H_1} q_{H_3} \frac{t + O((q_{prk} + q_{sc} + q_{re} q_{H_2} + q_{dsc} q_{H_2}) t_p)}{\varepsilon(1 - \frac{1}{2^\lambda})}$$

where t_p represents time for a single pairing operation.

Proof. It is illustrated how C uses $\mathcal{F}_1$ as a subroutine to resolve a given scenario (P, aP, bP) of the *CDHP*.

Initial: C performs the *setup* with λ and sends *params* with $P_{pub} = sP$ to $\mathcal{F}_1$. Note that C is unaware of s. In this game, the UKG secret key is s.

Attack: According to the *IND-CCA2* proof, C responds to $\mathcal{F}_1$ inquiries, except for H_3 inquiries. When $\mathcal{F}_1$ queries H_3 on (v_i). First, C verifies whether L_3 has a tuple (v_i, h_i). If a tuple is found, C yields h_i to $\mathcal{F}_1$. Otherwise, C selects a random $h_i \in \{0,1\}^n$, adds it to L_3, and returns it to $\mathcal{F}_1$.

Forgery: $\mathcal{F}_1$ outputs a triple $(ID_A, ID_s, ID_r, \sigma^*)$ where $\sigma^* = (m_\omega, S^*, C^*, U^*)$. For an identityless chosen message attack, generic forged message (ID_s, m) are utilized. $\mathcal{F}_1'$ generates $((ID_s, m), r, S)$ and $((ID_s, m), r^*, S^*)$ utilizing the forking lemma, maintaining the same commitment but with distinct random values r and r^*. Machine C addresses the *CDH* problem by employing $\mathcal{F}_1'$.

1. Through the execution of $\mathcal{F}_1'$, C generates $(ID_s, m), r, S$ and (ID_s, m).
2. It computes $abP = (r - r^*)^{-1}(S - S^*)$.
3. It then returns abP as the solution to the *CDH* problem.

If $\mathcal{F}_1$ succeeds within time t with a certain probability, based on the forking lemma (Choon and Hee Cheon, 2002), the following is true:

$$\varepsilon_{cdh} \geq \frac{10\left(q_{sc}+1\right)\left(q_{sc}+q_{H_3}\right)q_{H_1}}{(2^\lambda - 1)}$$

C resolves the *CDH* problem within a specific timeframe.

$$t' \leq 120686 q_{H_1} q_{H_3} \frac{t + O(\left(q_{prk} + q_{sc} + q_{re}q_{H_2} + q_{dsc}q_{H_2}\right) t_p)}{\varepsilon(1 - \frac{1}{2^\lambda})}$$

EUF-CMA-II: *In the random oracle model, if adversary $\mathcal{F}_2$ holds a nonnegligible advantage ε in breaching the EUF-CMA-II security of the HOOPSC scheme within a time frame t and conducting q_{sk} inquiries, q_{pk} inquiries, q_{ke} inquiries, q_{pd} inquiries, q_{kp} inquiries, q_{sc} inquiries, and q_{H_i} inquiries to oracles H_i $(i = 1,2,3,4)$, then there is a C that can solve the CDHP with an advantage.*

$$\varepsilon_{cdh} \geq \frac{10(q_{sc}+1)(q_{sc}+q_{H_3})q_{H_1}}{(2^\lambda - 1)}$$

In a time

$$t' \leq 120686 q_{H_1} q_{H_3} \frac{t + O((q_{prk} + q_{sc} + q_{re}q_{H_2} + q_{dsc}q_{H_2})t_p)}{\varepsilon(1 - \frac{1}{2^\lambda})}$$

where t_p represents one pairing operation time.

Proof. It is illustrated how C uses $\mathcal{F}_2$ as a subroutine to resolve a given scenario (P, aP, bP) of the *CDHP*.

Initial: C performs the *setup* using λ and sends *params* with $P_{pub} = sP$ to $\mathcal{F}_2$. Here, C randomly selects s.

Attack: C mimics $\mathcal{F}_2$ in the *EUF-CMA-II* game. C maintains four lists L_i (where i ranges from 1 to 4) to simulate the hash oracles H_1, H_2, H_3 and H_4, respectively. It keeps private and public keys in L_k, L_{pk} for the proxy key, and L_{sc} for the signcrypt. It is assumed that the inquiries in H_1 are distinct and that $\mathcal{F}_2$ requests the queries in $H_1(ID_i)$ prior to the identity ID_i being utilized in the remaining queries. Furthermore, by employing the irreflexivity assumption (Boyen, 2003), it assumed that the identities of the sender and recipient are distinct. C picks a random $\lambda \in \{1, 2, \ldots, q_s + q_p + q_{pd} + q_{kp} + q_{sc}\}$. C answers H_2, H_3, H_4, *proxy delegation*, *proxy key*, and *signcrypt* inquiries by applying the same procedure as *Theorem 1* queries. The details of the other inquiries are as follows.

