# Blockchain-Based Mutual-Healing Group Key Distribution Scheme in Unmanned Aerial Vehicles Ad-Hoc Network

Xinghua Li, *Member, IEEE*, Yunwei Wang, Pandi Vijayakumar, *Member, IEEE*, Debiao He, *Member, IEEE*, Neeraj Kumar, *Senior Member, IEEE*, and Jianfeng Ma, *Member, IEEE*

***Abstract*—A dynamic group key is required for secure communication in the Unmanned Aerial Vehicles Ad-Hoc Network (UAANET). However, due to the unreliable wireless channel and high-dynamic topology of UAANET, the situation that a node is missing certain group key broadcast messages occurs frequently. Existing group key distribution schemes cannot be directly applied to the UAANET, because of their poor security or real-time. Therefore, we present a mutual-healing group key distribution scheme based on the blockchain. Firstly, the Ground Control Station (GCS) builds a private blockchain where the group keys distributed by GCS are recorded. Meanwhile, through the blockchain, a dynamic list of UAANET membership certificates is also managed. According to different attack models, a basic mutual-healing protocol and an enhanced one are designed based on the Longest-Lost-Chain mechanism to recover the node's lost group keys with the aid of its neighbors. Security analysis and extensive experiments show that, compared with the existing mutual-healing schemes, our proposed solution can effectively resist various attacks with small overhead on time and storage.**

***Index Terms*—Unmanned Aerial Vehicles Ad-Hoc Network (UAANET), group key distribution, mutual-healing, blockchain.**

## I. Introduction

UNMANNED Aerial Vehicle (UAV) has been widely used in military and civil fields [1] with its low cost, good maneuverability, and convenient operation. A recent report[1] indicates that the global UAV market will reach 300 billion US dollar in 2036, showing explosive growth. Meanwhile, in many applications, it has become a trend that UAVs form a network, called Unmanned Aerial Vehicles Ad-Hoc Network (UAANET), under the control of Ground Control Station (GCS) to accomplish tasks by group communications [2], [3], e.g., traffic monitoring [4], disaster management [5] and relay communications [6].

However, group communications in UAANET are vulnerable to passive and active attacks such as eavesdropping and deception. Therefore, a group key is usually established within the UAANET where all group communications are encrypted with it. Meanwhile, the key needs to be updated dynamically when membership changes or after a fixed period. Generally, the above process is implemented by GCS to distribute the group key broadcast message through broadcast channels. Compared with traditional wireless networks, UAANET has the unique features of high-speed node movement, high-dynamic topology change, complex external environment and so on [7]. Those will cause intermittent interruption of communication links between UAVs and further lead to frequent loss of group key broadcast messages, which has been pointed out in [8] and our extensive experiments also verify this point (as shown in Fig. 7). If the UAV fails to recover the lost group keys in time, it will be unable to participate in the task cooperation normally. This poses a serious challenge to group key distribution in UAANET.

To address the aforementioned challenges, a naive approach is for each UAV to directly contact the GCS to request a retransmission of the updated keys. Due to frequent change in the highly dynamic UAV network topology, it is often necessary to establish dynamic routing with GCS, which increases the recovery latency (as shown in Fig. 8) and the overhead of GCS. Therefore, existing works often adopt the following two methods. One is the self-healing group key distribution [9]–[23], where nodes missing multiple consecutive broadcasts can still recover the lost group keys by using the broadcast messages it has received at a previous session and the packets it will receive in the future, without requesting additional transmission from the group manager. Nevertheless, in such schemes, nodes have to wait for the next broadcast passively, which results in lower real-time (as shown in Fig. 9). The other method is called mutual-healing group key distribution [24]–[27], where nodes can actively request the lost group keys from their neighbors. Although it improves the real-time responsiveness, the existing schemes are

Manuscript received June 2, 2019; revised August 1, 2019; accepted September 5, 2019. Date of publication September 23, 2019; date of current version November 12, 2019. This work was supported by National Natural Science Foundation of China under Grants U1708262, U1736203, 61772173, and 61672413. The review of this article was coordinated by Prof. Liehuang Zhu. *(Corresponding author: Yunwei Wang.)*

X. Li, Y. Wang, and J. Ma are with the School of Cyber Engineering, Xidian University, Xian, Shaanxi, China (e-mail: xhli1@mail.xidian.edu.cn; wywxidian@foxmail.com; jfma@mail.xidian.edu.cn).

P. Vijayakumar is with the Department of Computer Science and Engineering, University College of Engineering Tindivanam, Tindivanam, Tamil Nadu, India (e-mail: vijibond2000@gmail.com).

D. He is with the State Key Lab of Software Engineering, Computer School, Wuhan University, Wuhan, China (e-mail: hedebiao@163.com).

N. Kumar is with the Computer Science Engineering Department, Thapar Institute of Engineering and Technology, Patiala, Punjab, India (e-mail: neeraj.kumar@thapar.edu).

Digital Object Identifier 10.1109/TVT.2019.2943118

[1]http://www.avascent.com/2018/02/think-bigger-large-unmanned-systems-and-the-next-major-shift-in-aviation/

mainly applied to stable wireless sensor networks where the group members and their relative positions are basically fixed. In the mutual-healing process, adjacent nodes establish the trust relationship based on the fixed neighboring location relations and historical group keys. In other words, the mutual-healing method relies on stable network topology to establish trust relationships between neighbors in the network. However, this is not the case in UAANET, the topology and group membership change frequently, and as a result, previous approaches would not be able to establish a reliable trust relationship between UAVs. Without trust, the mutual-healing can no longer proceed securely.

Consequently, the existing schemes fail to recover the lost group keys securely and timely, if they are applied to UAANET directly. To tackle the challenge of group key distribution in highly dynamic network topology, we propose to design a distributed, multi-party trusted database in UAANET. This database provides two key capabilities. First, it is used to store the group key broadcasts so the UAVs can request the lost group keys directly from this distributed database. Second, it can be used to build a reliable dynamic trust relationship in UAANET, which is the foundation for securing interactive messages in the mutual-healing process.

Blockchain, an emerging technology which realizes distributed, tamper-resistant data storage and has been widely used in various scenarios, such as cloud environment [28], smart grid networks [29], [30], can meet the above requirements well. Therefore, we take advantage of the blockchain technology to design a mutual-healing group key distribution scheme in UAANET, and the main contributions are summarized as follows:

1) By redefining the block structure, a private blockchain constructed by GCS is designed to distribute and store group key broadcast messages. And the UAANET member's joining and revoking are also treated as transactions to be recorded in the blockchain.
2) Based on the constructed blockchain, a basic mutual-healing protocol and an enhanced one are designed according to different attack models, through which UAVs can recover the lost group keys securely and timely.
3) Security analysis and extensive experiments show that, compared with the existing mutual-healing schemes, the proposed scheme can effectively resist various attacks with limited time and storage overhead.

The remainder of this paper is organized as follows. In Section II, we present an overview of existing group key distribution works in unreliable networks. In Section III, we introduce some preliminaries. In Section IV, our scheme is presented in detail. The theory analysis and experiments are given in Sections V and VI, respectively. Finally, we conclude this paper in Section VII.

## II. Related Work

The existing group key distribution schemes in unreliable networks fall into two significant categories. One is the self-healing group key distribution with the help of group manager (referred to as GM), another is the mutual-healing scheme with the help of neighbors.

### A. Self-Healing Group Key Distribution

The concept of self-healing group key distribution was first introduced by Staddon *et al.* [9] Its core idea is that GM adds some redundant messages to the group key broadcasts, so that the group members can recover the lost group keys by receiving any two group key broadcasts which " sandwich " the lost broadcasts. Subsequently, many works on self-healing key distribution were done in terms of entitling special features or improving efficiency.

Among them, Blundo *et al.* [10] pointed out that both constructions of self-healing in Staddon scheme are vulnerable. Specifically, attackers could directly use continuous broadcasts to obtain the group key in the first basic construction, and the newly joined node could recover the past group keys, which it did not belong to, in the second construction. Liu *et al.* [11] gave a trade-off between the broadcasts size and the maximum number of recoverable lost group keys by introducing a sliding window, which reduced the communication and storage overhead of Staddon's. Furthermore, Zou *et al.* [12] proposed a new scheme not only solved the inherent problems of the above schemes [9]–[11], such as the need to pre-set the maximum number of sessions and the inability to support the temporary revocation of nodes, but also reduced the storage overhead to a constant level.

However, all the schemes mentioned above were based on the polynomial secret sharing algorithm, so that their ability to resist collusion attacks was limited by the degree of polynomials, and they were also vulnerable to the polynomial factorization attack [13], [14]. Therefore, researchers successively proposed vector space secret sharing (VSS) based self-healing schemes [15], [16], where VSS made use of general monotone decreasing access structures for the subsets of users that can be revoked, to achieve more flexible performance. But due to the access structure was chosen in advance, the maximum number of revoked nodes was predetermined and constrained. Meanwhile, to further reduce the communication and computation overhead, many schemes based on hash chains had also been proposed [17]–[19]. But unfortunately, they were vulnerable to collusion attacks between the revoked users and the newly joined users. Although the schemes [20], [21] solve this problem by combining revocation polynomial, they also inherit the inherent shortcomings of polynomial-based schemes. What's more, all above hash chain-based schemes need to pre-set the fixed maximum number of sessions in advance which cannot be applied in long-lived system [22]. Efforts never stop, researchers proposed a self-healing key scheme based on bilinear pairing [23], where any number of nodes could be revoked, any collusion of non-authorized nodes could be resisted and also no limitation on the maximum number of sessions. Despite it had solved most of the inherent problems in the previous schemes, its expensive

computation and communication overhead were not suitable for scenarios with resource constraints of nodes.

What's more regrettable is that all the existing self-healing key schemes [9]–[23] mentioned above are passive methods, where the real-time of the node to recover the lost group keys depends on the delay of receiving the next group key broadcast distributed by the GM in future. When the node is unable to recover the lost group key in time, it will result in its failure to send and receive the group messages. This cannot meet the requirements of real-time in UAANET.

