Chapter

Key Management Techniques for Wireless Mesh Network

Vinh Truong Quang and Hoa Le Viet

Abstract

Key management is one of the most important tasks in wireless mesh network. This service is responsible for key generation, distribution, and key exchange in a cryptography-based system. Due to the shared nature of WMNs and absence of globally trusted central authority, key management becomes more challenging. This chapter introduces several key management methods that can address these challenges. The fundamental approach is the secret sharing scheme created by A. Shamir, which effectively distributes keys to all participants' network. Based on Shamir's scheme, many authors proposed other algorithms to secure the communication channel in such a way that adversary cannot steal any information about the secret. In addition, in this chapter, a new secret sharing method using real-time synchronization among transceiver devices is presented. In this method, each node generates its key depending on its physical information and the real-time clock. Therefore, public and private keys can be managed efficiently for data encryption and prevent several external attacks to WMNs. A specific protocol is proposed to secure keys while transferring between devices to prevent internal attacks.

Keywords: wireless mesh network, key management, wireless encryption, secret sharing, cryptography

1. Introduction

WMNs are increasingly becoming a prominent architecture that are used in various applications such as home networking, transportation, enterprise networking, etc. [1]. WMNs are very vulnerable to be attacked by opponents. There are three types of attack: active attacks, passive attacks, and message distortion [2]. In order to guarantee the security of data in such networks, cryptography is one of the most popular choices. Therefore, key management services are in demand.

Key management refers to process of cryptographic key generation, distribution, and storage [3]. In addition, the responsibility of key management is establishment of trusted and secure communication between nodes. Due to the unique nature of WMNs, there are three challenges that many existing key management schemes are facing [4]. Firstly, it is difficult to share, transport, and update keys because of the lack of infrastructure in WMNs. Secondly, a distributed certificate authority (CA) model is needed to tackle the absence of fixed central infrastructure in WMNs which is not suitable for public key infrastructure (PKI). Finally, the concern of scalability is undeniable to take advantage of being expandable of WMNs (nodes can join or leave the network).

Many researchers have proposed numerous approaches for group key management. One of the most common group key management methods is secret sharing introduced by Shamir [5]. The schemes allow a master key (secret) to be shared to all authenticated users, but it can just be reconstructed when a node has enough number of shares. Combining with Shamir's method, Li and Xin used the self-certified public key system for proposal of a distributed key management approach [6]. All keys are generated and managed in a self-organizing way within the network, while there is no need of prebuilt trusted relationship between nodes. Lan Yun et al. introduced secret sharing-based management (SSKM) based on Shamir's scheme [7]. The proposed method dynamically generates a different key based on different polynomials from the base station in different periods which can protect the network from the compromised nodes and reduce the high probability of the common key. Filippo Gandino et al. [8] proposed a new key negotiation routine to deal with the case when a node is compromised by adversary. The goal of this algorithm is to reduce the time for the initialization phase as well as reduce the probability of compromised master secret. Singh et al. [9] combined Shamir's scheme and encryption method together by using only hash and XOR function to reduce the overhead for realistic WMNs which have limited resource. All attempts of researchers are to enhance security reliability for key management.

The remaining sections of this chapter are organized as follows. The detail approaches of other authors are introduced in section II, III, and IV. In section V, we proposed a new key management method using real-time synchronization among transceiver devices. In addition, we present our experiments and the result analysis. Finally, the conclusion is drawn in Section VI.

2. Shamir's secret sharing scheme (SSSS)

Shamir's secret sharing scheme (SSSS), also called a (k, n) threshold scheme, is a basic algorithm in cryptography created by Shamir [5]. A secret is divided into multiple parts which each part is given to every participant. To reconstruct the secret, participants have to possess sufficient number of parts. Otherwise, there is no way to reveal the original secret. The goal of Shamir's scheme is to divide S (secret) into n pieces of data $S_1, S_2, \ldots, S_n$ in such a way that knowledge of any k or more S_i pieces makes S easily computable. This means any group of k pieces of data can reconstruct the secret.

Knowledge of $k - 1$ or fewer S_i pieces leaves S completely undetermined. This means secret S cannot be reconstructed with fewer than k pieces.

The (k, n) threshold scheme is based on polynomial interpolation. To build polynomial $f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{k-1}x^{k-1}$, $(k - 1)$ elements $a_1, a_2, \ldots, a_{k-1}$ are selected randomly and let $a_0 = S$. With a set id_i, $i = 1, \ldots, n,$ we calculate $f(id_i)$ and distribute a pair of $(id_i, f(id_i))$ to n participants. For secret recovery phase, we need any subset of k of these pairs. From that, we can find the coefficients of the polynomial using interpolation and evaluate secret $S = f(0)$. To be more efficient, the modular arithmetic is used instead of real arithmetic. The coefficients are randomly chosen in the finite field $GF(q)$. Then, the pairs become $(id_i, f(id_i))\ mod\ q)$.

