Wireless Networks (2020) 26:4061–4078
https://doi.org/10.1007/s11276-020-02317-z

A centralized key management scheme for space network with resistance of nonlinear channel noise

Jie Liu[1,3] · Xiaojun Tong[1] · Zhu Wang[2] · Miao Zhang[1] · Jing Ma[4]

Published online: 31 March 2020

Abstract
The channel noise in space is nonlinear and pseudorandom so that the efficiency and security of existing group key management schemes are constrained seriously. To solve these problems, we proposed a centralized and identity-based key management scheme by using McEliece public key cryptosystem. In this scheme, the node identity is used as the parameter to generate the public key. Thus the authentication can be embedded into the verification of the public key. The group key is distributed with the protection of public key so that it can be implemented safely. Furthermore, the error correction capacity provided by McEliece public cryptosystem can eliminate the disturbance of noise. It transfers the negative influence caused by pseudorandom noise to an enhancement of security and increases the efficiency of the group key distribution over the noisy channel. The security of public key generation, forward secrecy and backward secrecy is analyzed. The performance is analyzed and compared with other schemes. The error correction capacity is simulated. The results show that our scheme can provide confidentiality, integrity, authentication, non-repudiation, failure tolerance and error correction with lower computation overhead and interaction rounds.

Keywords Centralized group key management · McEliece PKC · Identity-based public key · Pseudorandom noise · Space network

1 Introduction

As the disturbance of channel noise and openness are inherent for space network (SN) [1], the security and efficiency of SN communication are challenged hugely. To ensure integrity and confidentiality, the cryptosystem is widely used in SN communication. Various key management schemes are designed to manage the key of these cryptosystems. Generally, key management plays an important role in space network security. It provides identification, authentication, access control, key generation, key distribution and rekeying [2] for space communication.

The key management schemes presented in the literature are divided into three classes [3–6]: centralized key management, decentralized key management and distributed key management. In centralized key management, there is a centralized key distribution center (KDC) which is employed for controlling the whole management of keys. It is fully trusted to generate, distribute and update the group key. The centralized key management schemes such as Logical Key Hierarchy (LKH) [7], One-Way Function Tree (OFT) [8] and Flat Table (FT) [9] provide a higher degree of security and efficiency. The major weakness is that the dependence on KDC may cause a single-point failure. In the decentralized key management, the large group is split into small subgroups [10, 11] or multi clusters [12, 13]. Different controllers are used to manage each subgroup or cluster. The failure of one group controller will not affect the other groups since it minimizes the problem to a single cluster. The distributed key management approach is

✉ Xiaojun Tong
tong_xiaojun@163.com

1 School of Computer Science and Technology, Harbin Institute of Technology, Weihai 264209, China

2 School of Information and Electrical Engineering, Harbin Institute of Technology, Weihai 264209, China

3 School of Rongcheng, Harbin University of Science and Technology, Rongcheng 264300, China

4 Science and Technology on Information Assurance Laboratory, Beijing 100072, China

characterized by having no central controller. The group key is generated by using the threshold encryption scheme. All of the members contribute their shares to compute the group key. The distributed key management such as Group Diffie-Hellman Key Exchange (GDH) and Distributed Logical Key Hierarchy (DLKH) can avoid the single-point failure. But they need several rounds of negotiation.

Currently, some LKH-based group rekeying schemes are presented [14, 15] to reduce the consumption of rekeying by optimizing the original LKH scheme. A modification of LKH which based on one-encryption key and multi-decryption key protocol is presented in [16]. However, it takes huge computational overhead and more interaction rounds for initialization and rekeying. The key management architecture of GNSS provides authentication by using public key infrastructure (PKI) [17]. It increases complexity since it needs the support of PKI. To overcome the single-point failure problem, key management based on multi-servers is presented in [18]. But it doesn't provide identification. The verifiable group key management schemes without using PKI are present in [19, 20]. However, authentication and key distribution require many rounds of interaction, leading to significant delays in key management schemes. The decentralized key management scheme is usually put forward for the large-scale or hybrid network. Liu et al. [21] propose a hierarchical domain one-way function tree (HD-OFT)-based scheme to reduce the consumption of computation and bandwidth in LEO satellite network. In [22], the rekeying cost is reduced by clustering the scale of the satellite network. Both of them extend the tree-based key management to multi-cluster so that it can improve efficiency. But the communication between different levels needs to be translated by different root members. It increases the delay and node load. Jiao et al. [23] put forward a threshold value-based group key management for satellite networks. In the scheme, the key distribution is implemented through choosing n satellites that can keep long time visibility. The distributed key management schemes based ECC are presented in [24, 25], which using the ECC cryptosystem to provide integrity, confidentiality and verification without the requirement of key escrow. Another ECC-based key management scheme is presented by Hsiao et al. [26]. But it needs the support of PKI. The key management schemes used in wireless sensor networks are proposed in [27–29]. A chaotic map based key agreement scheme is proposed in [30]. However, most of them are unsuitable for space network since the multi-rounds of interactivity is inefficiency in the noisy and delayed environment.

In conclusion, the centralized key management schemes are more security and efficiency. However, they either don't provide authentication or provide authentication by using PKI certificates which increases the complexity. The decentralized key management schemes are less scalable and efficient since the message transmitting in different sub-groups needs to be translated. The distributed key management schemes require multiple rounds of interaction to obtain the key materials so that they increase the delay. In a word, the present group key management schemes are inefficiency because the space network is characterized by the long delay, low bandwidth and limited computational resources. In other words, the group key management scheme for space networks should be secure, reliable, less interactive and simple. To ensure the integrity and authentication of the group key, the digital signature technology (DSA) based on a public cryptosystem is employed. Because the traditional DSA requires the support of PKI, it is inefficient in space network. In this work, the node identity of the space network and McEliece public cryptosystem are employed to provide authentication and integrity. It provides the error correction capacity additionally. The McEliece cryptosystem is a public-key cryptosystem based on algebraic coding theory. A hybrid McEliece cryptosystem using pseudorandom generator is presented in [31] to enhance security. In 2020, an improved McEliece cryptosystem based on LDPC is proposed in [32]. The security is enhanced by cascading Goppa coding and LDPC coding. But they are unsuitable for the space key management as they do not provide authentication of identity and error correction capacity. Furthermore, the key size of the cryptosystem presented in [32] is much larger than the classical McEliece cryptosystem based on LDPC codes. It is infeasible to be implemented on the space node. Recently, a McEliece cryptosystem based on quasi-cyclic moderate density parity check (QC-MDPC) is proposed [33] to against all of the attacks aimed at McEliece cryptosystem while decreasing the key size. Another advantage is that the QC-MDPC-based McEliece cryptosystem is a lightweight cryptosystem [34]. For the same code length, it is more secure than the McEliece cryptosystem based on QC-LDPC. Thus we designed an identity-based group key management scheme by using the McEliece cryptosystem, which provides secure group key management and management of verifiable public key without a certificate.

The contributions of this paper are shown as follows: (1) A verifiable public key generation scheme which based on QC-MDPC code and the node ID is designed. In this scheme, the public key is verified without a certificate. (2) A group key management scheme is put forwarded by using the Hash function. The method to generate, distribute and update of the group key is designed. (3) The security of the proposed scheme is analyzed. (IV) Compared the proposed scheme with classical key management schemes in terms of storage cost, computation cost, message cost, interaction rounds and robustness.

The rest of this paper is organized as follows. The McEliece cryptosystem based on QC-MDPC is introduced in Sect. 2. The centralized identity-based key management scheme is presented in Sect. 3. In Sect. 4, the security is analyzed. The performance is analyzed and compared in Sect. 5. The paper is concluded in Sect. 6.

2 Preliminary

2.1 Construction of QC-MDPC code

We gather here a few basic definitions which are used in this paper.

Definition 1 (Quasi-cyclic code) An (n, r)-linear code is quasi-cyclic (QC) if there is some integer n_0 such that every cyclic shift of a codeword by n_0 places is again a codeword.

Definition 2 (LDPC/MDPC codes) An (n, r, w)-LDPC or MDPC code is a linear code of length n, codimension r which admits a parity-check matrix of constant row weight w.

The QC-MDPC McEliece cryptosystem was proposed to resist the attacks caused by the sparse of QC-LDPC code. The difference is that the QC-MDPC is denser than QC-LDPC, which the row weight is approximately $O(\sqrt{n \log n})$. When these codes are also quasi-cyclic, we call them QC-LDPC or QC-MDPC codes.

