TELKOMNIKA, Vol.11, No.2, February 2013, pp. 809~818
ISSN: 2302-4046

Key Agreement Procotol in DSN

Jian Zhou*[1,2], Xianwei Zhou[1]
[1]School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, China, 100083
[2]School of Management Science and Engineer, Anhui University of Finance and Economics, Anhui, china, 233041
*Corresponding author, e-mail: ac_zj_course@163.com

Abstract

In space networks, the long time delay, large scale and maintainability of difficultly nature makes key management more difficult than ground wireless networks. The main challenge is how to handle 1-affects-n problem that become more serious as entities spread over a wide geographic area. To solve it, this paper proposes a one-to-many mapping shared key agreement, which is based on one-to-many encryption mechanism model. In the proposed key agreement, each entity has different decryption key and shares an encryption key. When an entity joins or leaves network, updated keys only are public encryption key and its decryption key. However, the other entities' secret key remains unchanged, so as to each member has the ability to update key autonomously. Consequently the performance of the proposed key management scheme is unrelated to the network scale, node mobility and topology structure. It is shown that our proposed key management scheme not only improves the efficiency and flexibility for space networks, but also achieves good security properties, including forward security and backward security and many more by theoretical analyses.

Keywords*: Space Network, Key Management, Rekeying, 1-affects-n Problem, Security*

1. Introduction

Similar to the ground wireless networks, space networks [1] also need key management technologies to protect the network with the quick development of space technologies and wireless network [2], key management is a pressing issue for space network security. Therefore, it is very necessary to adopt appropriate key management schemes to guarantee network security [3]. However, space entities spread over a wide geographic area, so it is difficult to build the powerful on-line key manage center to implement key management. Contrary to ground wireless networks, space networks have some distinguishing features. Firstly, the space network environment is more complex than ground, such as the channel is easily interfered so as to the bit error rates of space channel is high, so the communication load and time delay is strictly restricted in designing security protocol, in other words reducing the round complexity and improving rekeying efficiency are important targets in designing key management scheme for space network. Secondly, in view of wireless and mobility, similar to ground wireless entities space hardware level is also limited including hardware size, hardware weight, energy, computational capability and memory and so on. However, space entities have stronger performance on above capabilities than ground wireless entities. For instance, many space entities' size and weight are bigger than ground wireless entities, which also allow space entity to equip with big memory and powerful processor [4, 5]. And most of space entities have solar batteries, so entities in space networks not only have longer life-time, but also support more complex operation than ground wireless networks. At last, like ground wireless network all entities organizes into a dynamitic group in space network, so space entities have the ability of joining in or leaving space network. However, space entities movement trend can be forecast reasonably, such as many satellites and aircrafts run in orbit under the law of celestial mechanics. According to the regularity, space entities periodic time is an advantage for designing key management scheme. Therefore, traditional key management solutions cannot be directly applied in space networks.

Received August 5, 2012; Revised December 28, 2012; Accepted January 13, 2013

2. Related Work

In the literature, the existing key distribution and key establishment can be divided into four classes as follows: (1) Centralized key management. There is an entity having powerful capability and covering with the whole network, so it can support the key management in time and efficiency including generating and distributing keys, Such as GKMP [6] and Secure Lock [7] The powerful entities is responsible for key management as a whole and rekey happens between update member and key manage server, so 1-affects-*n* problem is not exist; (2) Contributory key management. In public channel, all legitimate members contribute their shares in a round of message exchanges to generate a common key, those schemes target is to reduce the complexity of round and calculation, such as Ingemarson's scheme [8], GDH [9], Octopus [10] and BD [11]. When a member leaves or joins, all members need to update their shared key, which results in 1-affects-*n* problem [12], so the schemes are suitable to the small-scale network; (3) Key management based on some especially topology structure. Those schemes deal with the efficiency question in large scale network with especially topology structures [13]. For instance, in the literature, [14,15,16] suggested key management schemes based on tree structure, such as STR [17], DH-LKH [18] and LKH [17]; and [20] suggested key management schemes based on cluster. These schemes alleviate 1-affects-*n* problem, but the shortage of these schemes include limited mobility and the cluster head and root are the bottleneck of these protocols; (4) Pre-configuration key management, member can get a key or some keys in advance. Such as, kronos [21] is put forwards to distribute single common secret key for members in advances, the next periodic time key is generated by the last periodic time key. However, in scheme [22] network members select part of keys into alternative key set from key pool which is built by the off-line key management center (KMC). To correctly establish a secure channel, communication entities share a common key in alternative key set. Members can move freely at random and spend less time in agreeing a shared key. But alternative key set should have enough keys to share a common key with a high probability. Therefore, rekeying becomes a difficult question as the updated key is shared by a lot of members and those members locations are unknown to the KMC, thus in the pre-configuration key management scheme, it is difficult to guarantee forward security and backward security, and 1-affects-*n* problem is an pressing issue in these schemes for space networks. Because currently space key management schemes is based on above ground key management, such as [23] is based on LKH protocol, [24] is the combine of LKH and GDH. These limiting factors motivate the need for designing autonomous and secure key management schemes without 1-affects-*n* problem for space networks.