SSSS has a powerful mechanism in key distribution. The number of S_i pieces can be dynamically adjusted according to number of members who join or leave the network. In order to enhance the security level, the shares must be changed frequently without changing the original secret by generating differently the polynomial $f(x)$ with the same free term. SSSS is also suitable for hierarchical system in which the quantity of shares is given to members based on their importance in the system.

The main issue of SSSS comes from cheaters inside the network. In the reconstruction phase, if a member accidently or intentionally provides his fake share, the original secret reconstructed might be wrong as well. Therefore, there is a need of verifying the correctness of the retrieved shares during the reconstruction process.

3. Secret sharing-based key management (SSKM)

Lan Yun et al. [7] presented an algorithm called secret sharing-based key management (SSKM) which can prevent several attacks effectively and reduce the energy consumption. SSKM includes two-level key management. One is to protect communication between base station (BS) and cluster head (CH); another relates to communication between CH and member nodes. Besides, the SSKM utilizes Shamir's secret sharing scheme to distribute keys. The proposed method also dynamically generates different keys based on different polynomials from BS in different periods which can protect the network from the compromised nodes and reduce the high probability of the common keys.

The SSKM was proposed to perform in a hierarchical architecture which consists of a base station and several clusters. Each cluster includes cluster heads and member node. CH manages the cluster and deals with information from member nodes forward to base station (BS).

The SSKM is based on Shamir's scheme to distribute the keys. Though this scheme is information theoretically secured, there are some requirements in this situation:

1. The delivery of shares between dealer and users must operate in a secure channel.

2. The pairs $(x, f(x))$ are made publicly known. However, for security purposes, the pairs must be kept as each user's secret. Therefore, the discrete logarithm in the finite field and DDH (decisional Diffie-Hellman) assumption are adopted to address the problems.

In initial phase, during each session period l $(l = 1,\ldots,M)$, BS randomly generates m polynomials $f(x)$ of $(t - 1)$ degree. One of the polynomials $f_{Cin}(x)$ is used to key distribute between BS and cluster heads. Other $m - 1$ polynomials are used to key distribute among cluster head and member nodes in $m - 1$ clusters, respectively. After that, BS selects M session keys $\{K_{Cin}\}$ and $\{K_{CHi}\}$ from GF(Q) in the finite field Q. $K_{C_{in}}$ is the session key in network key management (between BS and CH), while K_{CH_i} is the session key in cluster key management (between CH and member nodes). Session keys are hidden by calculating $Z_{Cin} = K_{Cin} + S_{Cin}$ and $Z_{CHi} = K_{CHi} + S_{CHi}$; S_{Cin} and S_{CHi} are the secrets. BS/CHs broadcast information Z to each CH/member node.

To protect the communication, the discrete logarithm in the finite field and DDH (decisional Diffie-Hellman) assumption are adopted. As a result, secrets $\{S_{Cin}\}$, $\{S_{CHi}\}$ and session keys $\{K_{Cin}\}$ and $\{K_{CHi}\}$ are kept confidential. After recovering the secret S using (t, n) threshold scheme, users can get session key $K = Z - S$.

SSKM provides an energy-efficient solution in which almost computations were performed by BS, and CHs just exchange parameters to BS to adjust polynomials for key generation/cancelation. SSKM also resolves challenging security issues by localizing key things based on secret sharing scheme. Network key and cluster key management are salient solutions in this work which are mainly responsible for the security protection in a group of members as well as the whole network.

Despite the outstanding advantages, this method is just well-performed in a hierarchical architecture which needs trusted central authorities (BS or CHs). Consequently, the network may be broken down when those authorities are compromised.

4. Hierarchical scheme with transitory master key (HSTMK)

Filippo Gandino et al. [8] introduced a new key management scheme called hierarchical scheme with transitory master key (HSTMK). This approach is based on transitory master key approach and is designed for static wireless sensor network. The proposed scheme includes two elementary key managements. One is plain global key (PKG), which is used by all nodes during the initialization phase and deleted during the working phase. Another is full pairwise keys (FPWK) in which each node shares a specific key with another node. Once an adversary compromises a node before the deletion of key materials, the network is broken down. Therefore, the idea of this method is to split the initialization phase into multi-sub-phase which will increase the overall security level. Furthermore, the HSTMK reduces the time for initialization phase, thus reducing the probability that the master secret is compromised.

A setup task is performed before the initial deployment of the network. In this task, each node is given a common key, called global master key K_{IN}. After that, node u produces its own master key K_U by using a pseudorandom function $f_K(.)$ and global master key K_{IN}:

$$K_U = f_{Kin}(ID_U) \tag{1}$$

Each node has an interval timer which measures the duration of the initialization phase. When the time finishes, all key materials must be deleted to prevent from being stolen by adversaries. This interval value must be selected carefully according to the characteristics of the network.