To achieve a flexibility code rate, we usually construct the parity check H as $H = [H_0|H_1|\cdots|H_{m-1}] \in F_2^{r\times n}$. Here, $n = mr$, $m, r \in Z^*$, r is the block size of H_i. Each block H_i has row weight w_i so that $w = \sum_0^{m-1} w_i$. Thus the generator matrix G can be computed as formula (1):

$$G = \begin{bmatrix} & \left(H_{m-1}^{-1}H_0\right)^T \\ & \left(H_{m-1}^{-1}H_1\right)^T \\ I_{(m-1)\times r} & \vdots \\ & \left(H_{m-1}^{-1}H_{m-2}\right)^T \end{bmatrix} \quad (1)$$

The construction of (n, r, w)-QC-MDPC code is as follows:

Step 1: Choose a code length n and block size r. Generate the vector $h_i \in F_2^r$ with length r and weight w/m at random. Here, $m = n/r$ is the number of blocks.
Step 2: Generate the quasi-cyclic sub-matrix $H_i \in F_2^{r\times r}$, in which the first row or column is defined by vector h_i. Other $r-1$ rows or columns of H_i are obtained from the $r-1$ quasi-cyclic shifts of h_i. For convenience, we use the vector as the first column of the quasi-cyclic sub-matrix.
Step 3: Obtain the check matrix $H = [H_0|H_1|\cdots|H_{m-1}]$.

2.2 McEliece cryptosystem based on QC-MDPC code

The McEliece cryptosystem based on QC-MDPC code consists of key generation, encryption algorithm and decryption algorithm.

1. Key generation
   Step 1: Generate a parity-check matrix $H \in F_2^{r\times n}$ of a t-error-correcting (n, r, w)-QC-MDPC code.
   Step 2: Generate the generator matrix $G \in F_2^{(n-r)\times n}$ in a row reduced echelon form by formula (1).
   The public key is G and the private key is H.
2. Enryption algorithm
   Step 1: Randomly choose an error vector $e \in F_2^n$ of Hamming weight $wt(e) \le t$. Here, $t = n/w$ is the error correction capacity, $wt(e)$ is the Hamming weight of e.
   Step 2: Compute the ciphertext as $C = MG + e$, where M is the plaintext.
3. Decryption algorithm
   Step 1: Compute MG by using the decoding algorithm.
   Step 2: Extract the plaintext M from the first $n-r$ columns of MG.

3 The identity-based key management scheme

As mentioned above, an identity-based and verifiable public key cryptosystem without using PKI is required eagerly in the space key management. Furthermore, it is better to provide tolerance of single-point failure. In this section, an identity-based key management scheme is presented which meets the requirements all above mentioned. It consists of the design of verifiable McEliece public key generation scheme, initialization of the key management scheme, node joining phase, node leaving phase and KDC election algorithm. The flow chart is shown in Fig. 1.

As shown in Fig. 1, construction of the centralized key management system for space network consists of initialization phase, node joining or leaving phase and KDC election phase. The initialization phase includes the design of verifiable McEliece public key generation scheme and group key generation scheme. The details are described below.

3.1 Design of verifiable McEliece public key generation scheme

As the QC-MDPC code-based cryptosystem is lightweight, it is suitable to be used in the space network for key management. Thus we designed a QC-MDPC code-based

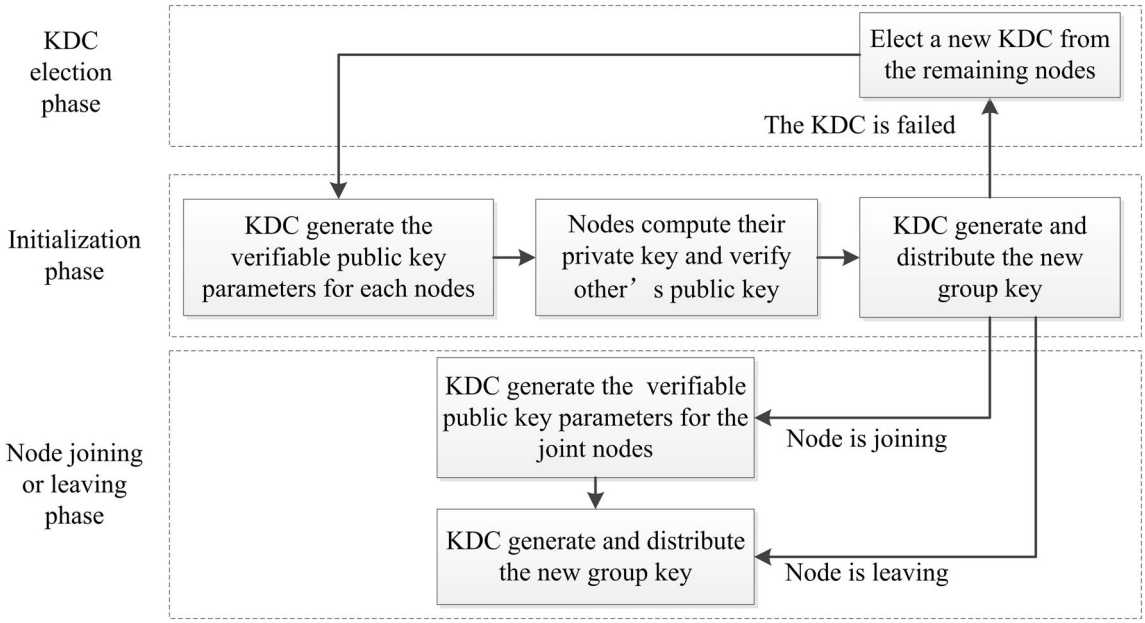

Fig. 1 The construction of centralized key management system for satellite network

public key generation scheme which embeds identity in the verification of public key so that the space nodes can authenticate each other by computing their public key. The detail is as following.

3.1.1 Generation of the public key and private key for KDC

Without loss of generality, we set $m = 2$ and $\sqrt{n \log n}/3 \le w \le \sqrt{n \log n}$. The security is increased with the increase of w while the error correction capacity is decreased with the increase of w. To achieve optimal efficiency and security, weight w can be decreased from $\sqrt{n \log n}$ to $\sqrt{n \log n}/3$ with the increasing of code length. The public key and private key of KDC is generated as following:

Step 1: The KDC randomly chooses vector h_{kdc_0}, h_{kdc_1} with weight $w/2$. Here, w is the weight of (n, r, w)-QC-MDPC code, m is the number of sub-matrix.
Step 2: The KDC constructs a check matrix $H_{kdc} = [H_{kdc_0} H_{kdc_1}]$ by using the method in Sect. 2.1.
Step 3: The KDC computes the generator matrix G_{kdc} by using formula (1) and publishes it.

3.1.2 Generation of the public key and private key for space nodes

The generation and verification of the public and private key for space nodes are as bellow:

Step 1: For each node i, the KDC randomly chooses a vector s_i with a weight of $\sqrt[3]{w/2}$ and publishes it.
Step 2: Node i generates its quasi-cycle sub-matrix S_i by s_i.
Step 3: Node i randomly chooses vector $h_{i,2}$ and $h_{i,3}$, where $wt(h_{i,2}) \le \sqrt[3]{w/2}$ and $wt(h_{i,3}) = w/2$. Then generate quasi-cycle sub-matrixes $H_{i,2}$ and $H_{i,3}$. Since the number of sub-matrix is $m = 2$, the Hamming weight of each sub-matrix should less than $w/2$.
Step 4: Node i maps its ID to a binary sequence h_{ID_i} by using formula (2). Here, the collision-free hash function is $hash : \{0,1\}^* \rightarrow \{0,1\}^*$. Then h_{ID_i} is transformed to decimal numbers which index the position of bit "1" in vector $h_{i,1}$ so that $wt(h_{i,1}) \le \sqrt[3]{w/2}$.

$$h_{ID_i} = hash(ID_i) \tag{2}$$

The matrix $H_{i,1}$ is generated by shifting $h_{i,1}$ cyclically in column. The detail is shown as formula (3):

$$H_{i,1} = \begin{bmatrix} h_{i,11} & \cdots & h_{i,12} \\ \vdots & \ddots & \vdots \\ h_{i,1r} & \cdots & h_{i,11} \end{bmatrix} \tag{3}$$

Step 5: Node i computes its check matrix H_i by formula (4). The private key is H_i, vectors $h_{i,2}$ and $h_{i,3}$ are kept secretly.

$$H_i = [H_{i,3}|H_{i,2}H_{i,1}S_i] \tag{4}$$

convenience, we use N denotes the total number of nodes. n is code length, r is block size, w is the code weight, m is block number, t is the Hamming weight of error vector e,