H_1 query: When $\mathcal{F}_2$ queries H_1 for ID_i, C first checks whether L_1 contains a pair of (ID_i, k_i, y_i). If a pair is identified, C returns y_iP to $\mathcal{F}_2$. Otherwise, C selects a random $e \in \mathbb{Z}_q^*$, inserts (ID_i, e) into L_1, and returns y_ie to $\mathcal{F}_2$.

Private key inquiries: When $\mathcal{F}_2$ asks for a *private key inquiry* on an identity ID_i, if $ID_i = ID_r$, C fails. Otherwise, C runs the H_1 oracle to obtain (ID_i, k_i, y_i). Then, C checks L_k for entry (ID_r, pk_i, x_i).

Forgery: $\mathcal{F}_2$ outputs a triple $(ID_A, ID_s, ID_r, \sigma^*)$ where $\sigma^* = (m_\omega, S^*, c^*, U^*)$. For an identityless chosen message attack, generic forged message (ID_s, m) are utilized. $\mathcal{F}_2'$ generates $((ID_s, m), r, S)$ and $((ID_s, m), r^*, S^*)$ by using the forking lemma, maintaining the same commitment but with distinct random values r and r^*. Machine C tackles the *CDH* problem by employing $\mathcal{F}_2'$.

1. By executing $\mathcal{F}_2'$, C generates $(ID_s, m), r, S$ and (ID_s, m).
2. It computes $abP = (r - r^*)^{-1}(S - S^*)$.
3. It then returns abP as the solution to the *CDH* problem.

If $\mathcal{F}_2$ succeeds within time t with a certain probability, based on the forking lemma (Choon and Hee Cheon, 2002), the following is true:

$$\varepsilon_{cdh} \geq \frac{10\left(q_{sc}+1\right)\left(q_{sc}+q_{H_3}\right)q_{H_1}}{(2^\lambda - 1)}$$

C resolves the *CDH* problem within a specific timeframe.

$$t' \leq 120686 q_{H_1} q_{H_3} \frac{t + O(\left(q_{prk} + q_{sc} + q_{re}q_{H_2} + q_{dsc}q_{H_2}\right) t_p)}{\varepsilon(1 - \frac{1}{2^\lambda})}$$

6. Performance

In this section, the major computational cost, communication overhead, security and environment of the proposed scheme are evaluated in comparison with those of existing schemes (Lo et al., 2014, Yu et al., 2018, Hundera et al., 2020, and Qu and Zeng, 2022), as presented in Tables 2 and 3. Table 2 outlines the operation P as the pairing in $\mathbb{G}_2$, M represents scalar multiplication in $\mathbb{G}_1$, and E signifies exponentiation in $\mathbb{G}_2$. Table 2 does not include other operations because these three operations consume the longest running time for the entire algorithm (Cui et al., 2007). In the security column, ✓denotes the fulfillment of a security property, and × indicates its absence. For the key size column, the combined sizes of the public, secret and proxy keys were considered. Here, $|x|$ indicates the number of bits in x.

Table 2 shows that HOOPSC has the lowest computational cost and divides signcryption (SC) into offline and online stages. Two-point multiplication was precalculated offline. The online phase is highly efficient and requires only one multiplication. That is, HOOPSC can perform the entire signcryption process more quickly than the existing schemes when a message is available. Moreover, Fig. 4 further demonstrates the efficiency of HOOPSC compared to the others. It provides a clear visual representation of HOOPSC has performance advantages, highlighting its effectiveness in a comparative analysis. This comparison clearly shows HOOPSC capabilities in terms of efficiency and effectiveness.

Regarding security, Lo et al. (2014), Yu et al. (2018), and Hundera et al. (2020) satisfied both the IND-CCA2 and EUF-CMA security properties for IBC environments, and Hundera et al. (2020) is publicly verifiable. The schemes of Qu and Zeng (2022) and HOOPSC satisfy both the IND-CCA2 and EUF-CMA security properties for CLC environments against Type 1 and II attacks, and established public verifiability; however, Qu and Zeng (2022) incurs higher computational costs and communication overhead than HOOPSC. Therefore, HOOPSC is highly suitable for providing security solutions to UAV networks.