### B. Mutual-Healing Group Key Distribution

To address the problem of real-time in self-healing key schemes, researchers further proposed some group key distribution schemes with mutual-healing property [24]–[27]. The primary motivation is that, if a node missed more than a fixed number group key broadcasts, instead of waiting for the next broadcast, it can get assistance from its neighbors to recover that lost keys instantly. Muhammad *et al.* [24] first proposed the concept of mutual-healing, but only discussed its feasibility without exploring technical details. Subsequently, Tian *et al.* [25] proposed a specific mutual-healing scheme based on bilinear pairing in wireless sensor networks where the nodes were stationary. In this solution, the location-based secret needed to be set up for each node by the range-based localization. When a node missed some group key broadcasts can broadcast a request message to neighbors with its identity, location and the serial number of lost group key. Upon receiving such request, a cooperate node firstly verified that the request was indeed from its one-hop neighbors by their location proximity relationship, then generated a shared key based on the location of both sides to encrypt the corresponding missed group key broadcast and responded to the request node. Later, Agrawal *et al.* [26] pointed out that in Tian's scheme, the location information of both request and cooperative nodes will be exposed to the adversary; Meanwhile, the bilinear pairing operation was adopted in the process of mutual-healing, which results in high computation overhead. Then, Agrawal improved Tian's scheme by symmetric encryption based on the historical group key, which is used not only to protect the confidentiality of location information, but also to authenticate the interactive messages. However, considering the risk of leaking historical group key by revoked nodes, the scheme still fails to prevent the location from leaking, as well as the message from forging. Additionally, Agrawal *et al.* [27] proposed a mutual-healing scheme based on the Chinese Remainder Theorem. The scheme constructed a series of auxiliary keys in advance based on the hash chain, and hid them in the group key broadcasts to synchronously update with group keys. In the mutual-healing process, the request node utilized the hash relation of adjacent auxiliary keys to verify the integrity and legality of the lost group key broadcast obtained from the cooperative nodes. However, when a node missed several consecutive group key broadcasts, its locally stored auxiliary keys will be out of sync with the latest auxiliary key in the group, which fails to recover the lost group key.

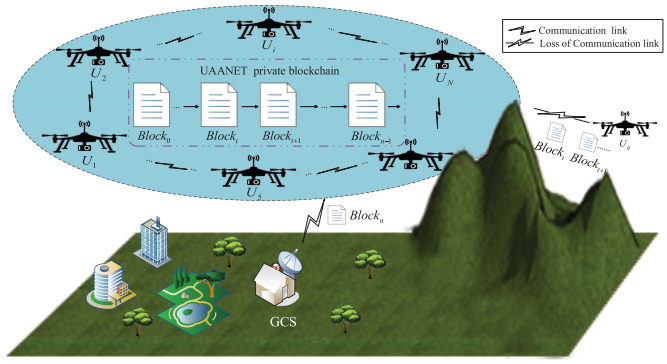



Fig. 1. The UAANET architecture. In this figure, when GCS distributes the $Block_t$ to update the group key $GK_t$, the $U_q$ flies out of the communication range of the group, which causes the loss of communication links, resulting in the failure to obtain $GK_t$ even the subsequent group keys.

It is worth noting that the existing schemes are all applied to static wireless sensor networks, where the security of mutual-healing depends on the authentication of neighbors. But, due to the fixed location and limited computing resources of nodes, as well as the low-frequency group membership changes in these networks, all the above schemes adopt a weak security mechanism, including the message authentication based on the neighboring location relation and historical group key. However, in UAANET, the group members change frequently, more importantly, the request node is unable to know the latest serial number that the group key had been updated when its communication links recover. So the attacker may replay the expired group key broadcast to deceive the request into recovering a non-latest group key, resulting in the leak of group messages sent by it.

In summary, all the existing group key distribution schemes in unreliable networks can not well solve the issue that nodes recover the lost group keys both securely and timely in UAANET.

## III. Preliminaries

### A. Network Architecture

Considering the scalability and wide coverage, the UAANET architecture deployed in this paper is shown in Fig. 1, which is an Ad Hoc network composed of GCS and a group of task-oriented UAVs [31].

For facilitate description, we let $U = \{U_1, U_2, \ldots\ldots, U_n\}$ be the finite universe of UAV nodes and each node $U_i$ has an unique identifier $ID_i$. The GCS sets up and manages the UAVs group by changing the group members dynamically based on task demands. Let $J_n \subseteq U \backslash UAV\_G_{n-1}$, $R_n \subseteq UAV\_G_{n-1}$ be the set of joined and revoked UAVs in $n$-th ($n \geq 1$) membership change with $J_1 \neq \emptyset$ and $J_n \cap R_n = \emptyset$, respectively. Hence, $UAV\_G_n = \{U_i | \forall U_i \in ((UAV\_G_{n-1} \cup J_n) \backslash R_n)\}$ denotes the communication group established by the GCS after the $n$-th membership change and specially $UAV\_G_0 = \emptyset$. At the same time, when the membership changes, the GCS will update group key (let $GK_n$ be the group key updated for the $n$-th) by updating the UAANET

blockchain, which acts as a distributed trusted database and its main function is to distribute and store group keys. Meanwhile, the blockchain runs in a private model, where UAVs need to be authenticated by the GCS and get permission to join or leave the network. Later, the GCS generates block $Block_n$ containing group key broadcast message and distributes through single-hop or multi-hop communication links established between UAVs. After receiving the block, the UAVs verify its legitimacy and update the blockchain, as well as the group key. Nevertheless, their communication links are unreliable (intermittent connection and disconnection), so it can not guarantee that all UAVs receive the block to update group key synchronously.

### B. Assumptions and Adversary Model

*Assumptions:* Firstly, In UAANET, the GCS is generally deployed in a secure environment, we assume that it is trusted. Secondly, for any UAV after communication links recover to normal, it is supposed that there is at least one legitimate UAV in its neighbors, which has the block containing the latest group key broadcast message, and also the links between it and its neighbors are symmetrical in a short time (namely, if node A is within the transmission range of node B, then node B is also within the transmission range of node A). It is worth noting that this assumption is an any-trust model, which has been widely used in service requests and its purpose is that request clients must trust that at least one legitimate response server behaves honestly, but not need to know or choose which server to trust [32], [33]. This is similar to the scenario in which the UAV node requests to recover the lost group key from its neighbors without knowing which neighbor to trust.

*Adversary Model:* Given the open nature of the broadcast channel and different external environments, we propose the following two attack models.

1) *The basic attack model:* This model assumes the presence of an active adversary, who can passively eavesdrop and intercept the messages being exchanged on the wireless channel. And they can also delete, modify or replay the message. Therefore, under the basic model, the attacker can launch forgery, tamper and replay attacks and so on.

2) *The enhanced attack model:* Besides the basic attack ability, we also assume that the adversary has a stronger capability which can suppress or shield messages sent by neighbors around the request node, such as electromagnetic interference, and further impersonates legal neighbors to respond messages to the request node.

## IV. Our Proposed Scheme

In this section, we describe the proposed mutual-healing group key distribution scheme based on blockchain in detail. And for readability, the notations used in the scheme are listed in Table I.

### A. Establishment of the Private Blockchain in UAANET.

The existing blockchain application modes fall into three categories: public blockchain, consortium blockchain, and private blockchain [34], [35]. The public one is also called permissionless blockchain, where anyone can participate in and compete to generate blocks. Both consortium blockchain and private blockchain can be categorized as permissioned blockchain, where participants need permission to join. Compared with the first two application modes, the private blockchain, in which the transactions are carried out only within the single organization and do not involve a large number of external users with low trust, has high consensus efficiency, fast transaction confirmation, and no forking attack [36]. Therefore, combining the network architecture and high real-time requirements, we design a private blockchain for the UAANET called $UAV\_BC$, as depicted in Fig. 2.

First, we redefine the structure of block to record the information of membership changes and group key broadcast messages. Then, referring to the application mode of private blockchain, where the node to generate blocks should be trusted by all others in the network, so it is natural for GCS to generate a block and distribute it to UAVs. After receiving the block, the UAVs verify its legitimacy, including the authenticity and integrity, and then update the $UAV\_BC$ stored locally. During this process, we need consensus mechanism to achieve consistency of the $UAV\_BC$. It is worth noting that in UAV networks, it is difficult to guarantee the consistency of the blockchain in a short time because some nodes may lose blocks due to the unstable communication links. So we will adopt the consensus mechanism which combines the broadcast and dynamic mutual-healing to ensure the consistency effectively. Specifically, it can be divided into two phases. In the first phase, the GCS broadcasts the block by the consensus mechanism raft [37], which has been

TABLE I
Symbols and Notations Used in the Scheme

| Notation | Description |
|---|---|
| $ID_x$ | A unique identifier for the entity $x$. |
| $GK_n$ | The group key updated by GCS for the $n$-th. |
| $B_n$ | Group key broadcast message sent by GCS to all the member of $UAV\_G_n$ for the $n$-th. |
| $(PK_x, SK_x)$ | Public/private key pair of entity $x$. |
| $Block_n$ | Block updated by GCS for the $n$-th and composed of header $Block_n^H$ and body $Block_n^B$. |
| $\|w\|$ | Size of sliding window $w$, where $\|w\| > 0$. |
| $m_1 \| \| m_2$ | Joint the messages $m_1$ and $m_2$. |
| $Sign_{SK_x}(\cdot)$ | Entity $x$ uses its private key $SK_x$ to sign message. |
| $Ver_{PK_x}(Sign_{SK_x}(\cdot))$ | Verification of signature $Sign_{SK_x}(\cdot)$ by the public key $PK_x$, and we define $Ver_{PK_x}(\cdot) = 1$, only it passes the verification. |

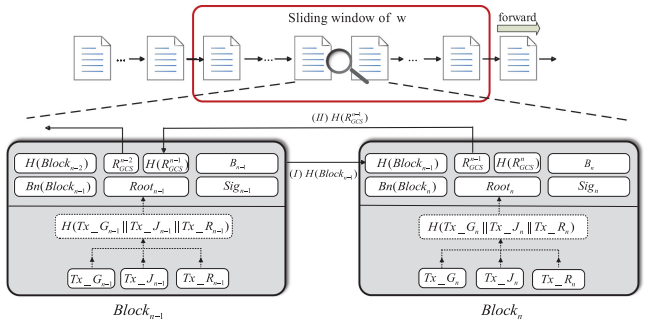



Fig. 2. The UAANET private blockchain.

widely adopted in private blockchain, to ensure that most UAVs in the network update the blockchain consistently. Meanwhile, considering the uniqueness of our scenario, instead of using the voting mechanism in raft protocol to determine who has the right to generate a block, which is known as leader election, in our proposal, only GCS can generate a block. In the second phase, the UAVs that have not completed the update of blockchain will adopt a mutual-healing method to retrieve lost blocks from their neighbors to update the blockchain $UAV\_BC$, where the specific details will be included in part B of this section.