Algorithm 1. Generation of the public and private key for space nodes

Input: $n,\ r,\ w,\ m,\ t,\ s_i, h_{i,2},\ h_{i,3}\ ID_i$

Output: $R_i,\ G_i,\ H_i,\ G_{i_{verify}}$

/* generate the public key, private key and verifiable public key for node i*/

1 **for** $i \leftarrow 1$ **to** N

2 KDC: randomly chooses a vector s_i for node i and publishes it;

3 Node i: generates $H_{i,2} \leftarrow h_{i,2}$ and $H_{i,3} \leftarrow h_{i,3}$;

4 Node i: generates vector $h_{ID_i} \leftarrow ID_i$, computes $h_{i,1} \leftarrow h_{ID_i}$

5 Node i:generates $H_{i,1} \leftarrow \begin{bmatrix} h_{i,1_1} & \cdots & h_{i,1_r} \\ \vdots & \ddots & \vdots \\ h_{i,1_2} & \cdots & h_{i,1_1} \end{bmatrix}$;

6 Node i : computes $H_i \leftarrow \left[H_{i,3}\middle|\, H_{i,2}\, H_{i,1}S_i\,\right]$ and $G_i \leftarrow \left[I\,\middle|(S_i^{-1}H_{i,1}^{-1}H_{i,2}^{-1}H_{i,3})^T\right]$;

7 Node i: computes $R_i \leftarrow H_{i,2}^{-1}H_{i,3}$ and publishes the first column r_i;

8 KDC and other nodes : compute and verify $G_{i_{verify}}$;

9 **end for**

$$G_i = \left[I\middle|\left(\left(H_{i,2}H_{i,1}S_i\right)^{-1}H_{i,3}\right)^T\right] = \left[I\middle|\left(S_i^{-1}H_{i,1}^{-1}H_{i,2}^{-1}H_{i,3}\right)^T\right] \tag{5}$$

Step 6: Node i computes a witness value $R_i = H_{i,2}^{-1}H_{i,3}$ and publishes the first column r_i. Then node i obtains its public key G_i as formula (5):

Step 7: KDC and other nodes compute and verify the public key of node i by formula (6) and store it:

$$G_{i_{verify}} = \left[I\middle|\left(S_i^{-1}H_{i,1}^{-1}R_i\right)^T\right] \tag{6}$$

When KDC or other nodes need to transmit secret to node i, they encrypt the secret with $G_{i_{verify}}$. It's obvious that $G_{i_{verify}} = G_i$. Note that S_i, R_i and the sub-matrixes $H_{i,j}$ are invertible with overcoming probability since they are quasi cycle.

The procedure of the public and private key generation for space nodes is described in Algorithm 1. For ID_i is the identity of node i, R_i is witness value, G_i is public key of node i, H_i is private key of node i, $G_{i_{verify}}$ is the verifiable public key of node i, Vectors s_i,$h_{i,2}$, $h_{i,3}$ are randomly chosen.

3.1.3 Encryption and decryption algorithm

The message can be encrypted by the McEliece public key through formula (7):

$$C = MG + e \tag{7}$$

Here, M is the message, e is a random vector with Hamming weight $wt(e) \leq t$ and $t = n/w$.

When the ciphertext is received by nodes or KDC, they compute MG by the decoder and extract the plaintext M from the first $n - r$ columns of MG.

The example of verifiable public key generation is reviewed in Sect. 3.6.1.

3.2 Initialization of the key management scheme

There are some parameters need to be initialized before the proposed key management scheme starts to work. At the initial phase, the parameters of QC-MDPC code such as n, r, w, m and t should be set. Then the public key of KDC and space nodes are computed. Finally, the group key is generated and distributed. The procedure is described in Algorithm 2.

Algorithm 2. Initialization of the key management

Input: $n,\ r,\ w,\ m,\ t,\ s_i, h_{i,2},\ h_{i,3}\ ID_i,\ a_1,\ a_2$

Output: $R_i,\ G_i,\ H_i,\ G_{i_{verify}},\ DEK$

/* generate the verifiable public key, private key for all of the nodes, generate and distribute the initial group key */

/* DEK is the group key, a_1 and a_2 are chosen randomly*/

/* DEK_{pk_i} is the encrypted group key which using the public key of node i */

1 KDC: Publishes (n, r, w, m, t);
2 **for** $i \leftarrow 1$ **to** N
3 All of the nodes: compute and store $G_{i_{verify}}$ by using algorithm 1;
4 KDC: computes $DEK \leftarrow hash(a_1, a_2)$;
5 KDC: computes DEK_{pk_i};
6 KDC: sends DEK_{pk_i} to node i;
7 **end for**

After receiving the encrypted group key, each node decrypts it and obtains the group key DEK. The algorithm is validated in Sect. 3.6.2.

3.3 Node joining phase

When a new node i joins the network, it needs to send a request to KDC. After KDC and other members verify it, the rekeying is triggered. The procedure is described in Algorithm 3.

Algorithm 3. Node joining phase

Input: $n,\ r,\ w,\ m,\ t,\ s_i, h_{i,2},\ h_{i,3}\ ID_i,\ a_1,\ a_2$

Output: $R_i,\ G_i,\ H_i,\ G_{i_{verify}},\ DEK$

/* new node i generate its public key and private key. KDC and other nodes verify public key of node i. KDC generates and distribute the new group key */

/* DEK is the group key, a_1 and a_2 are chosen randomly*/

/* DEK_{pk_i} is the encrypted group key which using the public key of node i */

1 Node i: sends a request to KDC;
2 KDC: randomly chooses a vector s_i for node i and publishes it;
3 Node i: generates $H_{i,2} \leftarrow h_{i,2}$ and $H_{i,3} \leftarrow h_{i,3}$;
4 Node i: computes $R_i \leftarrow H_{i,2}^{-1} H_{i,3}$ and publishes the first column r_i;
5 KDC and other nodes: Compute and store $G_{i_{verify}}$;
6 **for** $i \leftarrow 1$ **to** N
7 KDC: computes $DEK \leftarrow hash(a_1, a_2)$;
8 KDC: computes DEK_{pk_i};
9 KDC: sends DEK_{pk_i} to node i;
10 **end for**

The opportunity of rekeying usually is periodic. When this is a request for joining or leaving, the rekeying operation is triggered and the timer is reset. Otherwise, the rekeying operation is implemented periodically. The example of rekeying in the joining event is reviewed in Sect. 3.6.3.

3.4 Node leaving phase

If a node is leaving, it should send a request to KDC. Then a rekeying operation is triggered. The procedure is described in Algorithm 4.

Algorithm 4. Node leaving phase

Input: $a_1,\ a_2$
Output: DEK
/* KDC generates and distributes the new group key */
/* DEK is the group key, a_1 and a_2 are chosen randomly*/

/* DEK_{pk_i} is the encrypted group key which using the public key of node i */

1 Node j: sends a request to KDC;
2 **for** $i \leftarrow 1$ **to** N and $j \neq i$
3 KDC: computes the new group key $DEK \leftarrow hash(a_1, a_2)$;
4 KDC: computes DEK_{pk_i};
5 KDC: sends DEK_{pk_i} to node i;
6 **end for**

Since the public key of any node is recomputed for different time and different group, this is no need for revocation operation in the proposed scheme. The example of rekeying in the leaving event is reviewed in Sect. 3.6.4.

3.5 KDC election algorithm

The proposed scheme is tolerant of the single-point failure by using the verifiable McEliece public key generation mechanism. When the KDC is failed, the group member can elect a new KDC. The procedure is described in Algorithm 5.

Algorithm 5. KDC election algorithm

Input: ID_i, $(\mathrm{ID}_i)_{sk_i}$

Output: node i

/* elect the new KDC from the remaining nodes */

/*ID_i is the identify of node i, $(\mathrm{ID}_i)_{pk_j}$ is the encrypted identity of node i which using public key of node j, ID_i' is the decryption of $(\mathrm{ID}_i)_{pk_j}$ which using the private key of node j */

/* num_{agree} is the number of silent nodes , N is the total number of nodes*/

1 **for** $i \leftarrow 1$ **to** N

2 Node i: sends a challenge message $(\mathrm{ID}_i, (\mathrm{ID}_i)_{pk_j})$;

3 **for** $j \leftarrow 1$ **to** N **and** $j \neq i$

4 Node j: computes $\mathrm{ID}_i' \leftarrow ((\mathrm{ID}_i)_{pk_j})_{sk_j}$;

5 **if** $\mathrm{ID}_i' = \mathrm{ID}_i$ **then** keeps silence;

6 **else**

7 Node j: broadcasts a opposed message;

8 **end if**

9 **end for**

10 **if** $num_{agree} \geq 2N/3$, **then** node i is the new KDC;

11 **else** start a new challenge;

12 **end if**

13 **end for**

By this procedure, the new KDC can be elected so that the single point failure can be avoided absolutely. The example of the KDC election is reviewed in Sect. 3.6.5.

3.6 A numerical example of the proposed scheme

In this section, the proposed scheme is validated by taking a data set as an example. The generation of verifiable QC-MDPC code-based public key, the initialization of the key management, the rekeying in joining event, the rekeying in leaving event and the KDC election algorithm are validated in this example.