The three schemes proposed by Lo et al. (2014), Yu et al. (2018), and Hundera et al. (2020) belong to the IBC environment, whereas the scheme Qu and Zeng (2022) belongs to the CLC environment. However, in a heterogeneous UAV environment, the sender and receiver must be in different cryptosystems. Therefore, a scheme functioning within the same cryptosystem is impractical for use in such environments.

In Table 3, the communication cost of HOOPSC is compared with the schemes of Lo et al. (2014), Yu et al. (2018), Hundera et al.

Table 2
Comparison of computational cost and security.

Scheme	Computational cost		Security			Environment
	SC	DSC	IND-CCA2	EUF-CMA	Public verifiability	
Lo et al. (2014)	4M + P	5M + 3P	✓	✓	×	IBC
Yu et al. (2018)	4M + P + E	2M + 4P + E	✓	✓	×	IBC
Hundera et al. (2020)	2M + 2P + 2E	2M + 6P + 2E	✓	✓	✓	IBC
Qu and Zeng (2022)	7M	6M + 3P	✓	✓	✓	CLC
HOOPSC	2M(Off) + 1M(On)	7M	✓	✓	✓	CLC-IBC

Table 3
Comparison of communication cost.

Schemes	Key size	Delegation size	Ciphertext size	Offline storage
Lo et al. (2014)	$\lvert\mathbb{Z}_q^*\rvert + \lvert\mathbb{G}_1\rvert$	$\lvert\mathbb{Z}_q^*\rvert + \lvert\mathbb{G}_1\rvert + \lvert m_\omega\rvert$	$4\lvert\mathbb{G}_1\rvert + \lvert m\rvert + \lvert m_\omega\rvert$	0
Yu et al. (2018)	$2\lvert\mathbb{G}_1\rvert$	$2\lvert\mathbb{G}_1\rvert + \lvert m_\omega\rvert$	$4\lvert\mathbb{G}_1\rvert + \lvert m\rvert + \lvert m_\omega\rvert$	0
Hundera et al. (2020)	$2\lvert\mathbb{G}_1\rvert$	$3\lvert\mathbb{G}_1\rvert + \lvert m_\omega\rvert$	$\lvert\mathbb{Z}_q^*\rvert + 2\lvert\mathbb{G}_1\rvert + \lvert m\rvert + \lvert m_\omega\rvert$	0
Qu and Zeng (2022)	$\lvert\mathbb{Z}_q^*\rvert + 6\lvert\mathbb{G}_1\rvert$	$4\lvert\mathbb{G}_1\rvert + \lvert m_\omega\rvert$	$6\lvert\mathbb{G}_1\rvert + \lvert m\rvert + \lvert m_\omega\rvert$	0
HOOPSC	$\lvert\mathbb{Z}_q^*\rvert + 3\lvert\mathbb{G}_1\rvert$	$3\lvert\mathbb{G}_1\rvert + \lvert m_\omega\rvert$	$2\lvert\mathbb{G}_1\rvert + \lvert m\rvert + \lvert m_\omega\rvert$	$2\lvert\mathbb{G}_1\rvert$

Table 4
The comparative overview of security levels (bits).

Security level	Size of P	Size of q
80-bit	512	160
112-bit	1024	224
128-bit	1536	256

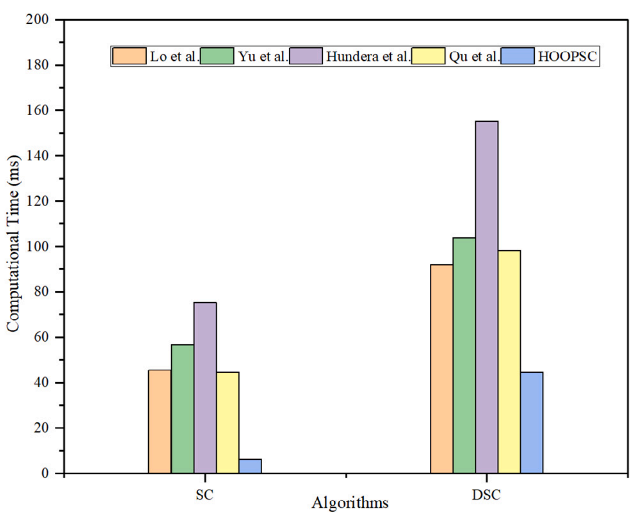

Fig. 4. Comparison of computational cost.