Meanwhile, considering the validity period of a group message, nodes may only require to recover several recent lost group keys, so that UAV nodes can utilize a sliding window to store recently updated blocks. That is different from the existing blockchain network where nodes need to store all the historical blocks, the sliding window will significantly reduce the storage overhead of UAVs. Finally, the lifecycle of the $UAV\_BC$ is from the establishment of the task-oriented UAVs group to the dissolution of this group. And during the whole lifecycle, the $UAV\_BC$ is updated on demand with the change of group members to manage the dynamic list of UAANET membership certificates.

Now we will introduce the establishment process of the $UAV\_BC$ in detail.

1) *Initialization:* The GCS generates the genesis block $Block_0$, where its relevant information is stored as follows.

$\{Bn(Block_0), PK_{GCS}, H(R^0_{GCS}), Sign_{SK_{GCS}}(M)\}$

Here, let $Bn(Block_0) = 0$ denote the serial number of block. $H(R^0_{GCS})$ is the hash operation, where $R^0_{GCS}$ is a random number selected by GCS in the finite field $Z^*_p$, and also $M = Bn(Block_0)||PK_{GCS}||H(R^0_{GCS})$.

2) *Update:* Take the $n$-th ($n \geq 1$) members change as an example, the specific update procedure of $UAV\_BC$ is described as below.

*1) The GCS Firstly Authenticates the Joined UAVs and Then Assigns Relevant Information:* Each $U_i$, which belongs to $J_n$, sends the message containing its identity $ID_{U_i}$, public key $PK_{U_i}$ and corresponding signature to GCS. After completing the authentication and registration by verifying the signature, it will obtain the message $\{Prk_{U_i}, UAV\_BC_{n-1}\}$ assigned by the GCS through the secure channel, where $Prk_{U_i}$ is the secret key used to extract group key from the broadcast message; $UAV\_BC_{n-1}$ is the latest blockchain in UAANET, in particular, $UAV\_BC_0 = \{Block_0\}$.

*2) The GCS Generates the Block $Block_n$ and Broadcasts it to Update the $UAV\_BC$, as well as the Group Key:* The storage contents of $Block_n$, shown as Fig. 2 above, include:

$$Block_n = \{Block^H_n, Block^B_n\}$$

Where, let $Block^H_n = \{H(Block_{n-1}), R^{n-1}_{GCS}, H(R^n_{GCS}), Bn(Block_n), B_{n-1}, Root_n, Sig_n\}$. Among them, $H(Block_{n-1})$ denotes the hash of the former block, which is utilized to establish the forward hash chain relation (as (I) shown in Fig. 2). For convenience, we denote the forward relation by the function $f^{(I)}(Block_{n-1}, Block_n)$. And if and only if it is correctly established, we define the function value is equal to 1. $H(R^n_{GCS})$ is the hash of random number $R^n_{GCS} \in Z^*_p$ chosen by the GCS, which is utilized to establish the backward hash chain relation (as (II) shown in Fig. 2). Similarly, we denote this relation by the function $f^{(II)}(Block_{n-1}, Block_n)$ and also define its value equal to 1, only it is correctly established. Let $Bn(Block_n) = n$ denote the serial number of block and also represent the $n$-th update of group key which is done by distributing the group key broadcast message $B_n$. It is worth noting that in our scheme we focus on addressing the problem of secure mutual-healing, for the generation of message $B_n$, instead of devising a new one, we adopt the existing security schemes which can be abstracted as $B_n = \eta(MSK, UAV\_G_n, GK_n)$(Specifically, $MSK$ is the master key stored by the GCS secretly; $\eta(\cdot)$ is the group key broadcast message generation algorithm and also there is a unique inverse algorithm $\eta^{-1}(\cdot)$ that for each $U_i$, if and only if $U_i \in UAV\_G_n$ can extract $GK_n = \eta^{-1}(B_n, Prk_{U_i})$). $Root_n = H(Tx\_G_n||Tx\_J_n||Tx\_R_n)$ denotes the hash of all transactions stored in the $Block^B_n$. In our scheme, transactions fall into three categories, and they are all generated by the GCS. Specifically, $Tx\_J_n = \{(ID_{U_i}, PK_{U_i})|\forall U_i \in J_n\}$, $Tx\_R_n = \{(ID_{U_i}, PK_{U_i})|\forall U_i \in R_n\}$ denote the details of the UAVs who are joined and revoked by GCS in the $n$-th group membership change, respectively. $Tx\_G_n$ denotes the information of group membership, formally defined as the following two cases:

- If $|w| = 1$ or $n$ mod $|w| = 1$.
  $Tx\_G_n = \{(ID_{U_i}, PK_{U_i})|\forall U_i \in UAV\_G_n\}$;
- Others.
  $Tx\_G_n = \emptyset$;

Therefore, it can be seen that the GCS stores global information of UAVs group every $|w|$ times group membership changes and only the information of changed members inside the sliding window $w$. This not only reduces the storage overhead of the UAV nodes but also ensures that they can grasp all the membership of UAVNET to maintain a dynamic trust relationship in real-time. Lastly, let $Sig_n = Sign_{SK_{GCS}}(H(Block_{n-1})||R^{n-1}_{GCS}||H(R^n_{GCS})||Bn(Block_n)||B_n||Root_n)$ denote the signature of block.

*3) The UAV Nodes Update the Local Blockchain and Group Key:* Each UAV node receiving the $Block_n$ verifies its legitimacy by comparing with its latest blockchian (assuming that $BlockChain_t$) stored locally. Simply, the verification process can be formally described as the following logical formulas: $(n == t+1) \wedge (f^{(I)}(Block_t, Block_n) == 1) \wedge (f^{(II)}(Block_t, Block_n) == 1) \wedge (Ver_{PK_{GCS}}(Sig_n) == 1)$, where $Block_t$ is the last block in $BlockChain_t$, "$\wedge$" denotes logical connectors "and". Only when all of them are verified to be true, the UAV nodes update the local blockchain to $UAV\_BC_n$ by sliding window to save the latest $|w|$ blocks, as well as the genesis block $Block_0$.

### *B. Recovery of the Lost Group Keys by Mutual-Healing*

Assuming that the UAV node $U_q$ loses its communication link with the UAANET after it has completed the $t$-th group key update. In order to recover communication with the UAVs group,

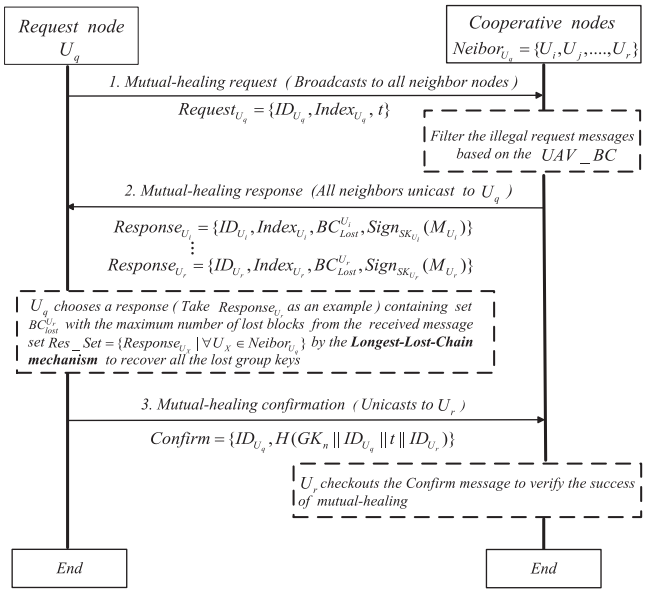



Fig. 3. The basic mutual-healing protocol.

$U_q$ will continuously detect its neighbors by secure periodic Hello messages mechanism [38]. When the communication links recover to normal, the $U_q$ can detect all neighboring nodes around it, which is denoted by the set $Neibor_{U_q}$ and also let $Q = |Neibor_{U_q}|$ be the number of nodes in the set. And then, the $U_q$ will request the neighbors to assist in recovering all the lost blocks, as well as the group keys $GK_m, \forall m \in (t, n]$ by mutual-healing. Next, we will propose two corresponding mutual-healing protocols under the basic and enhanced attack models, respectively.

*1) The Basic Mutual-healing Protocol:* The realization of this basic mutual-healing protocol can be described as the following simple and efficient three steps in Fig. 3.

*a) Mutual-healing request:* $U_q$ broadcasts a request message denoted by $Request_{U_q}$ to all around neighboring UAV nodes, where includes its identity $ID_{U_q}$, the serial number $Index_{U_q}$ of block which stored the transaction consisting of $(ID_{U_q}, PK_{U_q})$, and the serial number $t$ of the last block stored locally.