3.6.1 Validation of the verifiable public key generation

To validate the output of the public key generated by using the McEliece cryptosystem based on QC-MDPC codes, a real data set is chosen to be used as an example. In order to describe the example expediently, a small code length is selected. Let $n = 32$, $m = 2$, then $w = \sqrt{nlogn} = 7$, $wt(s_i) = 1$, $wt(h_{i,1}) = 1$, $wt(h_{i,2}) = 1$, $wt(h_{i,3}) = 3$. In this section, the public key generation and private key generation of KDC and group members are reviewed by an example.

1. Public and private key generation for KDC

Suppose that KDC chooses vector $h_{KDC,1}$ and $h_{KDC,2}$ randomly, where $h_{KDC,1} = 0000010000101000$ and $h_{KDC,2} = 0000000000000010$. For convenience, the binary vector can be denoted by a generator polynomial such as

$h_{KDC,1}(x) = x^{10} + x^5 + x^3$. The private key of KDC is shown as formula (8):

$$H_{KDC} = \begin{bmatrix} 0 & 0 & & 0 & 0 & 0 & & 0 \\ 0 & 0 & & 0 & 0 & 0 & & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & & 0 & 0 & 0 & & 0 \\ 0 & 0 & & 1 & 0 & 0 & & 0 \\ 1 & 0 & & 0 & 0 & 0 & & 0 \\ \vdots & & \ddots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 1 & & 0 & 0 & 0 & & 1 \\ 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & & 0 & 0 & 1 & & 0 \end{bmatrix} \tag{8}$$

The KDC computes its public key $G_{KDC} = [I|G_{KDC_{right}}]$ by using formula (1). Here, the corresponding generator polynomial is $g_{KDC_{right}}(x) = x^{13} + x^{11} + x^6$. The detail is shown as formula (9):

$$G_{KDC} = \begin{bmatrix} 1 & 0 & & 0 & 0 & 0 & & 0 \\ 0 & 1 & & 0 & 0 & 0 & & 1 \\ 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & & 0 & 0 & 1 & & 1 \\ 0 & 0 & & 0 & 1 & 0 & & 0 \\ \vdots & & \ddots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{9}$$

2. Public and private key generation for space nodes

The public key generation for nodes of the space network is described in algorithm 1. To validate the output of steps in algorithm 1, we set the identity of the first node is $ID_2 = 1000000002$. Its hash value of identity $h_{ID_2} ='$ 7450fdf04f36 5909db63d481f90fe79a' is computed by MD5 algorithm. As the code block size is $n/m = 16$ and $wt(h_{i,1}) = 1$, we just need to map ID_2 to one decimal number which is smaller than 16. So we transform the hash values h_{ID_1} into a binary sequence and extract the first 4 bits which denote as HID_2 = 1011. Then convert it to a decimal number 7 which denotes the index of '1' in vector. Therefore, the vector generated by the identity of the node is $h_{2,1} = 0000001000000000$. For convenience, the binary vector can be denoted by a generator polynomial $h_{2,1}(x) = x^9$. By using formula (3), we can obtain the matrix $H_{2,1}$ as formula (10):

$$H_{2,1} = \begin{bmatrix} 0 & 0 & & 0 \\ 0 & 0 & & 0 \\ 0 & 0 & & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & & 0 \\ 0 & 0 & & 1 \\ 1 & 0 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{10}$$

If $H_{2,1}$ is not invertible, extract the next 4 bits from the binary sequence which transformed from h_{ID_i}. Then regenerate the matrix $H_{2,1}$ until it is invertible. Obviously, the matrix $H_{2,1}$ is invertible in this example. Suppose that node ID_2 chooses its private parameters $h_{2,2} = x^{14}$ and $h_{2,3} = x^5 + x^3 + x^2$. The parameter $s_2 = x^{11}$ is assigned by KDC. By using formula (3), we can obtain the corresponding matrixes. All of the vectors are randomly chosen and constrained by the specified weight. Furthermore, all the matrixes generated by the vector should be invertible. The node computes R_2 and publishes its witness value r_2 with the generator polynomial $r_2(x) = x^6 + x^3 + x^2$. Thus we can see that its public key is $G_2 = [I|G_{2_{right}}]$, the generator polynomial of $G_{2_{right}}$ is $g_{2_{right}}(x) = x^{14} + x^2 + x$. The detail is shown as formula (11):

$$G_2 = \begin{bmatrix} 1 & 0 & & 0 & 0 & 0 & & 1 \\ 0 & 1 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & & 0 & 0 & 1 & & 0 \\ \vdots & & \ddots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 1 & \cdots & 0 \end{bmatrix} \tag{11}$$

Its private key is $H_2 = [H_{2,3}|H_{2_{right}}]$, where the generator polynomial of $H_{2_{right}}$ is $h_{2_{right}}(x) = x^4$. The detail of H_2 is shown in formula (12):

$$H_2 = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & & \ddots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 1 & & 0 & 1 & 0 & & 0 \\ 0 & 0 & & 1 & 0 & 1 & & 0 \\ 1 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & & 0 & 0 & 0 & & 0 \\ 0 & 1 & & 0 & 0 & 0 & & 0 \end{bmatrix} \tag{12}$$

The KDC and other nodes can verify the public key of node ID_2 by the public information such as s_2, $h_{2,1}$ and r_2. They compute and store the verifiable public key $G_{2_{verify}} = [I|G_{2_{verify\_r}}]$. Here, the generator polynomial of $G_{2_{verify\_r}}$ is $g_{2_{verify\_r}}(x) = x^{14} + x^2 + x$. Obviously, $G_{2_{verify}}$ equals to G_2.

3.6.2 Validation of the initialization

In the initialization of key management, the KDC generates a new group key. The new group key is encrypted by using each nodes' public key and sent to the corresponding node. The procedure of initialization is described in Algorithm 2. The initialization of node ID_2 is validated in this section.

Let the parameters used to generate the group key are $a_1 = 1001100100110000$ and $a_2 = 1100110100010110$. Note that the length of a_1 and a_2 should better larger than 128. For convenience, the length of a_1, a_2 and group key

are set to be 16 bits in this example. The parameters a_1 and a_2 are cascaded by KDC and used as the input of MD5 algorithm. The first 16 bits of the MD5 value $GK =' 5bf1e387adf1fe44cbb52edeb4ac9614'$ is used as the group key. That is $DEK = 0101101111110001$. Then KDC encrypts DEK by using $G_{2_{verify}}$. Thus the cipher is $C = DEK_{pk_2} + e =$ 0100101101110001110000100110000. Here, the encrypted group key is $DEK_{pk_2} = (DEK * G_{2_{verify}}) mod 2$, the error vector $e = 00010000\ 10000\ 0000000100000000000$ is chosen randomly and $\mathrm{wt}(e) \leq \frac{n}{w} = 4$. After received the cipher C, node ID_2 obtains DEK_{pk_1} by using the decoding algorithm and extracts the first 16 bits. The result is 0101101111110001. Thus, the group key is distributed correctly and safely.

3.6.3 Validation of rekeying in joining event

The procedure of rekeying in joining event is described in Algorithm 3. In this section, the output of steps in Algorithm 3 is validated. When a new node is joining, the KDC randomly chooses a s_i for it. The new node sets its $h_{i,2}$ and $h_{i,3}$. Then compute its public key, private key and the witness value. The witness value r_i is published to all other nodes. Here, we set the node id is $ID_3 =' 10000000003'$, $s_3(x) = x^9$, $h_{3,2}(x) = x^4$, $h_{3,3}(x) = x^{11} + x^4 + x^2$, $h_{3,1}(x) = x^5$. Thus, we can obtain that $r_3(x) = x^{15} + x^{13} + x^6$, $G_3 = [I|G_{3_{right}}]$ with $g_{3_{right}}(x) = x^{15} + x^8 + x$, $H_3 = [H_{3,3}|H_{3_{right}}]$ with $h_{3_{right}}(x) = x^4$ and $G_{3_{verify}} = [I|G_{3_{verify\_r}}]$ with $g_{3_{verify_r}}(x) = x^{15} + x^8 + x$. The KDC chooses new parameters $a_1 = 1000000100110011$ and $a_2 = 1001110101010110$. The new group key is $DEK = 0101000110100010$. The KDC encrypts DEK by each nodes' verifiable public key and sends it to them. As the rekeying of group key for node ID_2 has been validated Sect. 3.6.2, we validate the rekeying in node ID_3 this time. The KDC encrypts the new DEK by using the public key $G_{3_{verify}}$ and a new error vector $e =$ 10000000100000000000000000000100. The received ciphertext in new node ID_3 is $C = DEK_{pk_3} + e =$ 1101000100100010000010100111001110. After the decoding, the new node obtains the encrypted group key DEK_{pk_3}. The value of DEK_{pk_3} is 0101000110100010000 1010011100110. Then node ID_3 extracts the first 16 bits of DEK_{pk_3} as the new group key. Obviously, the first 16 bits equal to the new group key DEK.