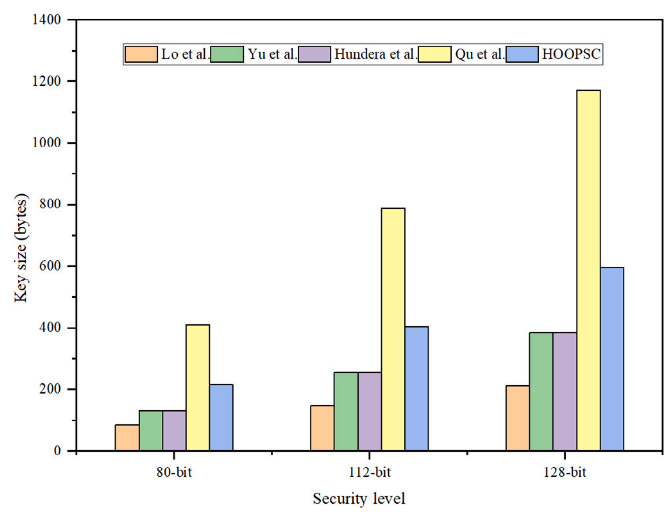

Fig. 5. The key size of the schemes.

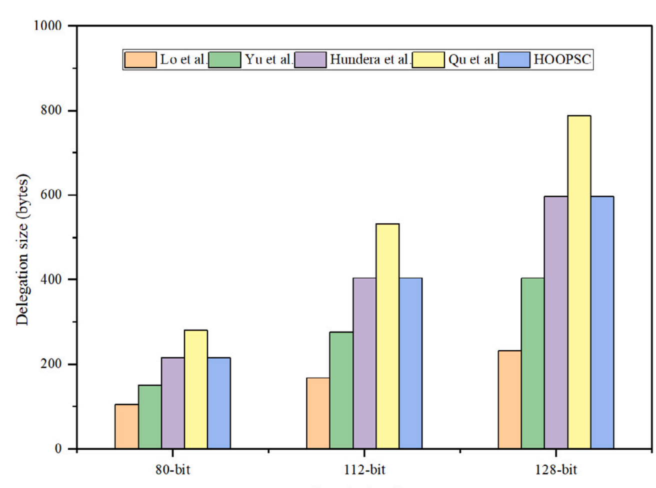

Fig. 6. The delegation size of the schemes.

(2020) and Qu and Zeng (2022) according to the size of the keys, delegation, ciphertext size and offline storage. The experiment was conducted using Type A pairing with the PBC library (Lynn, 2007), running on a desktop ONDA B760-VH4 Gen 13 instrument equipped with an Intel® Core™ i5-13600KF 3.50 GHz processor, 24-GB GPU (NVIDIA GeForce RTX 3090) and 64-GB RAM. Type A pairings are built on the curve $y^2 = (x^3 + x) \bmod p$ for some prime $p = 3 \bmod 4$, where the order of $\mathbb{G}_1$ is q and the embedding degree is 2. Here, three types of parameters corresponding to the security levels defined by 80-bit, 112-bit and 128-bit AES key sizes, as described previously (Islam and Biswas, 2017), were considered. A comparative overview of the security levels is presented in Table 4. According to Cao et al. (2010), the average execution time for a scalar multiplication operation in $\mathbb{G}_1$ is approximately 6.38 ms, the exponentiation computation in $\mathbb{G}_2$ is approximately 11.20 ms, and a pairing operation requires approximately 20.01 ms. For comparisons of computational costs, it is assumed that the size of a message and the size of $\lvert m_\omega\rvert$ are 160 bits each. When an 80-bit security level is used, p is 512 bits in size. As a result, by utilizing an elliptic curve with 160 bits q size, the size of an element in group $\mathbb{G}_1$ is 1024 bits. However, this can be reduced to 65 bytes by using standard