The initialization phase is divided into four sub-phases, including neighbor discovery, master key computation, pairwise key computation, and acknowledgment.

First, at the neighbor discovery phase, the node broadcast Hello packet to other nodes that identify their neighbors. The packets contain the identification (ID_I) of the senders.

At the master computation phase, the node u calculates the secret of their neighbor v using a pseudorandom function $f_K(.)$ and neighbor's ID_V. Then they delete the global key K_{IN}:

$$K_V = f_{Kin}(ID_V) \tag{2}$$

At the pairwise key computation phase, only one node in a couple of nodes computes the pairwise keys $K_{U\text{-}V}$ and $K_{V\text{-}U}$. Then they delete their neighbor's master key:

$$K_{V-U} = f_{Kv}(ID_u) \text{ or } K_{U-V} = f_{Ku}(ID_V) \tag{3}$$

At the acknowledgment phase, after deleting all key materials, the nodes send acknowledgement messages to authenticate the key establishment.

In addition, the author proposed a mechanism for adding new nodes to network. For the new nodes, at first sub-phase, they broadcast Hello messages to all existing nodes in network. Any nodes which receives these messages will respond with an acknowledgment packet (which contains the ID of the receiver). In the second sub-phase, the new nodes compute the master key of their available neighbors and then delete the global key. Other sub-phases remain unchanged.

An experiment was carried out to compare performance of HSTMK and another method called LEAP+ [10]. The results show that HSTMK is faster than LEAP+ in terms of establishing a pairwise key among nodes. In addition, by increasing the number of nodes in the network, the time before deleting key materials of HSTMK is from 3 to 150 times less than that of LEAPs. The experimental results also highlight the importance of the selection of initialization time. If this value is too low, the proportion of established pairwise keys can reduce, whereas a too long initialization phase would increase the risk of losing secret materials.

5. Real-time key management algorithm

The proposed security algorithm [11] is designed for the purpose of safely transferring keys and synchronous nodes in WMN. In sections 5.1 and 5.2, we will present our key management method based on Adi Shamir's algorithm; the synchronization between nodes by real-time clock helps our keys prevent different types of external attacks. We also present a protocol used for transferring those keys in WMN; this protocol will focus on preventing man-in-middle attack and detecting other abnormal activities in this network.

5.1 Real-time clock key management

The conventional key management methods are easy to be attacked by various attacks such as eavesdropping keys and data, de-authentication, and denial-of-service (DoS). Therefore, we propose to use real-time clock to change continuously private key in key management of each node and synchronize all nodes in WMN, so these nodes will be completely independent of each other. One of those nodes is the network time protocol (NTP) server, and the others are NTP clients. Using the WMN model, the NTP data are transferred quickly enough for synchronization. At a certain point, the nodes will together create a unique key, and every group of n keys is required to reconstruct the same secret for the encryption and decryption.

The process of the proposed method is reversely compared with Adi Shamir's method as shown in **Figure 1**. In the proposed method, the private key is created first instead of the master key. Therefore, the secret will not be detected when the attacker attacks on any node. Besides that, this secret is constantly changed by using a real-time clock module, and thus this makes it more difficult for attackers to be successful in penetrating the network (**Figure 2**).

Private key is generated by a unique value depending on each device—MAC address. A threshold level is required for this process. This parameter will be set depending on the number and installation location of nodes in WMN. The process of private key generation is shown in **Figure 3**.

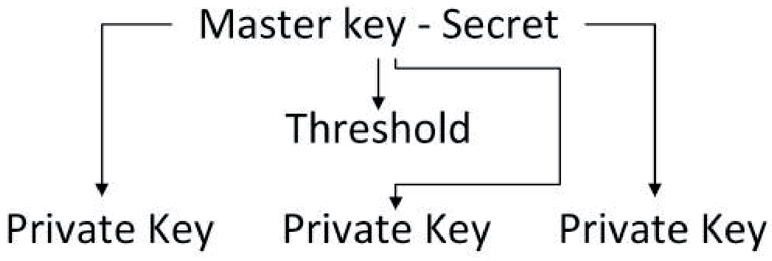


Figure 1.
Adi Shamir's secret sharing scheme.