3.6.4 Validation of rekeying in leaving event

When a node is leaving, the KDC generates a new group key and sends the encrypted group key to the rest nodes. The validation of group key distribution has been described in Sects. 3.6.2 and 3.6.3.

3.6.5 Validation of KDC election algorithm

In the KDC election algorithm, the challenger encrypts its ID by each other nodes' public key and sends it to them. Suppose node ID_2 is the challenger, it encrypts ID_2 by using the public key $G_{3_{verify}}$. As the plaintext length is 16 bits, we use the first 16 bits of HID_2 as the identity of node ID_2. Set the error vector to be $e = 1100000000000000000\ 00000000000001$. Therefore, the ciphertext is $(ID_2)_{pk_3} = 1011\ 0100010100000100000101111\ 111$. Node ID_3 obtains the 16 bits plaintext '0111010001 010000' after decoding the ciphertext. Then it compares it with the first 16 bits of HID_2. The result is true, so the node ID_3 keeps salience. Normally, the results obtained by all the group members are true. Thus, node ID_2 will be the new KDC.

4 Security analysis

In this section, the security of McEliece public key generation scheme, backward and forward secrecy and collusion attack are analyzed.

4.1 Key recovery attack

In this work, we designed a verifiable public key generation scheme using McEliece cryptosystem. The core of the security is that weather the attacker can work out the private key from the public information. The public information for each node i includes s_i, r_i and $h_{i,1}$.

Theorem 1 *The probability that deduces the private key from published parameters is negligible if the block size of the generator sub-matrix is larger than 512 bits.*

Proof As the McEliece cryptosystem has been proved security with property parameters. Thus the security of public key generation scheme depends on the security of the private key. There are two approaches to obtain the private key H_i: guessing H_i directly or computing the secret vector $h_{i,2}$ and $h_{i,3}$ from known information. For the first method, the probability that guessing H_i correctly is shown in formula (13):

$$P_H = \frac{1}{\binom{n}{w}} \tag{13}$$

When $r = 160$, the probability P_H is about $2^{-198.46}$. The probability P_H increases to $2^{-471.21}$ when $r = 512$. The work fact for this method is increasing exponentially with the increase of block size r.For the second method, the attacker needs to work out $H_{i,2}$ and $H_{i,3}$ from $H_{i,2}^{-1}H_{i,3}$. Suppose that $h_{i,2}^{-1} = (x_1, x_2, \ldots, x_r)$, $h_{i,3} = (y_1, y_2, \ldots, y_r)$ and the first column of $H_{i,2}^{-1}H_{i,3}$ is $(z_1, z_2, \ldots, z_r)$. The work

Table 1 The ISD work fact for different McEliece cryptosystem

Schemes	ISD work fact	Key size (kB)
Proposed (2048,1024,40)-QC-MDPC with $t = 104$	$2^{81.82}$	0.128
Proposed (9602,4801,63)-QC-QMDPC with $t = 148$	$2^{130.13}$	0.6
Proposed (22054,11027,103)-QC-QMDPC with $t = 214$	$2^{192.12}$	1.378
LDPC-based McEliece cryptosystem with (1024,64,12,30,70,4) Ref [32]	2^{172}	30

that computes $H_{i,2}$ and $H_{i,3}$ from $H_{i,2}^{-1}H_{i,3}$ equals to solve the equation set as formula (14):

$$\begin{cases} x_1y_1 + x_2y_r + \cdots x_ry_2 = z_1 \\ x_ry_2 + x_1y_1 + \cdots x_{r-1}y_3 = z_1 \\ \vdots \\ x_2y_r + x_3y_{r-1} + \cdots x_1y_1 = z_1 \\ \vdots \\ x_1y_r + x_2y_{r-1} + \cdots x_ry_1 = z_r \\ x_ry_1 + x_1y_r + \cdots x_{r-1}y_2 = z_r \\ \vdots \\ x_2y_{r-1} + x_3y_{r-2} + \cdots x_1y_r = z_r \end{cases} \tag{14}$$

In the r equations which equal to z_i ($i \in \{1, 2, \cdots, r\}$), the addends of the left part are shifted cyclically. Thus the formula (14) can be denoted as formula (15):

$$\begin{cases} x_1y_1 + x_2y_r + \cdots x_ry_2 = z_1 \\ x_1y_2 + x_2y_1 + \cdots x_ry_3 = z_2 \\ \vdots \\ x_1y_r + x_2y_{r-1} + \cdots x_ry_1 = z_r \end{cases} \tag{15}$$

Obviously, the equation set has no solution because only z_i is known. Thus the only way to obtain $H_{i,2}$ and $H_{i,3}$ from $H_{i,2}^{-1}H_{i,3}$ is that guessing $H_{i,2}$ and $H_{i,3}$ separately. For a (n, r, w)-QC-MDPC code, $H_{i,2}$ and $H_{i,3}$ are $r \times r$ quasi-cyclic matrixes. When $m = 2$, the probability that guessing $H_{i,2}$ and $H_{i,3}$ correctly is shown as formula (16):

$$P_h = \frac{1}{\binom{r}{\sqrt[3]{w/2}}\binom{r}{w/2}} \tag{16}$$

When $w = \sqrt{n \log n}/2$ and.$r = 512$, the probability P_h is $2^{-106.40}$. The probability will decrease with the increasing of code length. It decreases to $2^{-166.06}$ when $r = 1024$. When $w = \sqrt{n \log n}$, the probability P_h increases to $2^{-177.37}$ and $2^{-267.13}$ correspondingly.Thus, in both of the two conditions, the work factor that computes the private key from known information is higher than 2^{80} when $r \geq 512$. The security level is much higher than 2^{128} when $r \geq 1024$.

4.2 Information set decoding (ISD) attack

ISD attack is a critical message recovery attack against McEliece cryptosystem based on QC-MDPC code [35]. The work fact of ISD is shown as formula (17):

$$WF = \frac{\binom{n}{t}}{\binom{n-k-l}{t-p}\binom{k+l}{p}} \tag{17}$$

Here, l and p are parameters which relate to the code length n, $t = n/w$ is the maximum weight of the error vector e. We compute the minimum work fact and key size for different McEliece cryptosystems. In the proposed scheme, we set $p = 4$ and $l = 0.013n$ so that the work fact is optimal. The results are shown in Table 1.

From Table 1, we can see that the proposed scheme is secure when the block size $r \geq 1024$. The security is improving with the increase of the code length. Though the LDPC-based McEliece cryptosystem proposed in [32] is more secure, its key size is 10–300 times that of the proposed scheme.

Therefore, the proposed scheme is more suitable for space key management. In order to achieve higher level security against both key recovery attack and ISD attack, the block size should larger than 1024 bit.

4.3 Backward secrecy and forward secrecy

Backward secrecy is used to prevent the new member from decrypting messages exchanged before it joined the group. In this work, the group key will be updated by the KDC when a new member joins. The parameters a_1 and a_2 are randomly chosen by KDC to generate the new group key. Suppose that the old group key is DEK_{old} and the new group key is DEK_{new}. The message exchanged before new member jointed denotes as $E_{DEK_{old}}(M)$. As the new member only has a new group key DEK_{new}. The decryption process of the new member is shown in formula (18):

$$M' = D_{DEK_{new}}(E_{DEK_{old}}(M)) \tag{18}$$

Here $D_{DEK_{new}}(X)$ denotes the decryption of X by using secret key DEK_{new}. It is obvious that the new member

cannot decrypt $E_{DEK_{old}}(M)$ correctly, since the properties of hash function guarantees $DEK_{new} \neq DEK_{old}$. Thus the proposed scheme keeps backward secrecy.

Forward secrecy is used to prevent the leaving member from decrypting the group's communication. In this work, the group key will be updated by KDC as soon as the member leaving. The leaving member cannot decrypt the messages because KDC does not send the new group key to it. Therefore, the proposed scheme holds backward secrecy and forward secrecy in key management.

4.4 Collusion attack

Collusion attack is that the evicted members work together and share their individual pieces of information to compute the new group key. Suppose that the leaving node u_i collects all other leaving nodes' group keys, which denote as $DEK_k, k = \{1, 2, \cdots, t-1\}$. In this work, the group key is generated by using $DEK = hash(a_1, a_2)$. Thus we can obtain formula (19):

$$\begin{cases} DEK_k = hash(a_{1k}, a_{2k}), k \in \{1, 2, \cdots, t-1\} \\ DEK_c = hash(a_{1c}, a_{2c}) \end{cases} \tag{19}$$

Here, DEK_c is the new group key, the parameters a_{1c} and a_{2c} are randomly chosen by KDC. From the properties of hash function, we can obtain formula (20):

$$DEK_c \neq DEK_k, k \in \{1, 2, \cdots, t-1\} \tag{20}$$

The old group keys DEK_k are useless to compute the new group key DEK_c. Thus the proposed scheme can resist collusion attack.