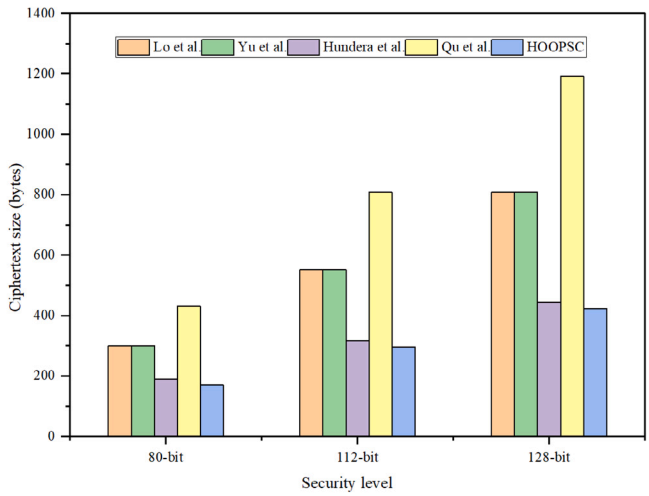

Fig. 7. The ciphertext size of the schemes.

compression techniques (Shim, 2012). The elements in $\mathbb{G}_2$ were 1024 bits.

Therefore, the key sizes of Lo et al. (2014), Yu et al. (2018), Hundera et al. (2020), Qu and Zeng (2022) and the proposed scheme are $|\mathbb{Z}_q^*| + |\mathbb{G}_1| = 20 + 65 = 85$ bytes, $2|\mathbb{G}_1| = 2 \times 65 = 130$ bytes, $2|\mathbb{G}_1| = 2 \times 65 = 130$ bytes, $|\mathbb{Z}_q^*| + 6|\mathbb{G}_1| = 20 + 6 \times 65 = 410$ bytes, and $|\mathbb{Z}_q^*| + 3|\mathbb{G}_1| = 20 + 3 \times 65 = 215$ bytes, respectively. The delegation sizes of Lo et al. (2014), Yu et al. (2018), Hundera et al. (2020), Qu and Zeng (2022) and the proposed scheme are $|\mathbb{Z}_q^*| + |\mathbb{G}_1| + |m_\omega| = 105$ bytes, $2|\mathbb{G}_1| + |m_\omega| = 2\times65+20 = 150$ bytes, $3|\mathbb{G}_1| + |m_\omega| = 3\times65+20 = 215$ bytes, $4|\mathbb{G}_1| + |m_\omega| = 4\times65+20 = 280$ bytes and $3|\mathbb{G}_1| + |m_\omega| = 3\times65+20 = 215$ bytes, respectively. The ciphertext sizes used by Lo et al. (2014), Yu et al. (2018), Hundera et al. (2020), Qu and Zeng (2022) and the proposed scheme are $4|\mathbb{G}_1| + |m| + |m_\omega| = 4 \times 65 + 20 + 20 = 300$ bytes, $4|\mathbb{G}_1| + |m| + |m_\omega| = 4\times65+20+20 = 300$ bytes, $|\mathbb{Z}_q^*| + 2|\mathbb{G}_1| + |m| + |m_\omega| = 20 + 2 \times 65 + 20 + 20 = 190$ bytes, $6|\mathbb{G}_1| + |m| + |m_\omega| = 6 \times 65 + 20 + 20 = 430$ bytes and $2|\mathbb{G}_1| + |m| + |m_\omega| = 2 \times 54 + 20 + 20 = 170$ bytes, respectively. Offline storage of $2|\mathbb{G}_1| = 2 \times 65 = 130$ bytes is required for our scheme. The computational costs for the 112−bit and 128−bit security levels can be determined using the same technique. Figs. 5, 6, and 7 show the key, delegation and ciphertext sizes, respectively, at different security levels. As depicted in Figs. 7, the proposed scheme has a smaller ciphertext size than the existing schemes. According to Figs. 5 and 6, HOOPSC has a larger key size than the schemes (Lo et al., 2014; Yu et al., 2018; Hundera et al., 2020) and a lower key size than the scheme (Qu and Zeng, 2022). Additionally, the proposed scheme shares a similar delegation size to Hundera et al. (2020) and has a greater delegation size than Lo et al. (2014) and Yu et al. (2018), whereas Qu and Zeng (2022) exhibits the largest delegation size of all. However, all schemes Lo et al. (2014), Yu et al. (2018), Hundera et al. (2020) and Qu and Zeng (2022) operate in homogeneous cryptosystems and cannot be effectively used in a practical heterogeneous UAV environment.