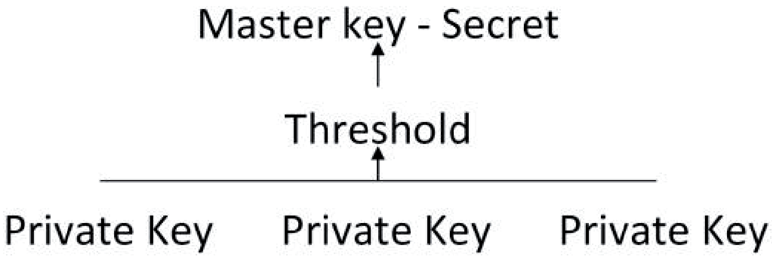


Figure 2.
Proposed secret sharing scheme.

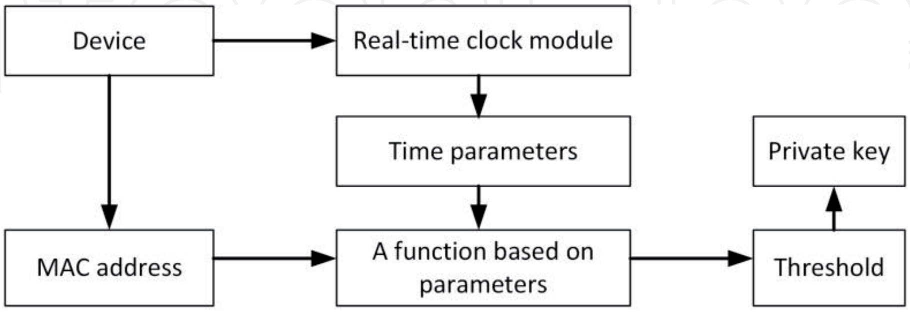


Figure 3.
The process of private key generation.

Reconstructed secret is implemented after each node has enough keys. It is received from nodes in WMN by simple BATMAN protocol which we will talk about in the next section. Lagrange interpolating polynomial was used for our purpose [12]. This is described by the following equations:

For i = (1:node).

$$S = \left(S + f(x_i)\prod_{m=0\wedge m\neq i}^{k-1}(x_m)/(x_m - x_i)\right) \tag{4}$$

If we use this original Lagrange interpolating polynomial, there is a security problem: attackers can gain a lot of information about S with every couple key $(x_i, f\ (x_i))$. They have numbers to guess from instead of an infinite number of natural numbers by using normal basic methods to solve this set of equations.

This problem can be fixed by using finite field arithmetic in a field of size p ∈ P:p > S, p > n. We calculated the couple keys as $(x_i, f(x_i)\ mod\ p)$ instead of $(x_i, f(x_i))$. The lower one sets p, the lower the number of possible values that the attackers have to guess from to set S. Therefore, we made a small change to our key generation function and reconstruction function by the following equations:

$$S = \left(S + p + y.\prod_{k=1\wedge k\neq i}^{k=node}(-x_k).\delta\right) \bmod p \tag{5}$$

$$\delta \times \left(\prod_{k=1\wedge k\neq i}^{k=node}(x_i - x_k)\right) \bmod p \;\; = \;\; 1 \tag{6}$$

where S is the secret value which we need for authentication in WMN. Pair of x and y serves as a key to reconstruct secret as we have presented.

5.2 Proposed security protocol

The protocol which we use for our key management scheme is based on the BATMAN protocol—an efficient protocol used to establish connection in WMN. **Figure 4** describes how our protocol works.

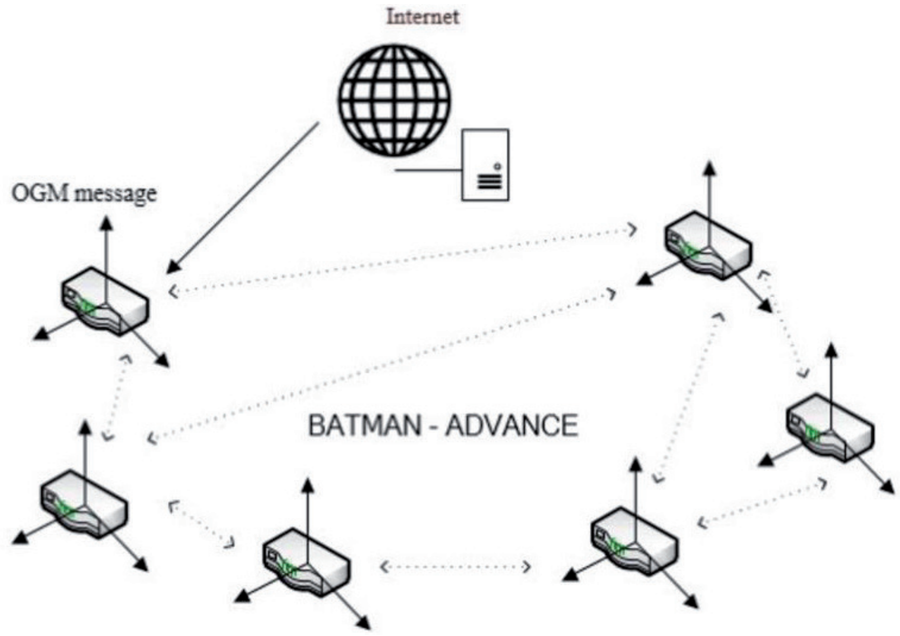


Figure 4.
Modified BATMAN protocol.