5 Efficiency analysis

In this section, we compared our scheme with four other key management schemes in terms of storage, computation cost, communication cost, interaction rounds and robustness. The four key management schemes are LKH [8], AGKM [16], the scheme proposed in [19] which is denoted as CGKEP and the scheme proposed in [25] which is denoted as DEKM. Furthermore, we simulated the error correction capacity of the proposed group key distribution scheme over a noisy channel. In order to facilitate comparison, some parameters are given below: K denotes a unit of key and key material, N denotes the number of the group member.

5.1 Storage cost

In the LKH scheme, the KDC keeps all of the logic keys while each member keeps keys from the leaf node to the root node in LKH tree. Thus the storage cost of KDC is $(2N-1)K$. Each member needs to store $(\log_2 N + 1)K$ keys. The storage cost of the root member in the AGKM scheme is $(2N-1)K$ while the leaf member is $(\log_2 N + 1)K$. In the CGKEP scheme, each node needs to store a master share and a pairwise shared key. So the storage cost is $2K$. The DEKM scheme is a fully distributed key management. The initiator will exit and delete all of the key materials after the initialization of the network. Hence the remaining nodes need to store their public–private key pairs, witness value and pairwise key. The storage cost is $4K$. In this work, in order to verify the other nodes and elect new KDC, all of the nodes need to store its public–private key pairs, the group key and $(N-1)$ other nodes' public keys. So the storage cost of the proposed scheme is $(N+3)K$. The comparison is shown in Table 2.

Table 2 Comparison of storage cost for different key management schemes

Scheme	KDC	Member
Proposed scheme	$(N+3)K$	$(N+3)K$
LKH Ref [8]	$(2N-1)K$	$(\log_2 N+1)K$
AGKM Ref [16]	$(2N-1)K$	$(\log_2 N+1)K$
CGKEP Ref [19]	–	$2K$
DEKM Ref [25]	–	$4K$

As shown in Table 2, the storage cost of the proposed scheme higher than the decentralized key management schemes. However, the character of centralized key management determines that its storage cost is higher than the decentralized key management. The storage cost of the proposed scheme is lower than the traditional centralized key management schemes such as LKH and AGKM. Furthermore, it decreases the storage cost of KDC as much as half of the original.

5.2 Computation cost

In the LKH scheme, in initialization, the KDC needs to compute $2N-1$ keys for logistical and terminal nodes. The KDC needs $3\log_2 N$ encryption operations and $\log_2 N$ Hash operations when a member is joining. The members need $\log_2 N + 1$ decryption operations. When a member is leaving, the KDC needs to encrypt $2\log_2 N$ keys for rekeying whiling the members need only $\log_2 N$ decryption operations. In the AGKM scheme, the new member needs to do $2(N-2)$ modular exponentiations in the member joining event. When a member is leaving, all the non-leaving members need to update the public keys from the leaf node to the root node. The maximal number of modular exponentiation in this event is $3N - \log_2 N - 4$. In the CGKEP scheme, the computation cost of all members in

Table 3 Comparison of computation cost for different key management schemes

Scheme	Initialization		Joining		Leaving	
	KDC	Member	KDC	Member	KDC	Member
Proposed Scheme	$N+2$	$N+2$	$N+3$	2	N	–
LKH Ref [8]	$2N-1$	–	$4\log_2 N$	$\log_2 N+1$	$2\log_2 N$	$\log_2 N$
AGKM Ref [16]	–	–	$2(N-2)$	$2(N-2)$	$3N-\log_2 N-4$	$3N-\log_2 N-4$
CGKEP Ref [19]	$N+3$	$N+3$	$N+2$	$N+3$	$N+2$	–
DEKM Ref [25]	$3N$	$N+1$	–	3	–	–

initialization is $N+3$. In the member joining event, the computation cost of each member is $N+3$. The initiator needs $N+2$ operations to generate the group key and encrypts it. When a member is leaving, only the initiator needs to compute a new group key and sends it to each group. Its computation cost is $N+2$. In the DEKM scheme, the initiator needs $3N$ operations in initialization. Each member needs $N+1$ operations to compute a witness value and verify the other members' public key. In a member joining event, the computation cost of the new member is 3. As this scheme is designed to support point to point communication, there is no need to consider the member leaving. In this work, KDC and each member need to compute the witness value, public–private key pairs of itself and the public key of other members in the initialization. The computation cost is $N+2$. When a new member is joining, the KDC needs to encrypt the new group key with each member's public key and send it to them. Both KDC and other members need to compute the new member's public key and verify it. So the computation cost of KDC is $N+3$. The computation cost of each member is 2. When a member is leaving, the KDC needs to generate a new group key. Then encrypt the new group key with each member's public key and send it to them. It's computation cost is N. The comparison is shown in Table 3.

The total computation cost of the proposed scheme is compared with other schemes. The number of nodes is changing from 1 to 500. For convenience, the computation cost is transformed into logarithmic form base 2. The result is shown in Fig. 2.

From Fig. 2 we can see that the total computation cost of the proposed scheme is lower than AGKM and CGKEP but higher than LKH. It approximately equals to the computation cost of DEKM. This is because both of them provide public key generation and verification. Though the LKH takes lower computation cost, it neither generates a public key for members nor provides verification.

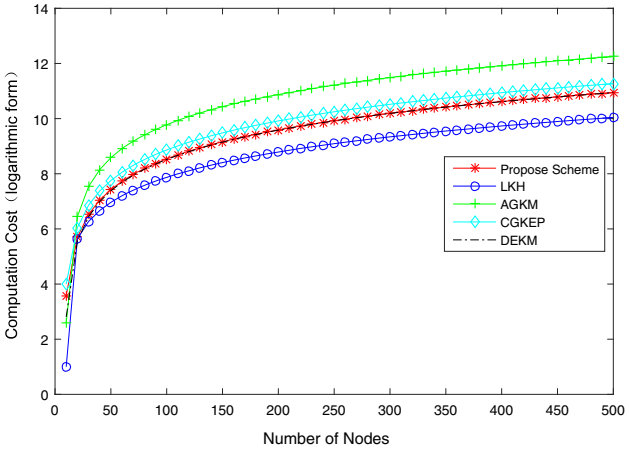

Fig. 2 Comparison of total computation cost

5.3 Message cost

In the LKH scheme, in initialization, the KDC needs to send N keys to terminal nodes. When a member is joining, the new keys must be sent to the members by using $O(2\log_2 N)$ multicasts and $O(\log_2 N)$ unicasts. So it's message cost is $O(3\log_2 N)$. In the leaving event, there are $O(2\log_2 N)$ keys need to be updated. So its message cost is $O(2\log_2 N)$. In the AGKM scheme, in the initialization, each member needs to send a secret value to KMC and the KMC issues a public encryption key set with size of $O(N)$. The total number of message is $O(2N)$. There are $O(\log_2 N)$ keys need to be updated in the member joining event. So its message cost is $O(\log_2 N)$. When a member is leaving, it only needs to send its decryption key to other members. Thus its message cost is $O(\log_2 N)$. In the CGKEP scheme, each member needs to generate sub-shares for other members and publish a witness value in initialization. Also, the initiator needs to send a group key to the members of the sub-group in which there are l members. So its message cost is $O(N^2+l)$. In the member joining event, the new member and the other members need to register with each other. Then the initiator sends the new group key to each group member. So its message cost is $O(3N+l)$. When a member is leaving, only the

initiator needs to send the new group key to each group member. Its message cost is $O(l)$. In the DEKM scheme, the message cost of the initiator is $O(2N)$ in initialization. Each member needs to send $O(N)$ messages to verify their public key. In the joining event, the new node needs to send $O(3t)$ messages to obtain t master shares and t partial private key shares from its neighbors. In this work, the DKC publishes each member's public key parameter and sends the group key to them in initialization. Each member publishes its witness value so that it can be verified. So the total message cost is $O(3N)$. In member joining event, the KDC sends a random vector to the new member and the new member publish its witness value. After verification, the KDC sends the new group key to each member. Its message cost is $O(N)$. When a member is leaving, the KDC only needs to send new group key to each member. So the message cost is $O(N)$. The comparison of the message cost is shown in Table 4.

The comparison of messages cost for different schemes is shown in Fig. 3.