*b) Mutual-healing response:* Upon receipt of a request, the neighboring UAV nodes first need to ascertain that the validity of identity $ID_{U_q}$ by querying the $UAV\_BC_n$ to filter a large number of invalid request messages. If it is valid, each node $U_X \in Neibor_{U_q}$ unicasts a response message $\{ID_{U_X}, Index_{U_X}, BC^{U_X}_{Lost}, Sign_{SK_{U_X}}(M_{U_X})\}$ denoted by $Response_{U_X}$ to $U_q$. Here, $BC^{U_X}_{Lost}$ (called the lost block set) denotes the set of blocks which updated by the GCS during the communication link interruption of $U_q$. And it can be expressed as $BC^{U_X}_{Lost} = \{Block_i | \forall i \in (t, n]\}$, if $U_X$ is the legitimate member of the current UAANET. $Sign_{SK_{U_X}}(M_{U_X})$ is the signature of response message, where $M_{U_X} = ID_{U_X} || Index_{U_X} || BC^{U_X}_{Lost}$.

Since the $U_q$ broadcasts request, it will receive multiple response messages from different neighbors at the same time. For further description, we define the set of response message as $Res\_Set = \{Response_{U_X} | \forall U_X \in Neibor_{U_q}\}$, and then the $U_q$ will adopt the Longest-Lost-Chain mechanism to recover all the lost group keys. More specifically, all the response messages in the $Res\_Set$ are sorted by the length of the containing lost blocks set, and the $U_q$ selects a legitimate response message containing the maximum number of lost blocks(namely the longest lost blockchain), where the legitimacy here refers to the fact that all blocks in the set $BC^{U_X}_{Lost}$ are generated by the GCS and have not been tampered with. For being easy to describe, we take the final selected response message $Response_{U_r}$ as an example, which satisfies:

$$\begin{cases} \forall m \in (t, k], f^{(I)}(Block_{m-1}, Block_m) == 1 \\ Ver_{PK_{GCS}}(Sig_k) == 1 \\ Ver_{PK_{U_r}}(Sign_{SK_{U_r}}(\cdot)) == 1 \end{cases}$$

Where $k = |BC^{U_r}_{Lost}| + t$, and "|.|" denotes the number of blocks contained in the lost block set (namely, the length of the set). It is noteworthy that in the above process, we utilize the hash chain relation of $UAV\_BC$ to verify the whole lost block set in batches only by verifying the signature of the last block. Compared with the block verification one by one, it can greatly improve the efficiency.

After that, the request node $U_q$ can recover all the lost group keys from the set $BC^{U_r}_{Lost}$ and sends a confirmation message $Confirm = \{ID_{U_q}, H(GK_n || ID_{U_q} || t || ID_{U_r})\}$ to indicate that the mutual-healing is successful.

From the above implementation process, we can see that the main advantage of this basic mutual-healing protocol is that the request nodes only rely on the trust of the stored data in the blockchain to ensure the security in the process of mutual healing, which is more secure and effective than the existing scheme under this basic attack model.

*2) The Enhanced Mutual-Healing Protocol:* Although the basic version has higher efficiency, it can not defend against the suppression attack under the enhanced attack model. Consider an extreme case that there is only one legitimate UAV around the request node, if adversary suppresses the messages sent by this only legitimate neighbor, such as electromagnetic interference, and then impersonates it and replays the non-latest lost blockchain (denoted by $BC'_{lost} \subsetneqq \{Block_i | \forall i \in (t, n]\}$) to deceive request node into recovering a non-latest group key. For this reason, we propose an enhanced mutual-healing protocol depicted in Fig. 4, which not only ensures that nodes recover the lost group key securely, but also establishes a secure communication connection between the request node and cooperative node.

*a) Mutual-healing request:* The first step is basically the same as that in the basic protocol, where the only difference is that a random number $R_{U_q} \in Z^*_p$ is contained to ensure the freshness of request message.

*b) Mutual-healing response:* Upon receipt of a request, each node $U_X \in Neibor_{U_q}$ does the same process as the basic

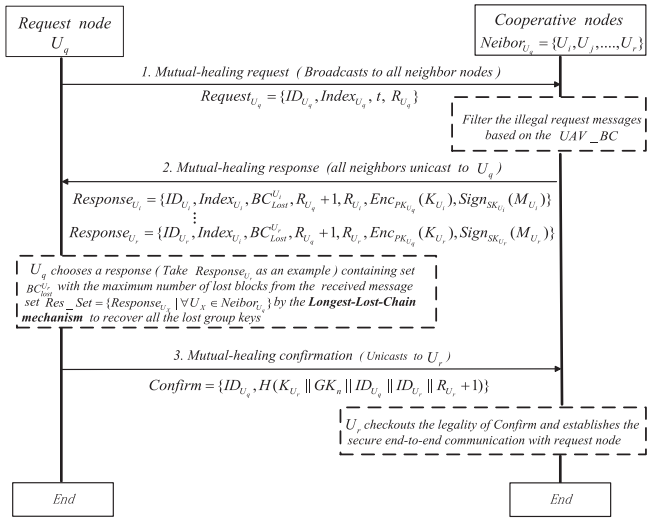



Fig. 4. The enhanced mutual-healing protocol.

protocol and returns a response message $Response_{U_X}$(shown as Fig. 4), which adds the following contents compared with the basic version. Specifically, $Enc_{PK_{U_q}}(K_{U_X})$ denotes the asymmetric encryption using the public key of $U_q$ and the $K_{U_X}\in Z_p^*$ is a session key selected by the node $U_X$, which is used to establish end-to-end secure communication with the request node $U_q$ after completing the mutual healing.

After receiving the respond message set $Res\_Set$, to prevent malicious nodes from launching suppression attack, the $U_q$ needs to ensure that the response messages in the set do come from $Q$ different neighbor nodes in $Neibor_{U_q}$ and the messages have not been tampered with. Specifically, for each $Response_{U_X}\in Res\_Set$, it needs to satisfy:

$$\begin{cases} ID_{U_X}\in Neibor_{U_q} \\ Ver_{PK_{GCS}}(Sig_{Index_{U_X}})==1 \\ Ver_{PK_{U_X}}(Sign_{U_X}(\cdot))==1 \end{cases}$$

Where $Sig_{Index_{U_X}}(\cdot)$ denotes the signature of the GCS stored in the $Block_{Index_{U_X}}$. In order to improve the efficiency of the above verification process, batch verification algorithm, such as [39], can be adopted, while the implementation process is beyond the scope of our scheme. After the set $Res\_Set$ passes validation, $U_q$ will recover all lost group keys by the Longest-Lost-Chain mechanism, which is the same process as the basic version. As before, we still assume that the selected legitimate response message comes from $U_r$, and $U_q$ can further authenticate the identity of $U_r$ by verifying whether both the conditions that $ID_{U_r}\in Tx\_J_{Index_{U_r}}\cup Tx\_G_{Index_{U_r}}$ and $ID_{U_r}\notin\bigcup_{c=Index_{U_r}+1}^{k}Tx\_R_c$ are satisfied.

*c) Mutual-healing confirmation:* Later, the request node $U_q$ obtains the end-to-end session key $K_{U_r}$ by decryption and unicasts the confirmation message $Confirm$ (shown as Fig. 4) to the cooperative node $U_r$. On one hand, it indicates that mutual-healing has been completed, on the other hand, it indicates the successful establishment of a secure end-to-end connection.

## V. Scheme Analysis

### A. Security Analysis

*1) Credibility of the Private Blockchain $UAV\_BC$:* Our scheme is implemented in two phases, where the establishment of the private blockchain is the security foundation for group key update and mutual-healing. In other words, it needs to ensure the authenticity and credibility of data stored in the $UAV\_BC$.

*Theorem 1:* In the blockchain establishment phase, the scheme can resist forgery, tampering and replay attacks effectively to ensure the credible establishment of the $UAV\_BC$.

*Proof:* We take the $n$-th membership change as an example. First, the node that generates the new block should provide the random number $R_{GCS}^{n-1}$ to satisfy the backward hash chain relationship. Meanwhile, it needs to be signed to ensure integrity. Considering the confidentiality of the random number $R_{GCS}^{n-1}$ and private key $SK_{GCS}$, it will make only the GCS can generate a block. Then, the previous block hash $H(Block_{n-1})$ must be included in the $Block_n$. When malicious nodes replay the historical legitimate blocks $Block_i(i\leq n-1)$, the forward hash chain relationship will not be satisfied. Therefore, the generation process of blocks can effectively resist forgery, tampering and replay attacks to ensure that the private blockchain $UAV\_BC$ is established credibly. ■

*2) Resistance to Replay Attack:* Due to the fact that the $U_q$ can not know the latest serial number that the group key had been updated when its communication link recovers to normal, the attacker may send a false response message by replaying a non-latest lost group key broadcast to spoof the $U_q$ to recover an expired group key, which will be mistaken as the latest one, resulting in the leak of group messages sent by $U_q$.

In the process of recovering lost group keys, we propose two specific mutual-healing protocols, the basic and enhanced version, under different attack models. Next, we will demonstrate that both the proposed protocols can resist the above replay attack under their respective attack model.

*Theorem 2:* In the basic mutual-healing protocol, the request node adopts the Longest-Lost-Chain mechanism, which can effectively resist the replay attack and ensure that the request node can get the latest group key under the basic attack model.

*Proof:* First, according to the second assumption in Section III, there is at least one legitimate member in the $U_q$'s neighbors who has the latest blockchain. Meanwhile, In the basic attack model, the attacker does not have the ability to suppress or shield response messages sent by neighbors, so there will be at least one valid response message in the set $Res\_Set$ which comes from the legitimate neighbor node and contains all the lost blocks of $U_q$ (denoted by $BC_{lost}^{real}=\{Block_i|\forall i,i\in(t,n]\}$).

Then, based on the above, we classify the set $Res\_Set$ into two categories: the response messages of malicious nodes and the response messages of legitimate nodes. For the first category, the contained lost block set is denoted by $BC'_{lost}$. Theorem 1 shows that malicious nodes cannot forge and tamper with blocks, in order to deceive the request UAV into mistaking an expired group key $GK_i(t<i<n)$ as the latest one, the lost block set sent from the malicious nodes should satisfy $BC'_{lost}\subsetneq BC_{lost}^{real}$, that is $|BC'_{lost}|<|BC_{lost}^{real}|$. So we can see that the response

messages of legitimate nodes contain the longest lost blockchain in set $Res\_Set$.