3. Autonomous Shared Key Management in Space Networks (AKMSN)

In AKMSN, there are three kinds of entities: legitimate entity u_i $i \in \{1,2,...,n\}$, bulletin board B and key management server C. Each legitimate entity has a common encryption key and a different decryption key, bulletin board B has enough memory to publish information on encryption key from legitimate members and the C , all information written in B is public for any entity in space networks, the C 's capability is limited, it can not cover the whole network, its management range is only overlap Bulletin board B, meanwhile it is absolute security to any legitimate member.

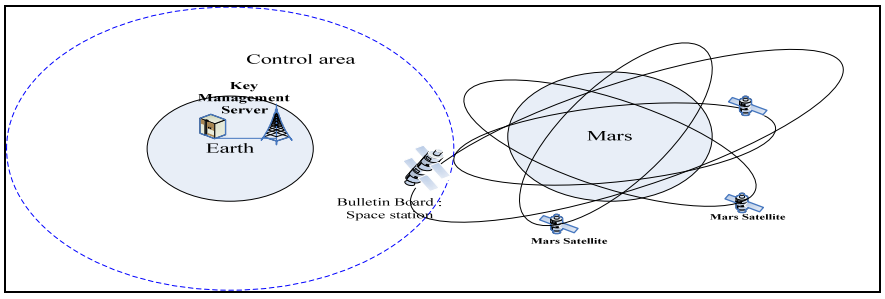

Figure 1. An Example of Space Networks

For example, as shown in Figure 1, the legitimate entities are satellites, they are composed into satellite network around mars, the key management server is built on earth whose control area is limited and difficult to overlap mars, the bulletin board is space station. Assume that the proposed protocol is based on the Diffie-Hellman (DH) protocol and all legitimate members select F_p^* as finite group and g is a the generator of F_p^*. The AKMSN scheme includes key agreement phase, key use phase and rekey phase.

3.1 Key Agreement Phase

In this phase, member $u_i \ \ i \in \{1,2,...,n\}$ agrees the public encryption key and secret decryption key with the C, and each member tell C the periodic motion that member comes into the C control area next time.

Step1: member $u_i \ \ i \in \{1,2,...,n\}$ selects the random value $x_i \ \ i \in \{1,2,...,n\}$ as the decryption key from the domain $[1, p-2]$, and send it to C with a security channel, the C keeps it secretly;

Step2: After the C gathers all legitimate members decryption key as the key set $\{x_i\}$, it selects the random value $t = p_0$ from $[1, p-2]$, it computes $P_0 = g^t \bmod p$ and P_i $i \in \{1,2,...,n\}$, which is shown as follows:

$$\begin{cases} P_1 = g^{p_1} \bmod p \ = g^{t(x_1+x_2+x_3...+x_n)} \bmod p \\ P_2 = g^{p_2} \bmod p \ = g^{t(x_1x_2+x_1x_3+...+x_1x_n+x_2x_3+...+x_{n-1}x_n)} \bmod p \\ \qquad\qquad ..... \\ P_n = g^{p_n} \bmod p \ = g^{t(x_1x_2...x_i...x_n)} \bmod p \end{cases} \tag{1}$$

Step3: The C issues the public encryption key $eKey = \{P_i\} \ \ i \in \{1,2,...,n\}$ in the bulletin board B, each encryption key has a launch time;

Step4: The C uses the value belonging to set $\{x_i\}$ to compute $P_0^i = g^{x_i t} \bmod p$, and publishes the value of $P_0^i \ \ i \in \{1,2,...,n\}$ in the bulletin board B.