While BATMAN advance protocol works as a communicate protocol for sending and receiving data in WMN and detecting node nearby with the same WMN, our proposed protocol uses list neighbor nodes of each node in WMN to work for our purpose. Every node sends its private key frequently to neighbor nodes; an authenticated address list has to be created and checked. This work can be easily done by designing a customized package frame on raw debug socket interface (**Figure 5**).

The objectives of proposed security protocol are the following:

- Encrypt the data by secret which is reconstructed with keys of nearby neighbor nodes.

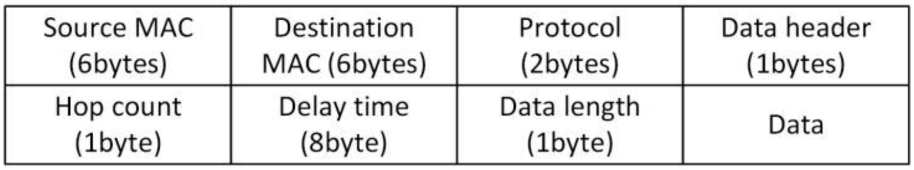
- Warn all nodes in the network when there is an intrusion attack in network.

- Send private key over a man-in-middle node by our frame to increase range of our protocol.

In order to reach these goals, we combine our protocol and scheme into a multi-thread program with the flow graph as shown in **Figure 6**.

Figures 7 and **8** show an example of development with three nodes; node 1 is within the communication range of the others, but the distance between node 0 and node 2 is too long to establish a link.

To prevent man-in-middle attack as we have mentioned before, we encrypt the node's keys each time they are transferred to the other node, so there is a problem on how the requesting node can receive exactly that key with this method. We add a



Source MAC (6bytes)	Destination MAC (6bytes)	Protocol (2bytes)	Data header (1bytes)
Hop count (1byte)	Delay time (8byte)	Data length (1byte)	Data

Figure 5.
Proposed security protocol header frame.