To sum up, under the basic attack model, It can be sure that the response message selected from the set $Res\_Set$ by the Longest-Lost-Chain machine is the response message coming from the legitimate node and containing the latest group key broadcast message, which can resist the group key replay attack. ■

*Theorem 3:* In the enhanced mutual-healing protocol, the request node adopts the Longest-Lost-Chain mechanism, which can effectively resist the replay attack and ensure that the request node can get the latest group key under the enhanced attack model.

*Proof:* Under the enhanced attack model, although the attacker has stronger attack ability, we add the signature of the node in the response message, which will make the malicious node unable to impersonate identity, even if a suppression attack is launched. When all the response massages in $Res\_Set$ passes the signature verification, which means they do come from Q neighbor nodes without impersonation and further indicates that the response messages sent by the legitimate node are contained in $Res\_Set$. Meanwhile, to deceive the request UAV node into mistaking an expired group key $GK_i(t < i < n)$ as the latest one, the lost block set from the malicious nodes still should satisfy $BC'_{lost} \subsetneqq BC^{real}_{lost}$, which means that the response messages of legitimate nodes still contain the longest lost blockchain in set $Res\_Set$.

Therefore, in the enhanced mutual-healing protocol, the Longest-Lost-Chain mechanism can still be adopted to select the response messages containing the longest blockchain from the legitimate node, which can also resists the group key replay attack. ■

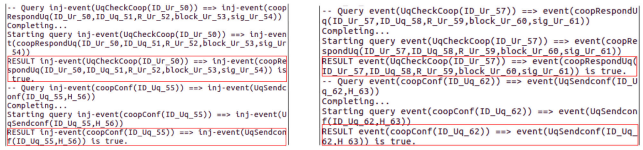
*3) Mutual Authentication:* In the enhanced mutual-healing protocol, two nodes participating in the mutual-healing session can authenticate each other.

*Theorem 4:* In the mutual-healing phase, the scheme can achieve mutual authentication of the request UAV and cooperate UAV to establish a secure end-to-end communication.

*Proof:* Based on the above Theorem 1, 3, we adopt ProVerif (PV) [40] to prove the security goal in Theorem 4 automatically. As an automatic cryptographic protocol verifier, PV can test various typical security properties of protocols. In this process, we first model the proposed mutual-healing protocol by defining the following key events:

- **event** $RespondU_q(ID_{U_r}, BC^{U_r}_{Lost}, R_{U_r}, Sign_{SK_{U_r}}(\cdot))$*:* The cooperative UAV receives the request message, verifies its legality and generates the response message.
- **event** $U_qCheckCoop(ID_{U_r})$*:* The request UAV receives the response message and verifies its legality by the theorem 2.
- **event** $U_qSendConf(ID_{U_q}, H_{GK_n}(\cdot))$*:* The request UAV generates confirmation message.
- **event** $CoopConf(ID_{U_q})$*:* The cooperative UAV receives confirmation message and verifies its legality.

And then we simulate arbitrarily many sessions between the request and cooperative UAV by instantiating the respective processes. Later, the following two security properties are described with the query assertion, and the corresponding proof results are given.




(a) security property 1. (b) security property 2.

Fig. 5. Proof results. (a) Security Property 1. (b) Security Property 2.

TABLE II
COMPARISON OF SECURITY FEATURES WITH THE EXISTING MUTUAL-HEALING SCHEMES

| **Schemes ⇒ Attributes⇓** | | Tain *et al.* [25] | Agrawal *et al.* [26] | Agrawal *et al.* [27] | Our Basic | Our Enhanced |
|---|---|---|---|---|---|---|
| **Resistance to Replay Attack** | Basic attack model | × | × | × | √ | √ |
| | Enhanced attack model | × | × | × | × | √ |
| **Mutual Authentication** | | × | × | × | × | √ |
| **Mutual-healing Confirmation** | | × | √ | √ | √ | √ |

*a) Interactive messages between request UAV and cooperative UAV are fresh:* This property is about freshness. Therefore, the corresponding event relationship can be described as: for each response message verification event, only one response message generation event has completed; Similarly, for each confirmation message verification event, only one confirmation message generation event has completed. The assertion that proves this property is as follows:

$$inj-event(U_qCheckCoop) \Rightarrow inj-event(RespondU_q);$$

$$inj-event(U_qSendConf) \Rightarrow inj-event(CoopConf).$$

*b) Request UAV and cooperative UAV can achieve mutual authentication:* This property is about identity authentication. Firstly, combining with the Longest-Lost-Chain mechanism, the request UAV can authenticate the cooperative UAV and recovers the latest group key correctly. Secondly, After validating the confirmation message, the cooperative UAV can authentication the request UAV. So the assertion that proves this property is as follows:

$$event(U_qCheckCoop) \Rightarrow event(RespondU_q);$$

$$event(U_qSendConf) \Rightarrow event(CoopConf).$$

As shown in Fig. 5, all the proof results are true, which indicates the security properties described in query assertion have been automatically proved, as well as the Theorem 4. ■

Meanwhile, we compare the security of our proposed with the existing mutual-healing schemes [25]–[27]. The results are shown in Table II. Taking full account of the frequent membership changes and unfixed locations in UAANET, we adopt the blockchain to manage the dynamic list of membership certificates and based on this, the trust between the request and cooperative node can be reliably established. Meanwhile, we implement authentication protection in the interactive messages

TABLE III
COMPARISON OF PERFORMANCE WITH THE EXISTING MUTUAL-HEALING SCHEMES

| Schemes / Attributes | Node | | Tian *et al.* [25] | Agrawal *et al.* [26] | Agrawal *et al.* [27] | Our Basic | Our Enhanced |
|---|---|---|---|---|---|---|---|
| Comm. Overhead | R | | $3\log p$ | $8\log p$ | $5\log p$ | $5\log p$ | $6\log p$ |
| | C | | $((2\lvert G_n\rvert+3)\cdot n+2)\log p$ | $((2\lvert G_n\rvert+3)+4)\log p$ | $(5\cdot(n-t)+3)\log p$ | $3\log p+\sum_{k=t+1}^{n} Size(Block_k)$ | $6\log p+\sum_{k=t+1}^{n} Size(Block_k)$ |
| Comp. Overhead | R | S_1 | $T_{bp}+T_h+T_{sd}$ | $2T_{se}+3T_h+T_{sd}$ | $(n-t-1)\cdot(3T_h+4T_b)+2T_h$ | $(2(n-t)+1)\cdot T_h+2T_v$ | $(2(n-t)+1)\cdot T_h+(2Q+1)\cdot T_v+T_d$ |
| | | S_2 | $2T_{bp}+\lvert G_n\rvert\cdot T_p+T_h+T_b$ | $2T_{bp}+T_h+T_b$ | $T_h+3T_b$ | $T_{\eta^{-1}}$ | $T_{\eta^{-1}}$ |
| | C | | $T_{bp}+T_h+T_{se}$ | $T_{se}+3T_h+2T_{sd}$ | $3T_h+T_b$ | $T_h$ | $2T_h+T_s+T_e$ |

**Notations:**
**R/C**: Request/Cooperative node; **S_1/2**: Get $B_n$ from the neighbors/Extract $GK_n$ from the $B_n$; $t$: The serial number of the latest group key owned by the request node; $n$: The serial number of the latest group key for the current group; $|G_n|$: Number of members in the group after the $n$-th membership change; $Q$: The number of neighbors around the request node; $T_{bp}$: Time for pairing operation; $T_p$: Time for scalar multiplication; $T_{se}/T_{sd}$: Time for symmetric encryption/decryption; $T_h$: Time for symmetric key operations such as Hash/PRF; $T_s/T_v$: Time for asymmetric signature/verification; $T_e/T_d$: Time for asymmetric encryption/decryption; $T_b$: Time for basic operation such as Mod/Xor; $log_p$: Size of prime $p$ (which used to generate finite fields $Z_p^*$) in bits.

through asymmetric keys. All of these provide strong security measures to ensure that UAV nodes correctly recover all the lost group keys, especially the latest one.

### B. Computation and Communication Overhead

In our scheme, the following algorithms are involved, namely, $\eta(\cdot)$, $\eta^{-1}(\cdot)$, $Sign(\cdot)$, $Ver(\cdot)$, $Enc(\cdot)$, $Dec(\cdot)$, $Hash(\cdot)$. And their computation overhead are represented by $T_\eta$, $T_{\eta^{-1}}$, $T_s$, $T_v$, $T_e$, $T_d$ and $T_h$, respectively. Next, we will analyze the overall computation overhead of the related entities in different processes.

1) *In the Process of Establishing the Private Blockchain*: the GCS first updates the broadcast message $B_n$ for distributing group key with the algorithm $\eta(\cdot)$, and then stores it with the membership information $UAV\_Info_n$ in block $Block_n$. The overall computation overhead of the GCS is $T_\eta+3T_h+T_s$.

When receiving the $Block_n$, the UAVs first verify the forward and backward hash chain relation based on the latest local block $Block_t$, and then verify the signature of the $Block_n$. After all the above verifications are passed, the UAVs extract the updated group key $GK_n$ with algorithm $\eta^{-1}(\cdot)$. During this process, the overall computation overhead needed for each UAV is $3T_h+T_v+T_{\eta^{-1}}$.

2) *In the Process of Recovering Lost Group Keys*: both the proposed two mutual-healing protocols have the same interaction process, the only difference is the construction and verification of interactive messages, which will result in different computation overhead.

Next, we will briefly analyze the computation overhead of entities in the corresponding protocols. Specifically, the request node $U_q$ first broadcasts a request message $Request_{U_q}$ to its neighbors. When receiving the $Request_{U_q}$, the cooperative node $U_r$ first queries the local $UAV\_BC_n$ to verify its validity, and then unicasts a response message $Response_{U_r}$. Later, it will receive the confirmation from $U_q$ to verify whether the mutual-healing process is successful. Hence, the overall computation overhead needed for the cooperative node in the basic and enhanced protocols are $T_h$ and $2T_h+T_s+T_e$, respectively.