3.2 Key Use Phase

3.2.1 Encryption Phase

In the encryption phase, from the bulletin board B member $u_k \ \ k \in \{1,2,...,n\}$ gets up-to-date $\{P_i\} \ \ i \in \{1,2,...,n\}$ and P_0^k. When u_k needs communicate with members u_j $j \in \{1,2,...,k-1,k+1,...,n\}$, it selects two random values s and r from $[1, p-2]$, and computes $g^{p_n+s} \bmod p$ to replace the parameter P_n which is the last item in encryption key $\{P_i\}$. Member u_k computes the set $\{g^{rp_i} \bmod p\} \ \ i \in \{1,2,...,n-1\}$ with $\{P_i\}$ $i \in \{1,2,...,n-1\}$ and r. At last, u_k encrypts the plaintext m into cixphertext $mg^{rs} \bmod p$ and sends $mg^{rs} \bmod p$ and $\{g^{rp_i} \bmod p \mid 0 \le i \le n\}$ to u_j. So the process is shown as follows:

$$E_{r,s,\{g^{p_i}\}}(m) = < mg^{rs} \bmod p, \{g^{rp_i} \bmod p\} > \ \ i \in \{1,2,...,n\} \tag{2}$$

3.2.2 Decryption Phase

In the decryption phase, the member $u_j \ \ j \in \{1,2,...,n\}$ uses the secret key x_j $j \in \{1,2,...,n\}$ to decrypt the cixphertext.

Member u_j computes set $\{g^{rp_i x_j^{n-i}} \bmod p\}$ and $\prod_{i=0}^{n} g^{(-1)^i rp_i x_j^{n-i}} \bmod p \quad i \in \{1,2,...,n\}$. We have that

$$\begin{aligned} &\prod_{i=0}^{n} g^{(-1)^i rp_i x_j^{n-i}} \bmod p \\ &= g^{\sum_{i=0}^{n}(-1)^i rp_i x_j^{n-i}} \bmod p \\ &= g^{r\prod_{i=1}^{n} t(x_j - x_i)+rs} \bmod p \end{aligned} \tag{3}$$

Because $x_j \in \{x_i\}$, so $r\prod_{i=1}^{n}(x_j - x_i)^n + rs = rs$. We have that

$$g^{r\prod_{i=1}^{n} t(x_j - x_i)+rs} \bmod p = g^{rs} \bmod p \tag{4}$$

Member u_j decrypts the cixphertext as follows:

$$\begin{aligned} &mg^{rs} / \prod_{i=0}^{n} g^{(-1)^i rp_i x_j^{n-i}} \bmod p \\ &= mg^{rs} / g^{rt\prod_{i=1}^{n}(x_j - x_i)+rs} \bmod p \\ &= m \end{aligned} \tag{5}$$

So the process of decryption is represented as follows:

$$D_{x_j \in \{x_i\},\{g^{rp_i}\}}(E_{r,s,\{g^{p_i}\}}(m)) = m \quad i \in \{1,2,...,n\} \tag{6}$$

3.3 Rekey Phase

A rekeying is triggered when membership changes in space networks.

3.3.1 Member Joining

When a new member u_{n+1} joins in space network, u_{n+1} selects a new decryption key to encryption key $\{P_i\} \quad i \in \{1,2,...,n\}$ and tells the C its periodic time.

Step 1: The member u_{n+1} selects a random value x_{n+1} as decryption key from $[1, p-2]$, and it sends the secret value x_{n+1} to the C via a secure channel (e.g., face-to-face manner).