The result shown in Fig. 3 implies that the message cost of the proposed scheme is higher than classical centralized schemes but lower than the decentralized schemes. The reason is that the proposed scheme takes a large number of messages to generate and verify the public key. However, the traditional centralized schemes such as LKH and AGKM only provide group key management. Their security of group key distribution is granted by the extra cryptosystem. So the proposed scheme achieves higher robustness and security by sacrificing the message cost.

To justify the efficiency analysis, we compared the proposed scheme with other key management schemes by the specified space network scenario. Let $N = 100$ and $K = 1\ KB$, $l = 10$ and $t = 2N/3$. The result is shown in Table 5.

From Table 5, we can see that the storage overhead of the proposed scheme is lower than traditional centralized key management schemes and its message cost is lower than distributed key management. Though the computation cost and message cost of the proposed scheme are higher than LKH, it provides authentication, public key management, group key management, error correction and KDC election. While the LKH, AGKM and CGKEP only provide group key management.

Table 4 Comparison of the message cost for different key management schemes

Scheme	Initialization	Joining	Leaving
Proposed scheme	$O(3N)$	$O(N)$	$O(N)$
LKH Ref [8]	$O(N)$	$O(3\log_2 N)$	$O(2\log_2 N)$
AGKM Ref [16]	$O(2N)$	$O(\log_2 N)$	$O(\log_2 N)$
CGKEP Ref [19]	$O(N^2 + l)$	$O(3N + l)$	$O(l)$
DEKM Ref [25]	$O(3N)$	$O(3t)$	-

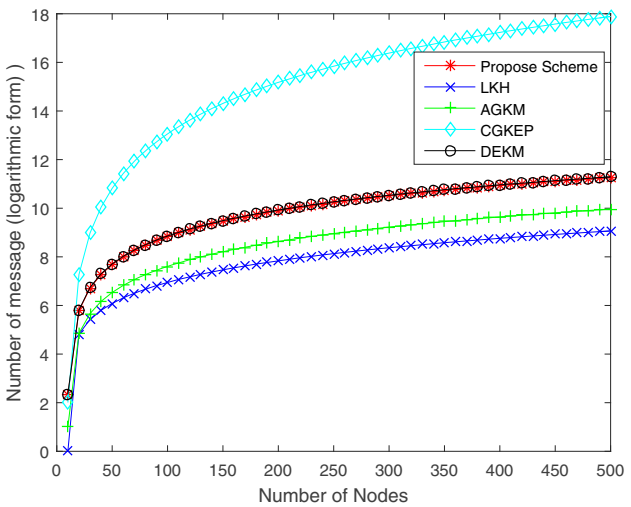

Fig. 3 Comparison of total message cost

5.4 Interaction round and robustness

The interaction round is another indicator to estimate the key management scheme. In the LKH scheme, only one round of interaction is needed in both joining and leaving event. In the AGKM scheme, one round of interaction is needed in all phases of initialization, member joining and member leaving event. In the CGKEP scheme, at the initialization stage, it needs one round of interaction in GMS, two rounds of interaction in VMS and one round of interaction in TGK. When a member is joining or leaving, it needs only one round of interaction to transmit the secret group key. In the DEKM scheme, it needs four rounds to initialize the scheme. When a member is joining, it needs two rounds to complete the verification and key generation for the new member. In this work, only one round is needed in the initialization, joining event and leaving event. The DEKM scheme and the proposed scheme are robust against single-point failures, while other schemes are vulnerable. The comparison is shown in Table 6.

From Table 6 we can see that the proposed scheme provides tolerance of single-point failure with lower interaction rounds. Though LKH and AGKM have lower interaction rounds, they are unavailable if the KDC is failed.

5.5 Error correction capacity

The group key is usually encrypted to guarantee security during its distribution. In this work, we encrypted the group

Table 5 Comparison of the efficiency for different key management schemes

Scheme	Storage cost (KB)		Computation cost (operation)			Message cost (KB)		
	KDC	Member	Initialization	Joining	Leaving	Initialization	Joining	Leaving
Proposed scheme	103	103	204	105	100	300	100	100
LKH Ref [8]	199	7	199	36	21	100	21	14
AGKM Ref [16]	199	7	–	392	578	200	7	7
CGKEP Ref [19]	–	2	206	205	102	10010	310	10
DEKM Ref [25]	–	4	401	3	–	300	200	

Table 6 Comparison of interaction round and robustness

Scheme	Interaction round			Tolerance of single-point failure
	Initialization	Joining	Leaving	
Proposed scheme	1	1	1	Y
LKH Ref [8]		1	1	N
AGKM Ref [16]	1	1	1	N
CGKEP Ref [19]	4	1	1	N
DEKM Ref [25]	4	2	0	Y

key by using McEilece cryptosystem based on QC-MDPC code. We simulated the error correction capacity of the proposed group key distribution scheme over the noisy channel, in which the group key is encrypted by the McEliece cryptosystem based on $(2048, 1024, 40)$ QC-MDPC. The channel type is binary additive white Gaussian noise (AWGN) channel and the noise power $P_{noise} = P_{sig}/SNR$. Here, P_{sig} is the signal power, SNR is the Signal to noise ratio. SHA-3 is used to map the identity to h_{ID_i}. $H_{i,1}$ is generated by using formula (3). The other parameters such as $h_{i,2}$, $h_{i,3}$ and s_i used in the proposed scheme are generated randomly. The Hamming weight of the generated QC-MDPC code is 40. Furthermore, we compared the bit error rate (BER) of the proposed scheme with the key management scheme which encrypted the group key by using AES. The result is shown in Fig. 4.

From Fig. 4, we can see that the BER of the proposed scheme reduced to 0 when the single noise rate (SNR) is higher than 5 while the BER of AES is always approximately 0.5. As the message is binary, a BER equals to 0.5 implies that the decoded message is always incorrect. This means that the proposed scheme can correct all of the errors in the received keys when $SNR \geq 5$, while the classical key management always obtains error keys. Furthermore, the error correction capacity is increasing with the increase of code length. Thus the proposed scheme can resistant channel noise and improve the success of key distribution greatly.

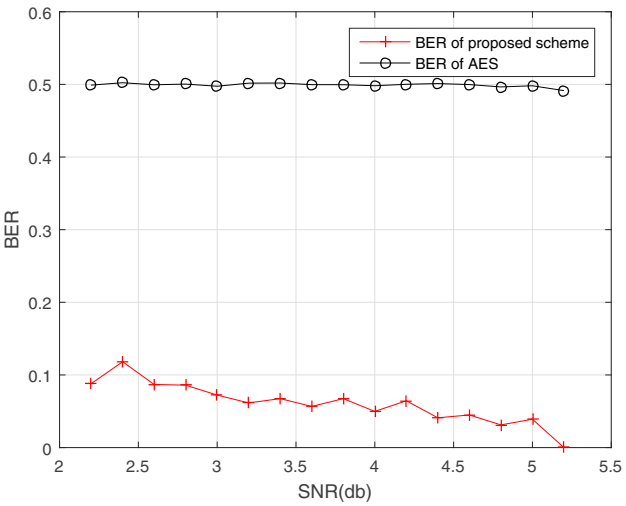

Fig. 4 Comparison of BER of key distribution schemes over noisy channel

From the above, we can conclude that the proposed scheme provides high security and robustness with lower computation cost and rounds. As a centralized key management scheme, the storage of our scheme is lower than

the classical centralized key management schemes such as LKH and AGKM. Though the computation cost and message cost of the proposed scheme are higher than LKH-based schemes, it is capable of KDC failure tolerance which overcomes the critical flaw of centralized key management schemes. Secondly, it provides authentication, public key management and group key management simultaneously while the traditional schemes provide only group key management. Furthermore, it can correct the errors caused by the noise so that the efficiency of key distribution is improved vastly. Thus the proposed scheme has a significant advantage against the other centralized key management schemes in space networks.

6 Conclusions

In this work, we proposed a novel centralized identity-based key management scheme by using McEliece public key cryptosystem for space network to resist the disturbance of nonlinear channel noise. The KDC generates each member's public and private key by using McEliece public key cryptosystem. The group key is encrypted by the node's public key to ensure both security and reliability. The identity of space nodes is used as a partial parameter of the nodes' public key to provide authentication. The hash function is employed to keep the forward and backward secrecy of the group key management. The elect mechanism is designed so that the robustness of the proposed key management is enhanced drastically. It can overcome the critical flaw that the traditional centralized key management schemes are unavailable when the KDC is failed. Furthermore, the pseudorandom noise of the channel can be transferred to enhance security. The comparison of the proposed scheme with other schemes shows that our scheme has higher security and robustness, lowest storage cost and lower interaction rounds. Thus the proposed key management scheme is suitable in space networks for its excellent security, reliability and efficiency.