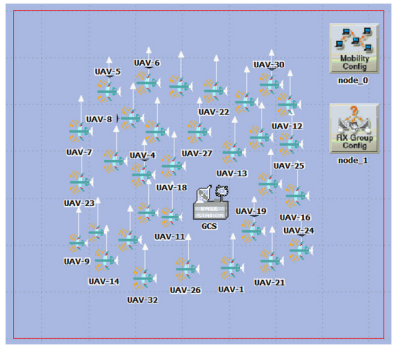



Fig. 6. The UAANET simulation scenario.

After receiving the response message set $Ret\_Set$, the request node $U_q$ first selects the legitimate response message by the Longest-Lost-Chain mechanism to recover all the lost group keys $GK_m, \forall m\in(t,n]$, and then unicasts the confirmation to the cooperative node to indicate that the lost group keys have been recovered successfully. So the overall computation overhead of $U_q$ in the basic and enhanced protocols are $(2(n-t)+1)\cdot T_h+2T_v+(n-t)\cdot T_{\eta^{-1}}$ and $(2(n-t)+1)\cdot T_h+(2Q+1)\cdot T_v+(n-t)\cdot T_{\eta^{-1}}+T_d$, respectively.

Meanwhile, the computation and communication overhead of related entities in the process of recovering the lost group keys (take the latest group key $GK_n$ as an example) are compared with the existing mutual-healing schemes [25]–[27], which is shown in Table III.

## VI. EXPERIMENTS

### A. Experiment Setup

In the experiment, we simulate the UAANET scenario as shown in Fig. 6. In this scenario, we set up several UAV nodes and a Ground Control Station, where their initial positions are randomly distributed. During the simulation, the GCS keeps

TABLE IV
PARAMETERS OF UAANET SIMULATION

| Parameter | Value |
|---|---|
| Flight speed | ① Uniform[40,60] m/s<br>② Uniform[100,120] m/s |
| Flight range | 78 $km^2$ |
| Communication distance | 1500 m |
| Date transmission rate | 11 Mbps |
| Initial group members | 32 |
| Simulation time | 300 s |

static at the initial position and the Random Waypoint (RWP) model [41] is adopted by UAV nodes to reflect the dynamic change of topology. At the same time, the UAANET adopts the AODV routing protocol [42]. As a typical on-demand-driven routing protocol, it realizes dynamic and spontaneous routing between mobile nodes to be fit for frequently changing of topology, which has been widely used in mobile Ad Hoc networks, including the UAANET [43].

Experimental environment: the PC hardware configuration information is 3.30 GHz Core i5-4590 CPU, 4 GB DDR3-1600 RAM and the operating system is Windows 7. The simulation software chooses OPNET Modeler 14.5, which is a conventional wireless network simulation tool with the advantages of a rich model library, simple model configuration, and fast data analysis. We set the simulation parameters of UAANET scenario as shown in Table IV.

We deploy our blockchain $UAV\_BC$ with Hyperledger Fabric 2.0. As an open-source blockchain development platform, it has been widely used in various scenarios [44]. Meanwhile, it consists of four layers from bottom to top by the blockchain model. They are the data layer, network layer, consensus layer and application layer, respectively. Although it is often designed as consortium blockchain platform, its each layer has pluggable and customizable characteristics. Therefore, in order to be suitable for the UAANET private blockchain, we combine the proposed scheme and make corresponding modifications to its layers. More specifically, in the data layer, we first add fields in the transaction to store group key broadcast messages and membership change information, and then redefine the storage structure of block header, as shown in Fig. 2. In the network layer, which is responsible for the transmission and validation of block between nodes, we still adopt the P2P networking mechanism and data transmission mechanism included in its source code. Next, in the consensus layer, based on the consensus algorithm raft, we modify the leader election method to fix the GCS as the leader node to generate blocks. Finally, in the contract and application layer, we combine Hyperleader's ChainCode technology and use the Java SDK "fabric-sdk-java" to design smart contract algorithms for updating, storing and acquiring group keys, and the parameters involved are set as follows: the prime $p$ and the group key $GK_n$ are 2014 bit and 128 bits, respectively; hash operation is achieved by the standard SHA256; ECC-secp256k1 is employed for encryption/decryption, and ECDSA-secp256k1 for signature/verification, which provides 256 bits of security protection.

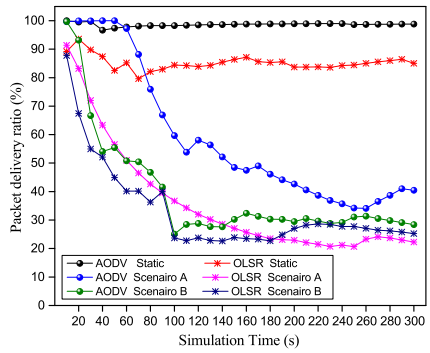



Fig. 7. Packet delivery ratio.

Also, we use the group key update scheme proposed in [45] to implement the algorithm $\eta(\cdot)$ and its inverse algorithm $\eta^{-1}(\cdot)$, which not only has lower computation and communication overhead but also satisfies various security attributes.

### B. Experimental Results and Analysis

*1) Packet Delivery Ratio in UAANET:* First, we test the stability of the communication link in the network. It can be reflected by the packet delivery ratio which is the ratio of data received by the destination node and data generated by the source node. In the experiment, we copy three UAANET scenarios according to the parameters in Table IV, specifically, one static scene where nodes remain stationary and two dynamic scenes (named A,B) where the flight speed of nodes is set as ① and ②, respectively. In all the above scenarios, we select the same UAV node (UAV-6) as destination node and the GCS as source node where its packet sending rate is set to 100 packets/sec and the size of each packet is 20000 bit. Meanwhile, to make the experiment more convincing, another common dynamic routing protocol OLSR [46] is also adopted. In the whole simulation, we measured the packet delivery ratio under different scenarios with different routing protocols, respectively, and the results are shown in Fig. 7.

From the above results, we can see that the UAANET has packet loss under different routing protocols, and the faster the node moves, the lower the packet packet delivery ratio will be. Take the AODV routing protocol as an example, when the speed changes from 0 to [100,120]m/s, the packet delivery ratio decreases from 96% to 30% and also changes dynamically. This indicates that in the dynamic UAV network, the stability of communication links is poor, which results in frequent loss of messages.

*2) Real-Time Comparison of Different Mechanisms:* We first compare the mechanisms of mutual-healing and retransmission. In both scenes A and B, the end-to-end communication delay of the selected node to the GCS and its surrounding nodes are measured during the whole simulation, which respectively reflects the real-time of the retransmission and mutual-healing mechanism. The results are shown in Fig. 8.

In the retransmission mechanism, the request node and the GCS need to establish a dynamic multi-hop routing to obtain the lost group key broadcasts, so that the communication delay

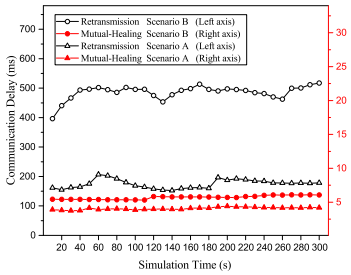



Fig. 8. Real-time comparison of retransmission.

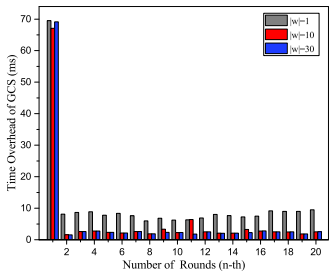



Fig. 10. Blockchain update time overhead.

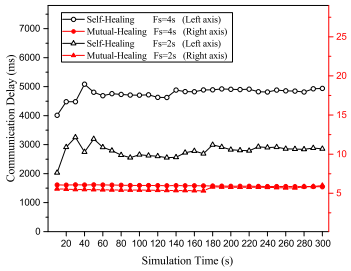



Fig. 9. Real-time comparison of self-healing.

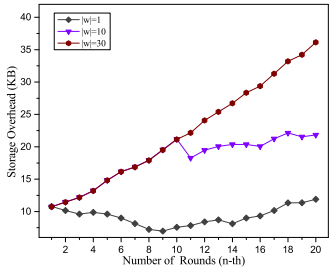



Fig. 11. Blockchain storage overhead.

will change dynamically. For example, in scenario A of Fig. 8, the delay of retransmission waves randomly with poor stability, wherein the maximum delay is 0.21 s, and the minimum delay is 0.15 s. However, in the mutual-healing mechanism, the lost group key broadcasts are sent directly by its neighbors, avoiding the multi-hop. For example, in scenario A of Fig. 8, the average delay of mutual-healing is about 0.004 s and keeps steady. So compared with the retransmission, the mutual-healing mechanism has higher real-time performance and stability. Meanwhile, by comparing the change of delay curve in scenarios A and B, it can be seen that with the increase of flight speed which will make the dynamic change of network topology intensify, the delay of nodes obtaining the lost broadcast messages will also increase. But in contrast, the retransmission mechanism has more obvious fluctuation.

Then we compare the mutual-healing mechanism with the self-healing mechanism. In the experiment, we take the UAANET scenario A as an example and the update frequency of group keys is set to 2 s and 4 s, respectively. And the communication delay of the selected node that receives the next broadcast message from GCS is measured, which represents the delay of obtaining the lost group key broadcast message by self-healing mechanism. The experimental results are shown in Fig. 9. In the self-healing mechanism, for recovering the lost group keys, the node needs to wait for the next group key broadcast passively, which mainly depends on the frequency of the group key updated by the GCS. However, in the mutual-healing mechanism, the node actively requests its neighbors, which has apparent higher real-time performance.