Step 2: The C receives the value x_{n+1} and keep it secretly, afterwards the C does the following:

(1) The C selects a random value t' from $[1, p-2]$ to compute $P_0' = g^{t'} \bmod p$, updates $P_i = ((P_i)^{t^{-1}})^{t'} \quad i \in \{1,2,...,n\}$ and $P_0^j = (P_0^j)^{t^{-1}t'} \quad j \in \{1,2,...,n\}$ with t' and $\{x_i\}$ $i \in \{1,2,...,n\}$, so $p_i' = p_i \times t^{-1} \times t'$;

(2) The C updates the public encryption key by computing $P_1' = P_1 \times g^{t' x_{n+1}} \bmod p$, $P_{n+1}' = (P_n)^{x_{n+1}}$ and $P_{i+1}' = P_{i+1} \times (g^{p_i})^{x_{n+1}} \bmod p$, $i \in \{1,2,...,n\}$. The up-to-date public encryption key is changed from $\{P_i\}$ to $\{P_i'\} \quad i \in \{1,2,...,n+1\}$, the C also computes the value $P_0^{n+1} = g^{x_{n+1}t'} \bmod p$;

(3) The C writes the encryption key $\{P_i^{'}\}$ and $\{P_0^i\}$ $i \in \{1,2,...,n+1\}$ in the B;

Step 3: The B issues the public encryption key $\{P_i^{'}\}$ and $\{P_0^i\}$. The issue time of encryption key is also given. The form of $\{P_i^{'}\}$ is shown as follows;

$$\begin{cases} P_1^{'} = g^{p_1^{'}} \bmod p = g^{t^{'}(x_1+x_2+x_3...+x_n+x_{n+1})} \bmod p \\ P_2^{'} = g^{p_2^{'}} \bmod p = g^{t^{'}(x_1x_2+x_1x_3+...+x_1x_{n+1}+x_2x_3+...+x_nx_{n+1})} \bmod p \\ ... \\ P_n = g^{p_n^{'}} \bmod p = g^{t^{'}(x_1x_2...x_i...x_n+x_2x_3...x_i...x_{n+1}+x_1x_2...x_ix_{i+2}...x_{n+1}+...++x_1x_2...x_{n-1}x_{n+1})} \bmod p \\ P_{n+1}^{'} = g^{p_{n+1}^{'}} \bmod p = g^{t^{'}x_1x_2...x_i...x_nx_{n+1}} \bmod p \end{cases} \quad (7)$$

Step4: The member u_{n+1} gets the updated public encryption key $\{P_i^{'}\}$ $i \in \{1,2,...,n+1\}$ and P_0^{n+1} from the B.

3.3.2 Member Leaving

When a member u_n leaves space networks, the C updates the public encryption key. The other members keep decryption key unchanged. The procedure is shown as follows.

Step1 : The member u_n sends the leaving messages to the C when u_n comes into the range of the C's control area;

Step2 : If C accepts the requisition of leaving, the C does the following:

(1) The C selects random value $t^{'}$ from $[1, p-2]$, computes $P_i^{''} = ((P_i)^{t^{-1}})^{t^{'}}$ and $P_0 = g^{t^{'}} \bmod p$, $i \in \{1,2,...,n-1\}$, so $p_i^{''} = p_i \times t^{-1} \times t^{'}$;

(2) The C computes $P_0^{''} = g^{t^{'}} \bmod p$, $P_1^{''} = (P_1^{''} / g^{t^{'}x_n}) \bmod p$, $P_{n-1}^{''} = (P_n^{''})^{x_n^{-1}}$ and $P_i^{''} = (P_i^{''} / (g^{p_{i-1}^{''}})^{x_n}) \bmod p$, $i \in \{2,3,...,n-1\}$;

(3) The C computes $P_0^i = g^{x_i t^{'}} \bmod p$ with P_0, $i \in \{1,2,...,n-1\}$;

(4) The C writes the encryption key $\{P_i^{''}\}$ $i \in \{1,2,...,n-1\}$ and $\{P_0^i\}$ $i \in \{1,2,3,...,n-1\}$ in the B;

Step 3: The B issues the public encryption key $\{P_i^{''}\}$ and $\{P_0^i\}$ $i \in \{1,2,...,n-1\}$. The issue time of encryption key is also given. $\{P_i^{''}\}$ is shown as follows;

$$\begin{cases} P_1^{''} = g^{p_1^{''}} \bmod p = g^{t^{'}(x_1+x_2.....+x_{n-1})} \bmod p \\ P_2^{''} = g^{p_2^{''}} \bmod p = g^{t^{'}(x_1x_2+...+x_1x_{n-1}+x_2x_3+...+x_2x_{n-1}+...+x_{n-2}x_{n-1})} \bmod p \\ ..... \\ P_{n-2}^{''} = g^{p_{n-2}^{''}} \bmod p = g^{t^{'}(x_2...x_{n-1}+x_1x_3...x_{n-1}+...+x_1x_2...x_{n-2})} \bmod p \\ P_{n-1}^{''} = g^{p_{n-1}^{''}} \bmod p = g^{t^{'}x_1x_2...x_{n-1}} \bmod p \end{cases} \quad (8)$$