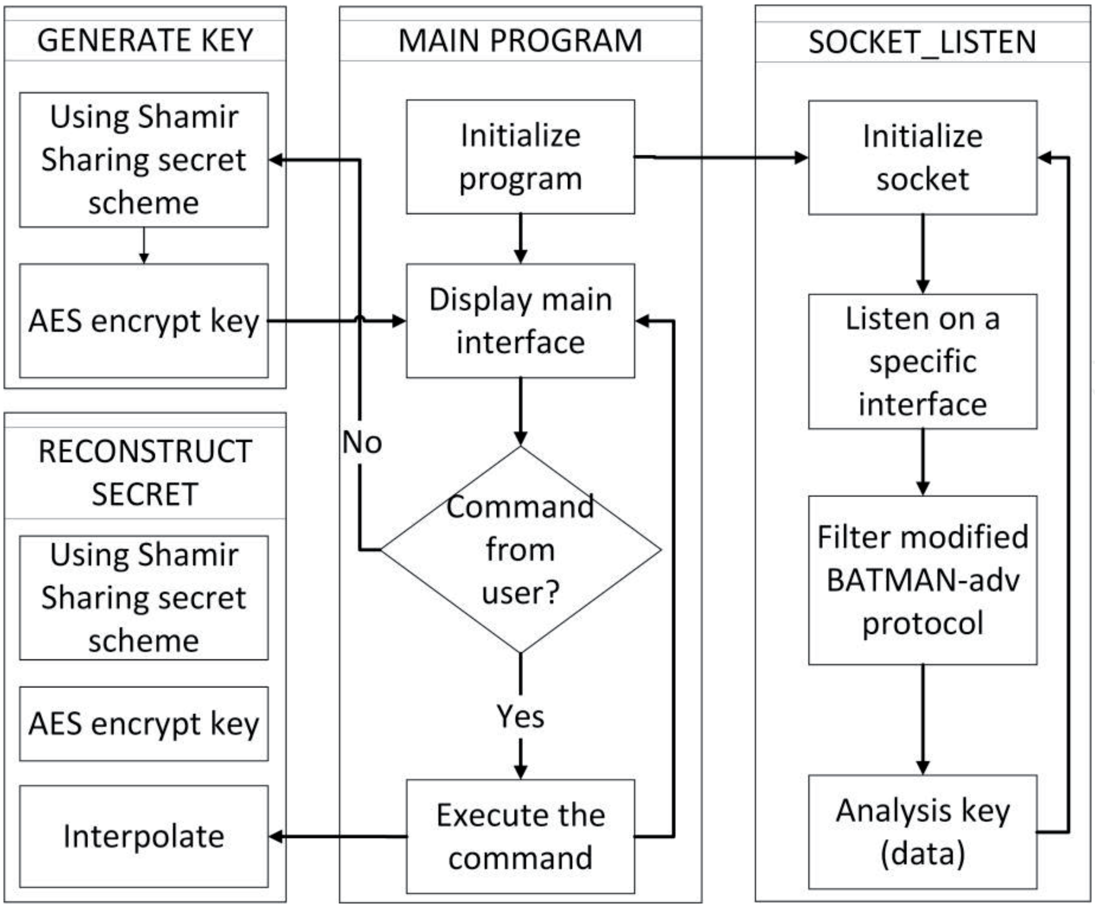


Figure 6.
Proposed program flow graph.

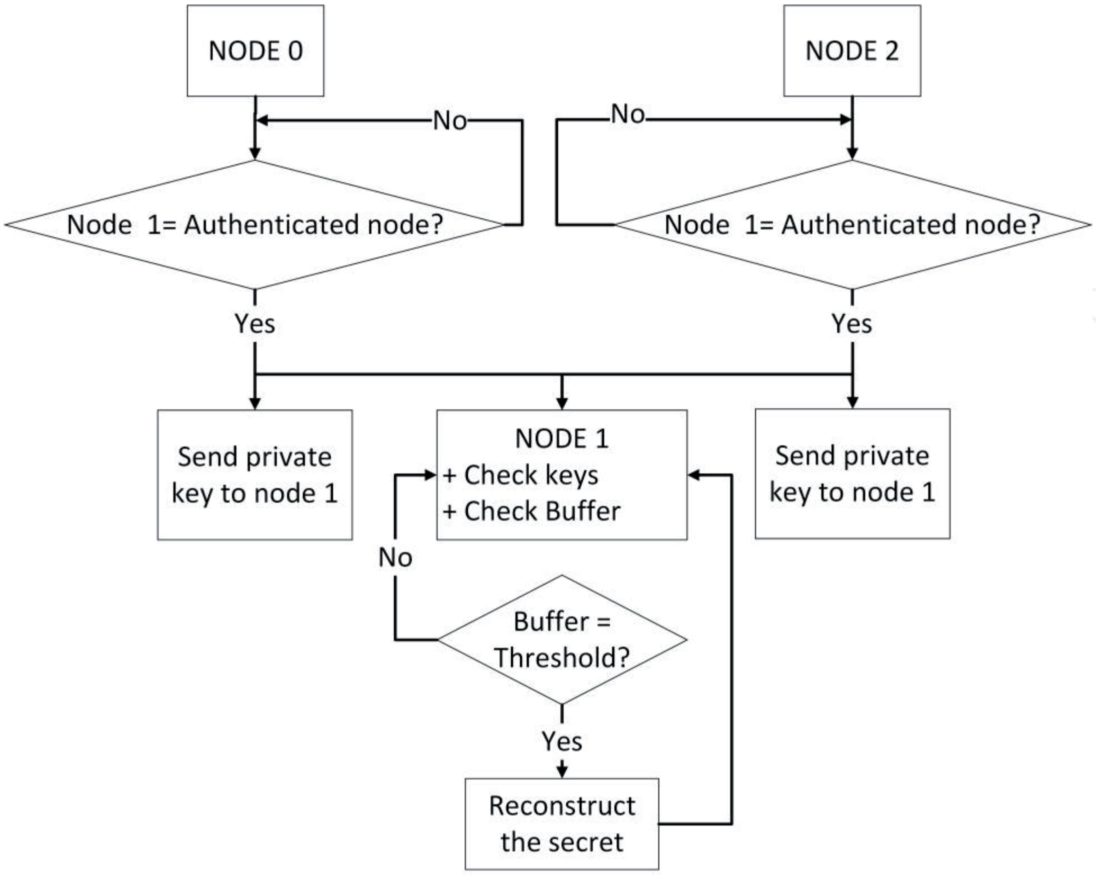

NODE 0
NODE 2
No
No
Node 1= Authenticated node?
Node 1= Authenticated node?
Yes
Yes
Send private key to node 1
NODE 1
+ Check keys
+ Check Buffer
Send private key to node 1
No
Buffer = Threshold?
Yes
Reconstruct the secret

Figure 7.
Secret reconstruction of node 1.