*3) Performance Analysis and Comparison With the Existing Schemes:*

*a) Establishment of the private blockchain $UAV\_BC$:* Firstly, we analyze the impact of membership changes and the size of sliding window on the blockchain $UAV\_BC$ update time and storage overhead. In the specific experiment, we set the sliding window size $|w| = 1, 10, 30$, and perform the membership change operation for 20 rounds to record the $UAV\_BC$ update latency, as well as the $UAV\_BC$ storage overhead for each round. The results are shown in Fig. 10 and Fig. 11.

In each round, the GCS generates a new block and broadcasts it to update the $UAV\_BC$. The time overhead of this process mainly contains two parts: *(i)* The generation time of the new block, which is determined by the computational complexity of $\eta(\cdot)$ algorithm. Specifically, in our experiment $O(\eta) = O(|J_n| + |R_n|)$. Therefore, the new block generation time changes according to the change of group memberships. *(ii)* The transmission time of the new block, which is directly influenced by the size of the block. In our scheme, for each $|w|$ rounds, the GCS will save all group members information, and in other cases, it only saves the changed membership information in the current time. Especially, when $|w| = 1$, the information of all group members has to be saved in each round, which will have the largest block size comparing with other $|w|$ values, leading to the longest transmission time delay. As seen in Fig. 10, on the whole, a larger sliding window $w$ will reduce the average delay of the $UAV\_BC$ update.

Fig. 11 shows the specific changes in the storage overhead of nodes under different sliding window sizes when the group members are dynamically changing. In our scheme, the sliding window mechanism makes each node store only $|w| + 1$ blocks. When $0 < n \leq |w|$, the storage overhead of nodes will increase linearly with the number of round $n$. When $n > |w|$, due to the sliding window, the change in storage overhead for two adjacent rounds is $\Delta = Size(Block_n) - Size(Block_{n-w})$, which indicates that the sliding window mechanism designed in our scheme can effectively reduce the storage overhead of $UAV\_BC$.

Note that, Fig. 10 shows that the block update time will be increased if the sliding window is too small, while the storage

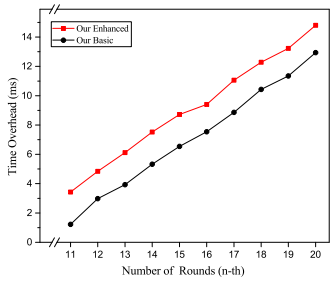



Fig. 12. Time overhead of the cooperative node.

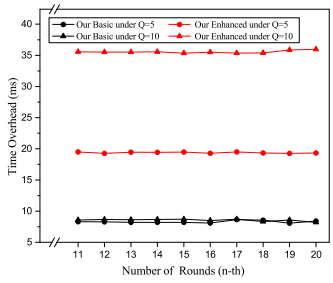



Fig. 13. Time overhead of the request node.

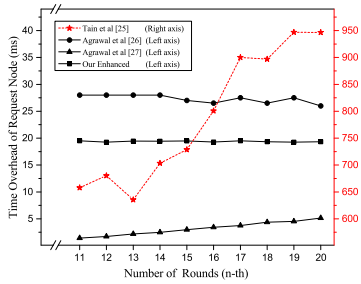



Fig. 14. Time overhead comparison of the request node.

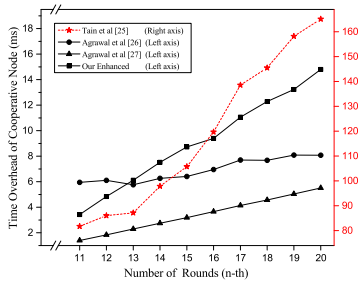



Fig. 15. Time overhead comparison of the cooperative node.

cost of $UAV\_BC$ will be increased if the sliding window is too large. Therefore, In practice, we should trade off the time overhead of group key update and the storage overhead of nodes to set appropriate sliding window size.

*b) Recovery of the lost group keys by mutual-healing:* There are two main factors that affect the time overhead of related nodes in the mutual-healing process, namely, the number of group keys (denoted by $R$) updated in the loss of communication links and the number of neighbors (denoted by $Q$) around the request node when the communication links recover. In the specific experiment, we set $|w| = 10$ and also perform 20 rounds group membership change. During this process, we disconnect a UAV node after the 10-th round, then re-connect it after the $n$-th ($t < n \leq 20$) and implement the basic and enhanced mutual-healing protocol, respectively, to record the time overhead of cooperative and request nodes with different $R$ (replaced by the change of $n$) and $Q$ (set $Q = 5, 10$ for example). The results are shown in Fig. 12 and Fig. 13.

In the mutual-healing process, the time overhead of cooperative node mainly contains two parts: *(i)* The time for the generation of response message and the verification of confirmation message, which all approximate to a constant. *(ii)* The transmission time of the respond message, which is mainly affected by the size of the containing set $BC_{Lost}$. Therefore, It can be seen that as $n$ increases, the number of updated group keys $R$ also increases, which leads to a longer delay for the cooperative node. As shown in Fig. 12. Taking the enhanced protocols as an example. when $n$ increases from 11 to 20 (corresponding to $R$ from 1 to 10), the delay of cooperative node rises from 3.4 ms to 14.8 ms.

After receiving the response message set $Res\_Set$ from the neighbors, the request node needs one and $Q$ times validation, respectively, to select a legitimate response message by the Longest-Lost-Chain mechanism in the basic protocol and enhanced one. When the $R$ and $Q$ increase, the time overhead for request node will also increase. As shown in Fig. 13, in the enhanced protocol, taking $Q = 10$ as an example, when $n$ increases from 11 to 20, the delay increases from 16.4 ms to 16.7 ms. And also taking $n = 20$ as an example, when $Q$ changes from 5 to 10, the delay increases from 19.2 ms to 35.8 ms.

It should be clear that although both the time overhead of cooperative and request nodes increase with $R$ and $Q$, the overall delay remains at the millisecond level with a low time overhead.

*c) Comparison with the existing mutual-healing schemes:* Finally, we compare the proposed scheme (Take the Enhanced version as an example) with the existing mutual-healing schemes [25]–[27]. Specifically, taking the UAV node to recover the latest lost group key $GK_n$ as an example, the time overhead of related nodes are compared. The experimental procedure is the same as described in above 2), but to be clear that in this experiment, we set $Q = 5$, $|w| = 10$. And the results are depicted in Fig. 14 and Fig. 15.

In the mutual-healing scheme, there are two stages for request node to recover the lost group key. Firstly, the request node uses the mutual-healing mechanism to obtain the lost group key broadcast $B_n$. And then, the request node decrypts $B_n$ to recover the lost group key $GK_n$. For the scheme [27] and our proposed scheme, the request node can verify the response message only through Hash and ECC operations. And also, both the group key broadcast generation algorithm are based on Chinese Remainder Theorem, which makes the node recover a lost group key only by calculating low-cost modulo operation. However, in schemes [25], [26], node needs to perform bilinear pairing operations both in obtaining and decrypting the group key broadcast message, that is very time-consuming. Therefore, as shown in Fig. 14, in

the scheme [27] and our proposal the request node has a lower delay.

Meanwhile, the time overhead of the cooperative node also includes two parts: one is the computation time for validating request message and constructing a response message, another is the transmission time of the response message. In the first part, the scheme [25] needs to perform bilinear operations, while the remaining schemes require only a small number of Hash and ECC operations. In the second part, the time overhead mainly depends on the communication overhead of cooperative node. As seen in the Table III, the number of group members, the number of group keys lost by the request node and so on will affect the communication overhead. The detailed time overhead comparison of the cooperative node is shown in Fig. 15.

It is worth noting that although our scheme increases some time overhead compared with the existing scheme (especially, the scheme [27]) in the mutual-healing process, it still has a limited time overhead. More importantly, our schemes has more strong security to resist reply attack and achieve mutual authentication.

## VII. Conclusion

The existing group key distribution schemes in the unreliable network cannot satisfy both the real-time and security for nodes to recover the lost group keys in UAANET. To address this problem, we firstly establish a private blockchain to distribute and store group keys, as well as manage the dynamic membership of UAANET. And then, based on the blockchain, a basic mutual-healing protocol and an enhanced one are designed under the different attack models to resist malicious attacks. Through security analysis and a large number of experiments, it is shown that our scheme can help the UAVs to recover the lost group keys both securely and efficiently, which means our scheme has good practical value.

## References


[1] S. Hayat, E. Yanmaz, and R. Muzaffar, "Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint," *IEEE Commun. Surv. Tut.*, vol. 18, no. 4, pp. 2624–2661, Oct. 2016.

[2] J. A. Maxa, M. S. B. Mahmoud, and N. Larrieu, "Survey on UAANET Routing protocols and network security challenges," *Ad Hoc Sensor Wireless Netw.*, 2017.

[3] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in UAV communication networks," *IEEE Commun. Surv. Tut.*, vol. 18, no. 2, pp. 1123–1152, Apr. 2016.

[4] H. Zhou, H. Kong, L. Wei, D. Creighton, and S. Nahavandi, "Efficient road detection and tracking for unmanned aerial vehicle," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 297–309, Feb. 2015.

[5] M. Erdelj, M. Król, and E. Natalizio, "Wireless sensor networks and multi-UAV systems for natural disaster management," *Comput. Netw.*, vol. 124, pp. 72–86, 2017.

[6] I. Jawhar, N. Mohamed, J. Al-Jaroodi, D. P. Agrawal, and S. Zhang, "Communication and networking of UAV-based systems: Classification and associated architectures," *J. Netw. Comput. Appl.*, vol. 84, pp. 93–108, 2017.

[7] I. Bekmezci, O. K. Sahingoz, and Ş. Temel, "Flying ad-hoc networks (FANETs): A survey," *Ad Hoc Netw.*, vol. 11, no. 3, pp. 1254-1270, 2013.

[8] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 36–42, 2016.

[9] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, and D. Dean, "Self-healing key distribution with revocation," in *Proc. IEEE Symp. Secur. Privacy*, 2002, pp. 241–257.

[10] C. Blundo, P. D'arco, A. De Santis, and M. Listo, "Design of self-healing key distribution schemes," *Des., Codes Cryptography*, vol. 32, no. 1–3, pp. 15–44, 2004.