3.3.3 Key Replacing

When a member u_n needs to update the decryption key from x_n to $x_n^{'}$, the $x_n^{'}$ is a legitimate element in set $\{x_i\}$ $i \in \{1,2,...,n\}$. The procedure is shown as follows.

Step1: The member u_n selects a secret value $x_n^{'}$ from $[1, p-2]$ at random and sends it to the C via a secure channel, it tells the C to replace x_n with $x_n^{'}$;

Step2: If the request is allowed, the C does the following:

(1) The C computes $\{P_i^{'}\}\ i \in \{1,2,...,n\}$ with $\{P_i\}\ i \in \{1,2,...,n\}$ and $\Delta x = x_n^{'} - x_n$;

$$\begin{cases} P_1^{'} = g^{p_1 - tx_n + tx_n^{'}} \\ P_2^{'} = g^{p_2} g^{(p_1 - tx_n)t\Delta x} \\ P_3^{'} = g^{p_3} g^{(p_2 - (p_1 - tx_n)tx_n)t\Delta x} \\ \qquad\qquad ..... \\ P_i^{'} = g^{p_i} g^{(p_{i-1} - (p_{i-2} - (...(p_2 - (p_1 - tx_n)tx_n)...tx_n)tx_n)t\Delta x} \end{cases} \quad (9)$$

(2) The C selects a random value $t^{'}$ from $[1, p-2]$, computes $P_i^{'} = g^{p_i^{'} \times t^{-1} \times t^{'}} \bmod p$, $g^{t^{'}} \bmod p$ and $P_0^i = g^{x_i t^{'}} \bmod p$ with $\{x_i\}$;

(3) The C writes updated decryption key $\{P_i^{'}\}\ i \in \{1,2,...,n\}$ and $\{P_0^i\}\ i \in \{1,2,...,n\}$ in the B.

Step 3: The B issues the public encryption key $\{P_i^{'}\}$ and $\{P_0^i\}$. The issue time of encryption key is also given again. $P_i^{'}\ i \in \{1,2,...,n\}$ is shown as following;

$$\begin{cases} P_1 = g^{p_1} \bmod p = g^{t^{'}(x_1 + x_2 + x_3 ... + x_n^{'})} \bmod p \\ P_2 = g^{p_2} \bmod p = g^{t^{'}(x_1x_2 + x_1x_3 + ... + x_1x_n^{'} + x_2x_3 + ... + x_{n-1}x_n^{'})} \bmod p \\ \qquad\qquad ..... \\ P_n = g^{p_n} \bmod p = g^{t^{'}(x_1x_2 ... x_i ... x_n^{'})} \bmod p \end{cases} \quad (10)$$

3.3.4 Maintenance Phase

When a member $u_i\ i \in \{1,2,...,n\}$ comes into the range of the C after a time period, the u_i's decryption key remains unchanged since the u_i does not implement the process of rekeying. Meanwhile, the u_i gets the up-to-date public encryption $\{P_i\}$ and P_0^i from the B in time. But if an existing member's periodic time in next meeting with the C is more than the pre-configuration time, the C would implement rekeying with member leaving's rekey method to revoke the overtime member's legitimate identity.

4. Security Proof

Theorem 1 The AKMSN key management scheme meets key independence.

Proof Key independence is noted that an PPT attacker who compromising some key can not computes other keys. Firstly any element x_i belonging to set $\{x_i\}$ is selected as decryption key at random by member u_i, so it is difficultly to guess successfully the secret value x_i for member $u_j (i \neq j)$ without any information. Secondly, whether compromised the u_j has capability to crack the value x_i according to $\{P_i\}$ in public channel or not. We supposed that u_j has known any element value in set $\{x_i\}$ except x_i, so each P_i represents as $P_i = g^{a_i x_i + b_i} \bmod p$. Thus the question of resolving value x_i with known value $\{P_i\}$, $\{a_i\}$ and $\{b_i\}$ is reduce to the question of resolving value x_i with known value $g^{x_i} \bmod p$, obviously this

is a DH problem [25], which is negligible to crack for the u_j. Above all It is negligible that a disclosure of a decryption key compromise other decryption keys, so the AKMSN scheme meets key independence.