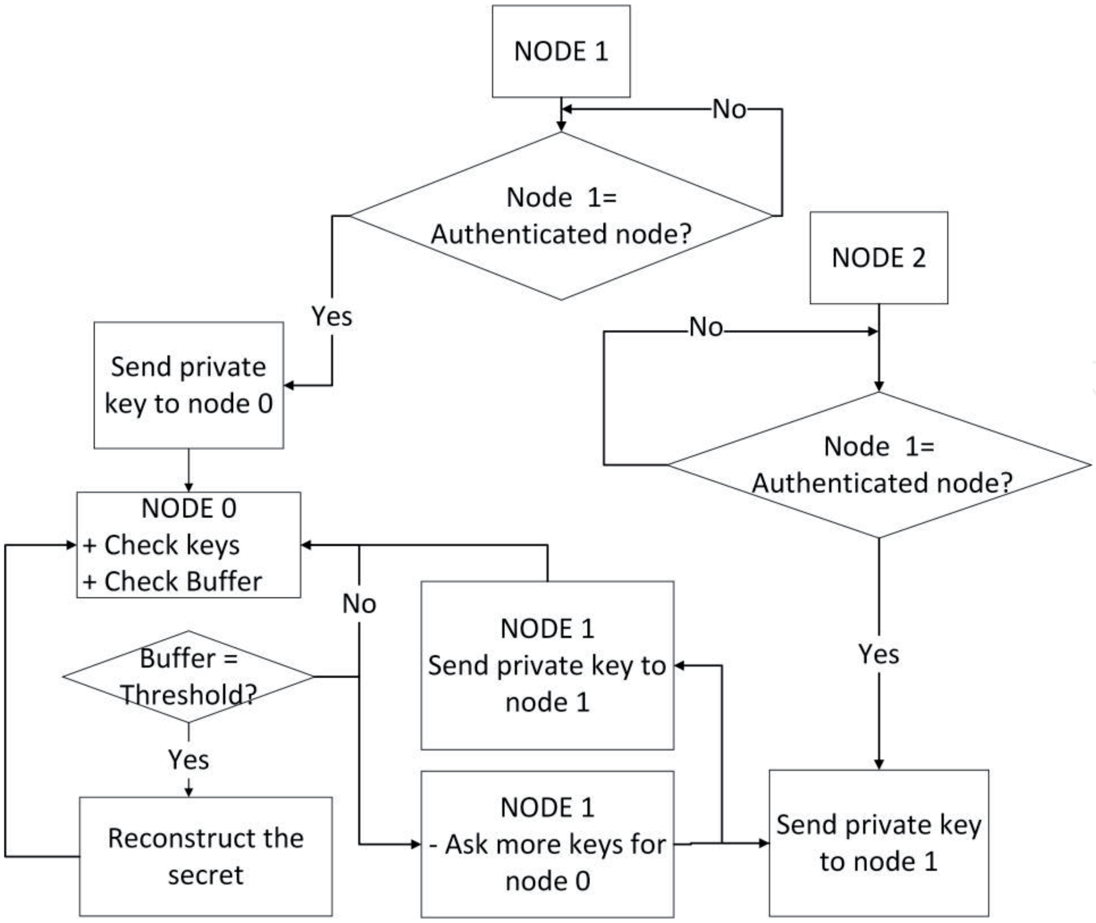


Figure 8.
Secret reconstruction of node 0.

new field "hop count" to our header frame. This field is used for counting the number of nodes which will help the packet from the requester to be transferred to the destination. Then the destination will take that value to encrypt its key <hop count> time s before sending it to the source who requested for more keys. The "key used for encrypting key" is also generated by real-time method; it must be known by all nodes between source and destination to ensure that the decryption on those nodes is correct. Another issue is the case that an intrusion node tries to decrypt more than one time to get the value of encrypted key. In order to solve this issue, we set hop count value equal to 0 so the intrusion node does not know how many times it has to decrypt for getting the key. Only the owner of the key knows this hop count value. The process of the key encryption with hop count values is shown in **Figure 9**.

Because of the delay when transferring the data, it is hard to synchronize the nodes in WMN with our real-time method; the encryption will be incorrect with only NTP system. Thus, we use a buffer at each node in WMN and a field to tell the delay time the request packet sends from the source to destination. When the destination receives the frame, it can send the key it generated at the time the source sends out its request packet; the buffer has responsibility to record all the key value in a minute latest. Therefore, if the delay value in WMN is over 1 minute, requester cannot receive the key from destination. **Figure 10** shows how the buffer and delay time field work.