[11] D. Liu, P. Ning, and K. Sun, "Efficient self-healing group key distribution with revocation capability," in *Proc. 10th ACM Conf. Comput. Commun. Secur.*, 2003, pp. 231–240.

[12] X. Zou and Y. S. Dai, "A robust and stateless self-healing group key management scheme," in *Proc. Int. Conf. Commun. Technol.*, 2007, pp. 1–4.

[13] X. Sun, X. Wu, C. Huang, Z. Xu, and J. Zhong, "Modified access polynomial based self-healing key management schemes with broadcast authentication and enhanced collusion resistance in wireless sensor networks," *Ad Hoc Netw.*, vol. 37, pp. 324–336, 2016.

[14] H. Guo, Y. Zheng, X. Li, Z. Li, and C. Xia, "Self-healing group key distribution protocol in wireless sensor networks for secure IoT communications," *Future Gener. Comput. Syst.*, vol. 89, pp. 713–721, 2018.

[15] G. Sáez, "On threshold self-healing key distribution schemes," in *Proc. IMA Int. Conf. Cryptography Coding*, 2005, pp. 340–354.

[16] R. Dutta, S. Mukhopadhyay, and M. Collier, "Computationally secure self-healing key distribution with revocation in wireless ad hoc networks," *Ad Hoc Netw.*, vol. 8, no. 6, pp. 597–613, 2018.

[17] M. Shi, X. Shen, Y. Jiang, and C. Lin, "Self-healing group-wise key distribution schemes with time-limited node revocation for wireless sensor networks," *IEEE Wireless Commun.*, vol. 14, no. 5, pp. 38–46, 2007.

[18] R. Dutta, E. C. Chang, and S. Mukhopadhyay, "Efficient self-healing key distribution with revocation for wireless sensor networks using one way key chains," in *Proc. Int. Conf. Appl. Cryptography Netw. Secur.*, 2007, pp. 385–400.

[19] B. Tian, S. Han, T. S. Dillon, and S. Das, "A self-healing key distribution scheme based on vector space secret sharing and one way hash chains," in *Proc. Int. Symp. World Wireless, Mobile Multimedia Netw.*, 2008, pp. 1–6.

[20] Q. Wang, H. Chen, L. Xie, and K. Wang, "One-way hash chain-based self-healing group key distribution scheme with collusion resistance capability in wireless sensor networks," *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2500–2511, 2013.

[21] H. Chen and L. Xie, "Improved one-way hash chain and revocation polynomial-based self-healing group key distribution schemes in resource-constrained wireless networks," *Sensors*, vol. 14, no. 12, pp. 24358–24380, 2014.

[22] T. Rams and P. Pacyna, "A survey of group key distribution schemes with self-healing property," *IEEE Commun. Surv. Tut.*, vol. 15, no. 2, pp. 820–842, Apr. 2013.

[23] S. Han, B. Tian, Y. Zhang, and J. Hu, "An efficient self-healing key distribution scheme with constant-size personal keys for wireless sensor networks," in *Proc. IEEE Int. Conf. Commun.*, 2010.

[24] M. J. Bohio and A. Miri, "Self-healing group key distribution," *Int. J. Netw. Secur.*, vol. 1, no. 2, pp. 110–117, 2005.

[25] B. Tian, S. Han, J. Hu, and T. Dillon, "A mutual-healing key distribution scheme in wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 80–88, 2011.

[26] S. Agrawal, J. Patel, and M. L. Das, "Pairing based mutual healing in wireless sensor networks," in *Proc. 8th Int. Conf. Commun. Syst. Netw.*, 2016, pp. 1–8.

[27] S. Agrawal and M. L. Das, "Mutual healing enabled group-key distribution protocol in wireless sensor networks," *Comput. Commun.*, vol. 112, pp. 131–140, 2017.

[28] L. Zhu, Y. Wu, K. Gai, and K. K. R. Choo, "Controllable and trustworthy blockchain-based cloud data management," *Future Gener. Comput. Syst.*, vol. 91, pp. 527–535, 2019.

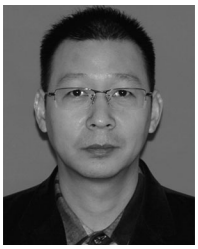
[29] K. Gai, Y. Wu, L. Zhu, L. Xu, and Y. Zhang, "Permissioned blockchain and edge computing empowered privacy-preserving smart grid networks," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7992–8004, Oct. 2019.

[30] K. Gai, Y. Wu, L. Zhu, M. Qiu, and M. Shen, "Privacy-preserving energy trading using consortium blockchain in smart grid," *IEEE Trans. Ind. Inform.*, vol. 15, no. 6, pp. 3548–3558, Jun. 2019.

[31] D. Y. Kim and J. W. Lee, "Integrated topology management in flying ad hoc networks: Topology construction and adjustment," *IEEE Access*, vol. 6, pp. 61196–61211, 2018.

[32] J. Li, M. Krohn, D. Maziéres, and D. Shasha, "Secure untrusted data repository (SUNDR)," in *Proc. 6th Conf. Symp. Operating Syst. Des. & Implementation*, vol. 4, pp. 121–136, 2004.

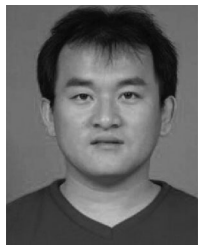
[33] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Scalable anonymous group communication in the anytrust model," in *Proc. Eur. Workshop Syst. Secur.*, 2012.
[34] J. Xie, H. Tang, T. Huang, F. R. Yu, R. Xie, J. Liu, and Y. Liu, "A Survey of blockchain technology applied to smart cities: Research issues and challenges," *IEEE Commun. Surv. Tut.*, vol. 21, no. 3, pp. 2794–2830, Jul. 2019.
[35] M. Shen, X. Tang, L. Zhu, X. Du, and M. Guizani, "Privacy-preserving support vector machine training over blockchain-based encrypted IoT data in smart cities," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7702–7712, Oct. 2019.
[36] M. Conti, E. S. Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Commun. Surv. Tut.*, vol. 20, no. 4, pp. 3416–3452, Oct. 2018.
[37] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. Annu. Tech. Conf.*, 2014, pp. 305–319.
[38] J. V. Mulert, I. Welch, and W. K. G. Seah, "Security threats and solutions in MANETs: A case study using AODV and SAODV," *J. Netw. Comput. Appl.*, vol. 35, no. 4, pp. 1249–1259, 2012.
[39] S. Karati and A. Das, "Faster batch verification of standard ECDSA signatures using summation polynomials," in *Proc. Int. Conf. Appl. Cryptography Netw. Secur.*, 2014, pp. 438–456.
[40] B. Blanchet, "Automatic verification of correspondences for security protocols," *Comput. Secur.*, vol. 17, no. 4, pp. 363–434, 2009.
[41] J. Xie, Y. Wan, J. H. Kim, S. Fu, and K. Namuduri, "A survey and analysis of mobility models for airborne networks," *IEEE Commun. Surv. Tut.*, vol. 16, no. 3, pp. 1221–1238, Jul. 2014.
[42] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," Internet Eng. Task Force. Fremont, CA, USA, 2003.
[43] J. Jiang and G. Han, "Routing protocols for unmanned aerial vehicles," *IEEE Commun. Mag.*, vol. 56, no. 1, pp. 58–63, Jan. 2018.
[44] E. Androulaki *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, 2018, Art. no. 30.
[45] P. Vijayakumar, M. Azees, A. Kannan, and L. J. Deborah, "Dual authentication and key management techniques for secure data transmission in vehicular ad Hoc networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1015–1028, Apr. 2016.
[46] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," Internet Eng. Task Force. Fremont, CA, USA, 2003.



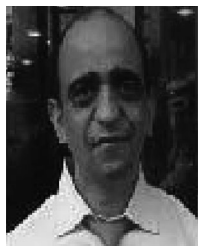
**Xinghua Li** received the M.E. and Ph.D. degrees in computer science from Xidian University, Xi'an, China, in 2004 and 2007, respectively. He is currently a Professor in the School of Cyber Engineering, Xidian University, China. His research interests include wireless networks security, privacy protection, cloud computing, and security protocol formal methodology.

**Yunwei Wang** received the B.S. degree in computer science from Shijiazhuang Tiedao University, Shaoxing, China, in 2015. He is currently working toward the Ph.D. degree in security of cyberspace with the Xidian University, Xi'an, China. His research interests include IoT, blockchain security, and privacy protection.

**Pandi Vijayakumar** received the B.E. degree in computer science and engineering from Madurai Kamaraj University, Madurai, India, in 2002, the M.E. degree in computer science and engineering from the Karunya Institute of Technology, Coimbatore, India, in 2005, and the Ph.D. degree in computer science and engineering from Anna University, Chennai, India, in 2013. He is the former Dean and is currently working as an Assistant Professor with the Department of Computer Science and Engineering at University College of Engineering, Tindivanam, which is a constituent college of Anna University, Chennai, India. His current research interests include key management in network security, VANET security, and multicasting in computer networks.



**Debiao He** received the Ph.D. degree in applied mathematics from the School of Mathematics and Statistics, Wuhan University, in 2009. He is currently a Professor with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, China. His main research interests include cryptography and information security, in particular, cryptographic protocols.



**Neeraj Kumar** received the Ph.D. degree in computer science from Shri Mata Vaishno Dev (SMVD) University, Katra, India. He was a Postdoctoral Research Fellow with Coventry University, Coventry, U.K. He is currently an Associate Professor with the Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala, India. He has published more than 200 research papers in leading journals and conferences. He is an internationally renowned researcher in the areas of VANET, CPS smart grid, IoT mobile cloud computing, big data, and cryptography.

**Jianfeng Ma** received the M. E. and Ph. D. degrees in computer software and communications engineering from Xidian University, Xi'an, China, in 1988 and 1995, respectively. He is currently a Professor in the School of Cyber Engineering, Xidian University. His research interests include information and network security, coding theory and cryptography.