Theorem 2 The AKMSN scheme guarantees backward security against any PPT attacker.

Proof It is sure that a new joining member u_{n+1} has capability of decrypting a secret which is generated with the updated encryption key $\{P_i^{'}\}$, because it's decryption key $x_{n+!}$ is the root of equation $f^{'}(x)=t^{'}\sum_{i=1}^{n+1}(x-x_i)$ $i\in\{1,2,...,n+1\}$. However, since the value $x_{n+!}$ is not the root of equation $f(x)=t\sum_{i=1}^{n}(x-x_i)$, a secret with onbeforeupdate encryption key $\{P_i\}$ $i\in\{1,2,...,n\}$ is not be decrypted by the $u_{n+!}$. At time, the value P_0 is encrypted with the secret set $\{x_i\}$ $i\in\{1,2,...,n\}$, it is negligibly that member u_{n+1} crack the $\{P_0^i\}$ due to $x_{n+!}$ is not belong to set $\{x_i\}$ $i\in\{1,2,...,n\}$. In a word, the AKMSN scheme guarantees backward security when a new member joins in network.

Theorem 3 The AKMSN scheme guarantees forward security against any PPT attacker.

Proof Without loss of generality, the exited member is u_n. it has revoked the capability of decrypt secret, which include two aspects, one is it's secret key x_n is not decryption key for encryption key $\{P_i\}$; the other one is the u_n can not revise the encryption key $\{P_i\}$ to reset the value x_n as decryption key. Point to the first one, after member u_n leaves space network, the value x_n is not belong to the set $\{x_i\}$, so the x_n is not the root of equation $f^{'}(x)=t^{'}\sum_{i=1}^{n-1}(x-x_i)$. If the member u_n uses x_n to decrypt a ciphertext, which is shown as follows:

$$\begin{aligned}&\prod_{i=0}^{n-1}g^{(-1)^i rp_i x_n^{n-i}+rs}\bmod p\\&=g^{\sum_{i=0}^{n-1}(-1)^i rp_i x_n^{n-i}}\bmod p\\&=g^{r\prod_{i=1}^{n-1}t(x_n-x_i)+rs}\bmod p\end{aligned}\tag{11}$$

As the equation $g^{r\prod_{i=1}^{n-1}t(x_n-x_i)+rs}\bmod p=g^{rs}\bmod p$ is not true, so u_n can not get plaintext m with x_n. Point to the second one, after the u_n leaves space network, the C re-selects the random value $t^{'}$, and computes $\{P_0^i\}$, so the u_n can not reset x_n as a decryption key again because the probability of getting the value P_0 is negligible based on DH problem. Above all, the AKMSN scheme guarantees forward security when a member leaves.

5. Performance Analyses

5.1 Storage Cost

Each member has a decryption key and a encryption key, their storage cost are a unit and $n+1$ units as encryption key is composed of $n+1$ parts. So storage cost of decryption key is $O(1)$ and storage cost of encryption key is $O(n)$, so the relationship between storage cost of our proposed encryption key and space network scale is a linear correlation.

5.2 Round Complexity

The space network time delay is shorten efficiently if round complexity is reduced. In agreeing the shared encryption key, the round number between member and C is two, in first

round member submits a random secret as a decryption key to C, the second round, the C sends the encryption key to members and bulletin board. In rekeying phase, if a new member joins network, firstly the new member sends the decryption key to C, secondly the C updates the public encryption key on bulletin board, finally the new member saves the updated shared encryption key from bulletin board. If a member leaves the network, the exiting member only sent a leaving message to the C, and the C updates the encryption key in bulletin board. So the round complexity of rekeying is constant value when membership changes.

5.3 Computation Complexity

In consulting shared encryption key phase, each member selects random value as decryption key and the C implements $\sum_{i=1}^{n} C_n^i + 1$ exponential modular computations for encryption key.