7. Conclusion

This paper presents a novel and efficient HOOPSC scheme for secure communication in UAV networks. Using online and offline signcryption techniques, the computational burden on both the GCS and the UAV is significantly reduced. Moreover, the proposed scheme established a secure channel between the CC, GCS and UAV, enabling end-to-end confidentiality, integrity, authentication and nonrepudiation. The security of the scheme is proven in terms of indistinguishability against adaptive chosen ciphertext attacks (IND-CCA2) and existential unforgeability against adaptive chosen message attacks (EUF-CMA) under the decisional bilinear Diffie–Hellman (DBDH) and computational Diffie–Hellman (CDH) problems in the random oracle model. An experimental analysis demonstrates that HOOPSC surpasses the existing schemes in terms of computational cost and communication overhead. Therefore, the HOOPSC scheme is highly suitable for long-range operations in UAV networks. Future work will focus on integrating HOOPSC with 5G and AI to enhance its performance and energy efficiency.

CRediT authorship contribution statement

Negalign Wake Hundera: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Wang Shumeng:** Conceptualization, Data curation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Dagmawit Mesfin:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Huiying Xu:** Conceptualization, Methodology, Validation, Visualization, Writing – review & editing. **Xinzhong Zhu:** Conceptualization, Funding acquisition, Methodology, Validation, Visualization, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

An, J.H., Dodis, Y., Rabin, T., 2002. On the security of joint signature and encryption. In: Advances in Cryptology — EUROCRYPT 2002, vol. 2332, pp. 83–107.

Baek, J., Steinfeld, R., Zheng, Y., 2007. Formal proofs for the security of signcryption. J. Cryptol. 20 (2), 203–235.

Barbosa, M., Farshim, P., 2008. Certificateless signcryption. In: Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, vol. 3788, pp. 369–372.

Boneh, D., Franklin, M., 2001. Identity-based encryption from the Weil pairing. In: Advances in Cryptology — CRYPTO 2001. 2139, pp. 213–229.

Boyen, X., 2003. Multipurpose identity-based signcryption: A Swiss army knife for identity-based cryptography. In: Annual International Cryptology Conference, vol. 2729, pp. 383–399.

Cao, X., Kou, W., Du, X., 2010. A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges. Inform. Sci. 180 (15), 2895–2903.

Chen, J., Wang, L., Wen, M., Zhang, K., Chen, K., 2021. Efficient certificateless online/offline signcryption scheme for edge IoT devices. IEEE Internet Things J. 9 (11), 8967–8979.

Cho, K.Y., Lee, D.H., 2007. Certificateless proxy signature scheme. J. korea Multimedia Soc..

Choon, J.C., Hee Cheon, J., 2002. An identity-based signature from gap Diffie-Hellman groups. In: Public Key Cryptography — PKC 2003, vol. 2567, pp. 18–30.

Cui, S., Duan, P., Chan, C.W., Cheng, X., 2007. An efficient identity-based signature scheme and its applications. Int. J. Netw. Secur. 5 (1), 89–98.

Deng, L., Zeng, J., Huang, H., 2016. Efficient certificateless proxy signature scheme. Internat. J. Found. Comput. Sci. 27 (01), 85–100.

Faiçal, B.S., Costa, F.G., Pessin, G., Ueyama, J., Freitas, H., Colombo, A., Fini, P.H., Villas, L., Osório, F.S., Vargas, P.A., Braun, T., 2014. The use of unmanned aerial vehicles and wireless sensor networks for spraying pesticides. J. Syst. Archit. 60 (4), 393–404.

Gamage, C., Leiwo, J., Zheng, Y., 1999. An efficient scheme for secure message transmission using proxy-signcryption. In: Computer Science Proceedings of the 22nd Australasian Computer Science Conference. pp. 18–21.

Ge, C., Ma, X., Liu, Z., 2020. A semi-autonomous distributed blockchain-based framework for UAVs system. J. Syst. Archit. 107, 101728.

He, D., Chan, S., Guizani, M., 2016. Communication security of unmanned aerial vehicles. IEEE Wirel. Commun. 24 (4), 134–139.

Hua, M., Wu, Q., Yang, L., Schober, R., Poor, H.V., 2021. A novel wireless communication paradigm for intelligent reflecting surface based symbiotic radio systems. IEEE Trans. Signal Process. 70, 550–565.

Hundera, N.W., Jin, C., Geressu, D.M., Aftab, M.U., Olanrewaju, O.A., Xiong, H., 2022. Proxy-based public-key cryptosystem for secure and efficient IoT-based cloud data sharing in the smart city. Multimedia Tools Appl. 81 (21), 29673–29697.

Hundera, N.W., Mei, Q., Xiong, H., Geressu, D.M., 2020. A secure and efficient identity-based proxy signcryption in cloud data sharing. KSII Trans. Internet Inf. Syst. 14 (1), 455–472.