Acknowledgements This research was funded by the following projects and foundations: Project ZR2019MF054 supported by Shandong Provincial Natural Science Foundation, the Foundation of Science and Technology on Information Assurance Laboratory(KJ-17-004), Equip Pre-research Projects of 2018 supported by Foundation of China Academy of Space Technology (WT-TXYY/WLZDFHJY003), the Fundamental Research Funds for the Central Universities (HIT.NSRIF.2020099), National Natural Science Foundation of China (61902091), 2017 Weihai University Co-construction Project, the engineering technology and research center of weihai information security.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Jiang, C., Wang, X., Wang, J., Chen, H. H., & Ren, Y. (2015). Security in space information networks. *IEEE Communications Magazine, 53*(8), 82–88.
2. McDaniel, P., Prakash, A., & Honeyman, P. (1999). Antigone: A flexible framework for secure group communication. Center for Information Technology Integration.
3. Rafaeli, S., & Hutchison, D. (2003). A survey of key management for secure group communication. *ACM Computing Surveys (CSUR), 35*(3), 309–329.
4. Merwe, J. V. D., Dawoud, D., & McDonald, S. (2007). A survey on peer-to-peer key management for mobile ad hoc networks. *ACM Computing Surveys (CSUR), 39*(1), 1-es.
5. Challal, Y., & Seba, H. (2005). Group key management protocols: A novel taxonomy. *International Journal of Information Technology, 2*(1), 105–118.
6. Rani, T. P., & Kumar, C. J. (2012). Survey on key pre distribution for security in wireless sensor networks. In *International conference on computer science and information technology* (pp. 248–252). Springer.
7. Wong, C. K., Gouda, M., & Lam, S. S. (2000). Secure group communications using key graphs. *IEEE/ACM Transactions on Networking, 8*(1), 16–30.
8. Sherman, A. T., & McGrew, D. A. (2003). Key establishment in large dynamic groups using one-way function trees. *IEEE Transactions on Software Engineering, 29*(5), 444–458.
9. Waldvogel, M., Caronni, G., Sun, D., Weiler, N., & Plattner, B. (1999). The VersaKey framework: Versatile group key management. *IEEE Journal on Selected Areas in Communications, 17*(9), 1614–1631.
10. Li, D., Zhang, R., & WAng, C. (2012). Efficient group key management scheme with hierarchy structure. *Chinese Journal of Electronics, 21*(2).
11. Seba, H., Lagraa, S., & Kheddouci, H. (2012). Alliance-based clustering scheme for group key management in mobile ad hoc networks. *The Journal of Supercomputing, 61*(3), 481–501.
12. Son, J. H., Lee, J. S., & Seo, S. W. (2010). Topological key hierarchy for energy-efficient group key management in wireless sensor networks. *Wireless Personal Communications, 52*(2), 359.
13. Konstantinou, E. (2011). Efficient cluster-based group key agreement protocols for wireless ad hoc networks. *Journal of Network and Computer Applications, 34*(1), 384–393.
14. Arslan, M. G., & Alagoz, F. (2006). Security issues and performance study of key management techniques over satellite links. In *2006 11th international workshop on computer-aided modeling, analysis and design of communication links and networks* (pp. 122–128). IEEE.
15. Ahmad, K., Bakhache, B., El Assad, S., & Sindian, S. (2012). A scalable key management scheme for secure IP multicast over DVB-S using chaos. In *2012 16th IEEE mediterranean electrotechnical conference* (pp. 736–740). IEEE.
16. Zhou, J., Song, M., Song, J., Zhou, X. W., & Sun, L. (2014). Autonomic group key management in deep space DTN. *Wireless Personal Communications, 77*(1), 269–287.
17. Caparra, G., Ceccato, S., Sturaro, S., & Laurenti, N. (2017). A key management architecture for GNSS open service Navigation Message Authentication. In *2017 European navigation conference (ENC)* (pp. 287–297). IEEE.
18. Sureshkumar, V., Amin, R., & Anitha, R. (2017). An enhanced bilinear pairing based authenticated key agreement protocol for multiserver environment. *International Journal of Communication Systems, 30*(17), e3358.

19. Harn, L., Hsu, C. F., & Li, B. (2018). Centralized group key establishment protocol without a mutually trusted third party. *Mobile Networks and Applications, 23*(5), 1132–1140.
20. Elmasri, M. H., Megahed, M. H., & Elazeem, M. H. A. (2016). Design and software implementation of new high performance group key management algorithm for tactical satellite. In *2016 33rd National radio science conference (NRSC)* (pp. 149–158). IEEE.
21. Qian, L., Ningning, S., & Wenlu, Z. (2013). Research of centralized multicast key management for LEO satellite networks.
22. Roy-Chowdhury, A., & Baras, J. S. (2004). Key management for secure multicast in hybrid satellite networks. In *IFIP international information security conference* (pp. 533–548). Springer.
23. Jiao, W., Hu, J., Lu, Z., & Xu, J. (2013). A threshold value-based group key management for satellite network. In 2013 IEEE Third International Conference on Information Science and Technology (ICIST) (pp. 718-721). IEEE.
24. Liu, Y., Zhang, A., Li, J., & Wu, J. (2016). An anonymous distributed key management system based on CL-PKC for space information network. In *2016 IEEE international conference on communications (ICC)* (pp. 1–7). IEEE.
25. Gharib, M., Moradlou, Z., Doostari, M. A., & Movaghar, A. (2017). Fully distributed ECC-based key management for mobile ad hoc networks. *Computer Networks, 113,* 269–283.
26. Zhang, Q., Yuan, J., Guo, G., Gan, Y., & Zhang, J. (2018). An authentication key establish protocol for WSNs based on combined key. *Wireless Personal Communications, 99*(1), 95–110.
27. Lavanya, S., & Usha, M. (2018). Dynamic key management in heterogeneous wireless sensor networks based on residual energy. In *International conference on mobile and wireless technology* (pp. 59–68). Springer.
28. Fakhrey, H., Johnston, M., Angelini, F., & Tiwari, R. (2018). The optimum design of location-dependent key management protocol for a multiple sink WSN using a random selected cell reporter. *IEEE Sensors Journal, 18*(24), 10163–10173.
29. Hsiao, T. C., Chen, T. L., Chen, T. S., & Chung, Y. F. (2019). Elliptic curve cryptosystems-based date-constrained hierarchical key management scheme in internet of things. *Sensors and Materials, 31*(2), 355–364.
30. Gong, P., Li, P., & Shi, W. (2012). A secure chaotic maps-based key agreement protocol without using smart cards. *Nonlinear Dynamics, 70*(4), 2401–2406.
31. Zajac, P. (2019). Hybrid encryption from McEliece cryptosystem with pseudo-random error vector. *Fundamenta Informaticae, 169*(4), 345–360.
32. Branco, P., Mateus, P., Salema, C., & Souto, A. (2020). Using low-density parity-check codes to improve the McEliece cryptosystem. *Information Sciences, 510,* 243–255.
33. Misoczki, R., Tillich, J. P., Sendrier, N., & Barreto, P. S. (2013). MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In *2013 IEEE international symposium on information theory* (pp. 2069–2073). IEEE.
34. Von Maurich, I., & Güneysu, T. (2014). Lightweight code-based cryptography: QC-MDPC McEliece encryption on reconfigurable devices. In *2014 Design, automation and test in Europe conference and exhibition (date)* (pp. 1–6). IEEE.
35. Becker, A., Joux, A., May, A., & Meurer, A. (2012). Decoding random binary linear codes in 2 n/20: How 1 + 1 = 0 improves information set decoding. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 520–536). Springer.

Jie Liu received the B.Eng. degree from the Department of Information Security, PLA Information Engineering University in 2003. He received the Master of Engineering degree from the Department of Computer Software and Theory, Harbin University of Science And Technology in 2007. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Harbin Institute of Technology. Since 2012, he has been a lecturer with the Department of Software Engineering, Harbin University of Science and Technology. His research interests include chaos cryptosystem, code-based cryptosystem and key management.

Xiaojun Tong is a professor and Ph.D. supervisor at Harbin Institute of Technology. Her current research interests are in the areas of chaos cryptography and information security. She is a member of the Program Committee for International Workshop on Chaosfractals Theories and Applications.

Zhu Wang is a professor at Harbin Institute of Technology. His current research interests are Wireless sensor network and networking technology and single processing.

Miao Zhang is a lecturer at Harbin Institute of Technology. From 2004 to 2006, she was an assistant at Harbin Institute of Technology (Weihai). Since 2006, she has been a lecturer in the Harbin Institute of Technology (Weihai). Her current research interests are in the areas of chaos cryptography, information security and image processing.

Jing Ma is an engineer of Science and Technology on Information Assurance Laboratory. Her interest is information security.