In encryption phase, encrypter selects two random value and implements $n+2$ exponential modular computations, decrypter implements $n(n-1)/2$ exponential modular computations for decrypting. In order to reduce the burden of computation, a few same exponential modular computation could be implemented by the C in the initial phase for decrypter, the number of exponential modular computations is reduce to n for decrypter, but the number of exponential modular computations is increased to $\sum_{i=1}^{n} C_n^i + (n^2+3n)/2+1$.

In member joining phase, the C selects a random value and implement $4n+7$ exponential modular computations, joining member select a random value.

In member exiting phase, the C selects a random value and implements $4n-1$ exponential modular computations, if member replaces its decryption key with new secret value, it needs to select a random value, C implements $4n+3$ exponential modular computations.

From above analysis, the relationship between the member's computation complexity and space network scale is a linear correlation.

5.4 Scalability

Due to the range of rekeying is limited to single member, so it is easily to new member join in network without other members agree, in other word, the performance of the proposed scheme is not deteriorated significantly when the scale of network becomes more bigger. And the procedure of extending network size is simple, new members select one random number as decryption key and agree public encryption key with the server *C*.

5.5 Communication Overload

As the 1-affects-*n* problem is solved in our proposed AKMSN scheme, the scheme's communication overload is minimal, only the updated member participates in rekey process. The legitimate members' communication overload includes sending a secret value to C and getting encryption key that has $n+1$ parts. So the legitimate members' communication overload is $n+2$.

5.6 Comparison

Currently space network key management schemes are the combine of several ground key management schemes, so the comparison result is only between our proposed scheme and exiting key management schemes. In Table 1, we compare some of the currently protocols with the proposed scheme against the following criteria, including key independence, computation, round, storage and sever capability. In comparison to other schemes, the round cost of the AKMSN scheme is constant value, so it is unrelated to space network scale. The complexity of computation and storage is has a linear correlation with network scale. However, the computation complexity is more than the schemes based on some topology structure, such as Octopus and DH-LKH.

Table 1. Comparison of Some Ground Key Management Protocols in Agreeing Key Phase

Scheme	Key Independence	Computation	round	storage	Key Manager Server capability
AKMSN	Yes	n	2	$n+2$	Limited
Ingemarson et al.	Yes	n	$n-1$	1	NO
GDH	Yes	$n+1$	n	n	NO
Octopus	Yes	$3n-4$	$2\left\lceil \frac{n-4}{4} \right\rceil +2$	1	NO
STR	Yes	n	n	n	NO
DH-LKH	Yes	$\log_2 n$	$\log_2 n$	$\log_2 n-1$	NO
BD	Yes	3	2	1	NO
GKMP	Yes	null	1	2	Powerful
Secure Lock	Yes	Chinese Remainder Theorem	2	1	Powerful
LKH	Yes	hash	$\log_2 n$	$\log_2 n$	Powerful
SAKM	NULL	0	1	1	Limited
Kronos	NULL	hash	0	1	No online
Probabilistic key sharing	Yes	0	0	Part of keys	Off-line and key pool

6. Conclusions and Future Work

In this paper, the AKMSN scheme is proposed, each member has a different decryption key and has a common public encryption key for secure communication. When a member leaves or joins the network, the range of rekeying is limited to single member. And non-updated members' decryption key still keeps its validity to public encryption key. As the 1-affects-*n* problem is deal with successfully, the performance of the purposed scheme is not related to the mobility, topology structure and network scale. Meanwhile the good scalability of the proposed scheme is advantage of extending the network scale. In the future, we plan to study more efficient key management schemes for space networks.