Islam, S.H., Biswas, G., 2017. A pairing-free identity-based two-party authenticated key agreement protocol for secure and efficient communication. J. King Saud Univ.-Comput. Inf. Sci. 29 (1), 63–73.

Javed, A.R., Shahzad, F., ur Rehman, S., Zikria, Y.B., Razzak, I., Jalil, Z., Xu, G., 2022. Future smart cities: Requirements, emerging technologies, applications, challenges, and future aspects. Cities 129, 103794.

Khan, A.A., Laghari, A.A., Awan, S.A., 2021a. Machine learning in computer vision: A review. EAI Endorsed Trans. Scalable Inf. Syst. 8 (32), e4.

Khan, A.A., Laghari, A.A., Li, P., Dootio, M.A., Karim, S., 2023. The collaborative role of blockchain, artificial intelligence, and industrial internet of things in digitalization of small and medium-size enterprises. Sci. Rep. 13 (1), 1656.

Khan, A.A., Laghari, A.A., Shaikh, Z.A., Dacko-Pikiewicz, Z., Kot, S., 2022a. Internet of things (IoT) security with blockchain technology: A state-of-the-art review. IEEE Access 10, 122679–122695.

Khan, A.A., Shaikh, Z.A., Baitenova, L., Mutaliyeva, L., Moiseev, N., Mikhaylov, A., Laghari, A.A., Idris, S.A., Alshazly, H., 2021b. QoS-ledger: Smart contracts and metaheuristic for secure quality-of-service and cost-efficient scheduling of medical-data processing. Electronics 10 (24), 3083.

Khan, A.A., Shaikh, A.A., Shaikh, Z.A., Laghari, A.A., Karim, S., 2022b. IPM-model: AI and metaheuristic-enabled face recognition using image partial matching for multimedia forensics investigation with genetic algorithm. Multimedia Tools Appl. 81 (17), 23533–23549.

Li, X.-x., Chen, K.-f., Sun, L., 2005. Certificateless signature and proxy signature schemes from bilinear pairings. Lith. Math. J. 45 (1), 76–83.

Li, F., Han, Y., Jin, C., 2016. Practical access control for sensor networks in the context of the internet of things. Comput. Commun. 89, 154–164.

Li, F., Han, Y., Jin, C., 2017a. Certificateless online/offline signcryption for the internet of things. Wirel. Netw. 23 (1), 145–158.

Li, F., Liu, B., Hong, J., 2017b. An efficient signcryption for data access control in cloud computing. Computing 99 (5), 465–479.

Li, F., Shirase, M., Takagi, T., 2013. Certificateless hybrid signcryption. Math. Comput. Modelling 57 (3–4), 324–343.

Li, W., Xia, C., Wang, C., Wang, T., 2022. Secure and temporary access delegation with equality test for cloud-assisted IoV. IEEE Trans. Intell. Transp. Syst. 23 (11), 20187–20201.

Li, F., Xiong, P., 2013. Practical secure communication for integrating wireless sensor networks into the internet of things. IEEE Sens. J. 13 (10), 3677–3684.

Lo, N.-W., Tsai, J.-L., et al., 2014. A provably secure proxy signcryption scheme using bilinear pairings. J. Appl. Math. 2014, 10.

Lu, R., He, D., Wang, C., 2007. Cryptanalysis and improvement of a certificateless proxy signature scheme from bilinear pairings. In: Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, Vol. 3. SNPD 2007, IEEE, pp. 285–290.

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

Lynn, B., 2007. Pbc library-pairing-based cryptography. http://crypto.stanford.edu/pbc/.

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

Mandal, S., Bera, B., Sutrala, A.K., Das, A.K., Choo, K.-K.R., Park, Y., 2020. Certificateless-signcryption-based three-factor user access control scheme for IoT environment. IEEE Internet Things J. 7 (4), 3184–3197.

MING, Y., 2014. Secure identity-based proxy signcryption scheme in standard model. J. Comput. Appl. 34 (10), 2834.

Ming, Y., Wang, Y., 2015. Proxy signcryption scheme in the standard model. Secur. Commun. Netw. 8 (8), 1431–1446.

Mohsan, S.A.H., Othman, N.Q.H., Li, Y., Alsharif, M.H., Khan, M.A., 2023. Unmanned aerial vehicles (UAVs): Practical aspects, applications, open challenges, security issues, and future trends. Intell. Serv. Robot. 16 (1), 109–137.