References

[1] Posner EC, Stevens R. Space communication-Past, Present, and Future*. IEEE Communications Magazine*. 1984; 22(5), 8-21.
[2] Zeng Feng, Yao Lan, Chen Zhigang. Impact of topology and traffic on interference and routing in IEEE 802.11 wireless mesh network. *Telkomnika*. 2012; 10(4): 823-830.
[3] Mantoro Teddy, Zakariya Andri. Securing e-mail communication using hybrid cryptosystem on android-based mobile devices. *Telkomnika.* 2012; 10(4): 827-834.
[4] Wen JP, Zhang YJ, Zhao B. *L-band SAR-processor for the Chinese SAR satellite*. ICCEA 2004-2004 3rd International Conference on Computational Electromagnetics and its Applications. BRSI. 2004; 3: 399-402.
[5] Bell DJ, Cesarone R, Ely T, Edwards C, Townes S. *Mars network: a Mars orbiting communications and navigation satellite constellation*. IEEE Aerospace Conference Proceedings. Pasadena. 2000; 7: 75-88.
[6] Harney H, Muckenhirn C. *Group Key Management Protocol (GKMP) Architecture*. Internet Engineering Task Force. RFC: 2093. 1997.
[7] Chiou GH, Chen WT. Secure Broadcast using Secure Lock. *IEEE Transactions on Software Engineering*. 1989; 15(8): 929-934.
[8] Ingemarson I, Tang D. Wong C. A Conference Key Distribution System. *IEEE Transactions on Information Theory*.1982; 28(5), 714–720.
[9] Steiner M, Tsudik G, Waidner M. *Diffie-Hellman key distribution extended to group communication*. Proceeding CCS '96 Proceedings of the 3rd ACM conference on Computer and communications security. New York.1996; 3: 31-37.
[10] Becker C, Wille U. *Communication complexity of group key distribution*. CCS '98 Proceedings of the 5th ACM conference on Computer and communications security. New York.1998; 5: 1-6.
[11] Burmester M, Desmedt Y. *A secure and efficient conference key distribution system*. Lecture Notes in Computer Science. 1994; 950(1995): 275–286.
[12] Yacine C, Hamida S. Group Key Management Protocols: A Novel Taxonomy. *International. Journal of Information Technology*. 2005; 2(2): 105-119.

[13] Chung KW, Gouda M, Lam SS. *Secure group communications using key graphs*. Networking. IEEE/ACM Transactions on.2000; 8(1): 16-30.
[14] Kim Y, Perrig A, Tsudik G. *Tree-based group key agreement*. ACM Transactions on Information and System Security. 2004: 7(1): 60-96.
[15] Lin Y, Kong XW, Wu GW, Lin C. Tree-based Multicast Key Management in ubiquitous computing environment. *International Journal of Ad Hoc and Ubiquitous Computing*. 2011; 8(1): 27-35.
[16] Omar C, Anis K. *A logical neighbor tree secure group communication scheme for wireless sensor networks*.Ad Hoc Networks. 2012; 10(7): 1419-1444.
[17] Steer D, Strawczynski LL, Diffie W, Weiner M. *A Secure Audio Teleconference System*. CRYPTO '88 Proceedings on Advances in cryptology. New York. 1988; 520-528.
[18] A Perrig. *Efficient Collaborative key Management protocols for Secure Autonomous Group Communication.* International Workshop on Cryptographic techniques and E-commerce: Hong Kong. 1999; 1-11.
[19] Wong C, Gouda KM, Lam SS. *Secure Group Communications Using Key Graphs.* IEEE/ACM Transactions on Networking. 2000; 8(1): 16–30.
[20] Klaoudatou E, Konstantinou E. *A Survey on Cluster-Based Group Key Agreement Protocols for WSNs*. Communications Surveys & Tutorials. 2011; 13(3): 429-442.
[21] Setia S, Koussih S, Jajodia Harder E. *Kronos: A scalable group re-keying approach for secure multicast.* IEEE Symposium on Security and Privacy. New York. 2000; 1: 215-228
[22] Eschenauer L, Gligor VD. *A key-management scheme for distributed sensor networks*. Proceeding CCS '02 Proceedings of the 9th ACM conference on Computer and communications security. New York. 2002; 1: 41-47.
[23] Howarth MP, Iyengar S, Sun ZL, Cruickshank H. Dynamics of Key Management in Secure Satellite Multicast. *IEEE Journal on Selected Areas in Communications*. 2004; 22(2): 308-319.
[24] Arslan MG, Alagoz F. *Security issues and performance study of key management techniques over satellite links*. 2006 11th International Workshop on Computer Aided Modeling and Design of Communication Links and Network.Trento. 2006: 122-128.
[25] Eng B, Robert HD, Zhu HF. *Variations of Diffie-Hellman Problem*. Lecture Notes in Computer Science. 2003; 836: 301-312.