Niu, S., Shao, H., Su, Y., Wang, C., 2023. Efficient heterogeneous signcryption scheme based on edge computing for industrial internet of things. J. Syst. Archit. 136, 102836.

Pan, X., Jin, Y., Wang, Z., Li, F., 2022. A pairing-free heterogeneous signcryption scheme for unmanned aerial vehicles. IEEE Internet Things J. 9 (19), 19426–19437.

Qi, F., Zhu, X., Mang, G., Kadoch, M., Li, W., 2019. UAV network and IoT in the sky for future smart cities. IEEE Network 33 (2), 96–101.

Qu, Y., Zeng, J., 2022. Certificateless proxy signcryption in the standard model for a UAV network. IEEE Internet Things J. 9 (16), 15116–15127.

Saraswat, V., Sahu, R.A., Awasthi, A.K., 2017. A secure anonymous proxy signcryption scheme. J. Math. Cryptol. 11 (2), 63–84.

Shamir, A., 1985. Identity-based cryptosystems and signature schemes. In: Advances in Cryptology: Proceedings of CRYPTO 84 4, vol. 196, pp. 47–53.

Shim, K.-A., 2012. An efficient conditional privacy-preserving authentication scheme for vehicular sensor networks. IEEE Trans. Veh. Technol. 61 (4), 1874–1883.

Shin, Y.A., Jeong, I.R., Byun, J.W., 2023. Identity-based multi-proxy signature with proxy signing key for internet-of-drones. IEEE Internet Things J. 11 (3), 4191–4205.

Spies, T., 2017. Public key infrastructure. In: Computer and Information Security Handbook, third ed. pp. 691–711.

Waheed, A., Umar, A.I., Zareei, M., Din, N., Amin, N.U., Iqbal, J., Saeed, Y., Mohamed, E.M., 2020. Cryptanalysis and improvement of a proxy signcryption scheme in the standard computational model. IEEE Access 8, 131188–131201.

Xu, G., Dong, J., Ma, C., Liu, J., Cliff, U.G.O., 2022. A certificateless signcryption mechanism based on blockchain for edge computing. IEEE Internet Things J. 10 (14), 11960–11974.

Yanfeng, Q., Chunming, T., Yu, L., Maozhi, X., Baoan, G., 2013. Certificateless proxy identity-based signcryption scheme without bilinear pairings. China Commun. 10 (11), 37–41.

Yang, W., Weng, J., Huang, X., Yang, A., 2020. A provably secure certificateless proxy signature scheme against malicious-but-passive KGC attacks. Comput. J. 63 (8), 1139–1147.

Yu, H., Wang, Z., 2019. Construction of certificateless proxy signcryption scheme from CMGs. IEEE Access 7, 141910–141919.

Yu, H., Wang, Z., Li, J., Gao, X., 2018. Identity-based proxy signcryption protocol with universal composability. Secur. Commun. Netw. 2018, 1–11.

Yu, X., Zhao, W., Tang, D., 2022. Efficient and provably secure multi-receiver signcryption scheme using implicit certificate in edge computing. J. Syst. Archit. 126, 102457.

Zhang, Q., Jiang, M., Feng, Z., Li, W., Zhang, W., Pan, M., 2019. IoT enabled UAV: Network architecture and routing algorithm. IEEE Internet Things J. 6 (2), 3727–3742.

Zheng, Y., 1997. Digital signcryption or how to achieve cost (signature & encryption)<< cost (signature)+ cost (encryption). In: Advances in Cryptology — CRYPTO '97". pp. 165–179.

Zhou, C.-X., 2016. Identity based generalized proxy signcryption scheme. Inf. Technol. Control 45 (1), 13–26.

Zhou, W., Fan, L., Zhou, F., Li, F., Lei, X., Xu, W., Nallanathan, A., 2023. Priority-aware resource scheduling for UAV-mounted mobile edge computing networks. IEEE Trans. Veh. Technol. 72 (7), 9682–9687.

Zhou, Y., Li, Z., Hu, F., Li, F., 2019. Identity-based combined public key schemes for signature, encryption, and signcryption. In: Information Technology and Applied Mathematics, vol. 699, pp. 3–22.

Zhou, C., Zhang, Y., Wang, L., 2018. A provable secure identity-based generalized proxy signcryption scheme. Int. J. Netw. Secur. 20 (6), 1183–1193.