5.3 System implementation

We consider a WMN consisting of sensor nodes that can work as base stations which will help non-mesh clients communicate together (**Figure 11**). In this experiment,

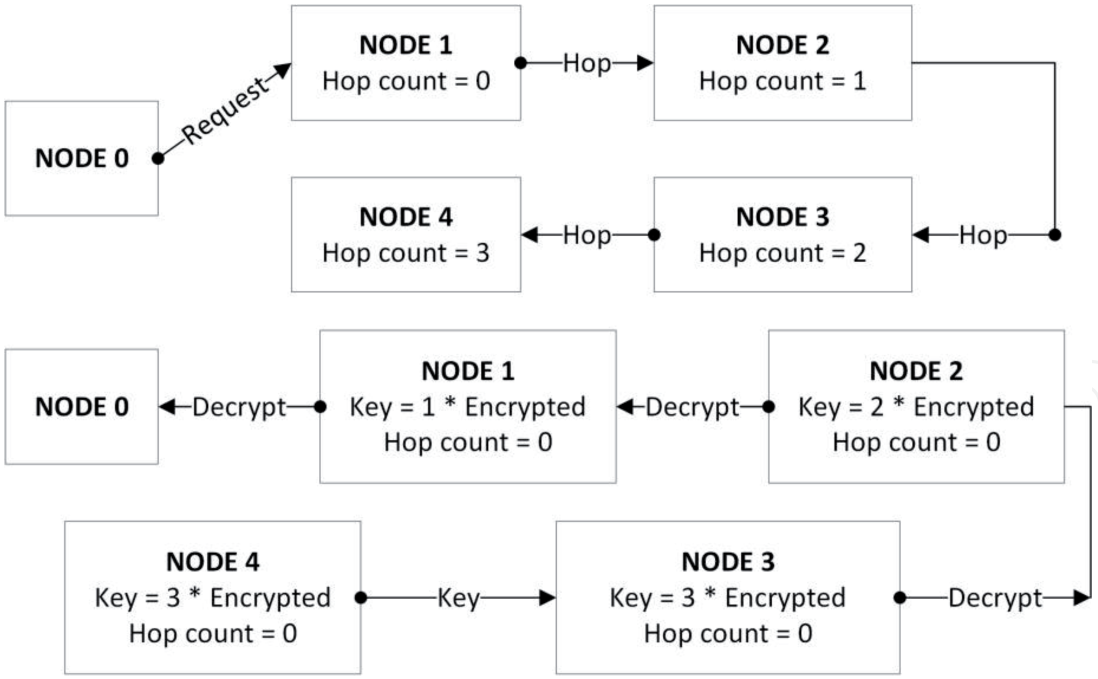


Figure 9.
Encryption of the key with hop count value.

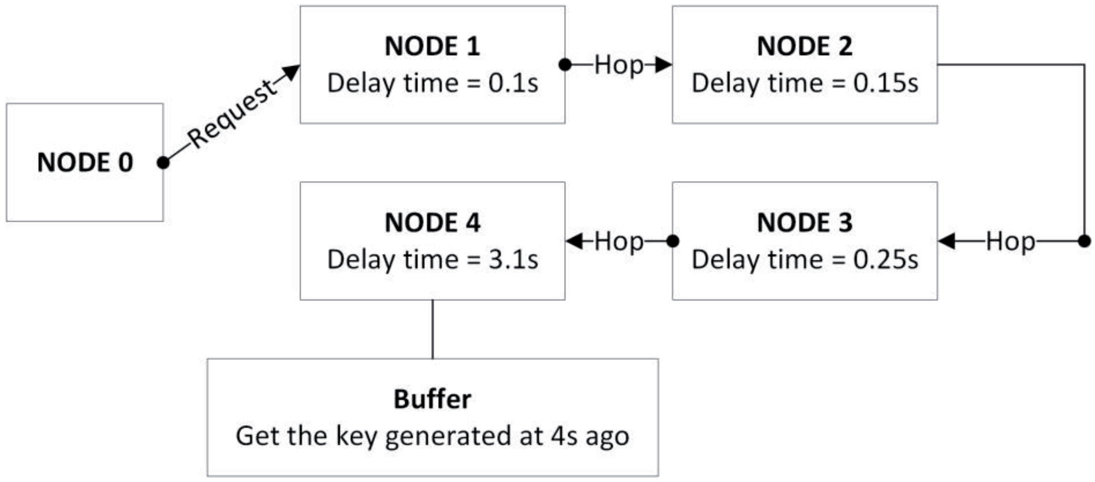


Figure 10.
How the buffer and delay time field work.

we bridge wireless local area network interface and mesh network interface together instead of putting WLAN behind the firewall in order to make our work easier, because we only check if our scheme works perfectly on data link layer not on network layer.

Our test security program has already been installed on every sensor node, and we put them in distance. As our scheme, node 1 receives private key from nodes 2 and 3, combined with its key to reconstruct the original secret which is used to encrypt data. This encrypted data is sent to one of the clients of node 4; after decrypting, we compare the decrypted with the original data to see if our scheme works completely. Another parameter which we have to check is secret after reconstruction. We will list all those parameters in the next section.

5.4 Results

In this experiment, we set secret—public key of WMN equal exactly to minute value of UTC time zone. Therefore, the time on every sensor node must be set at the

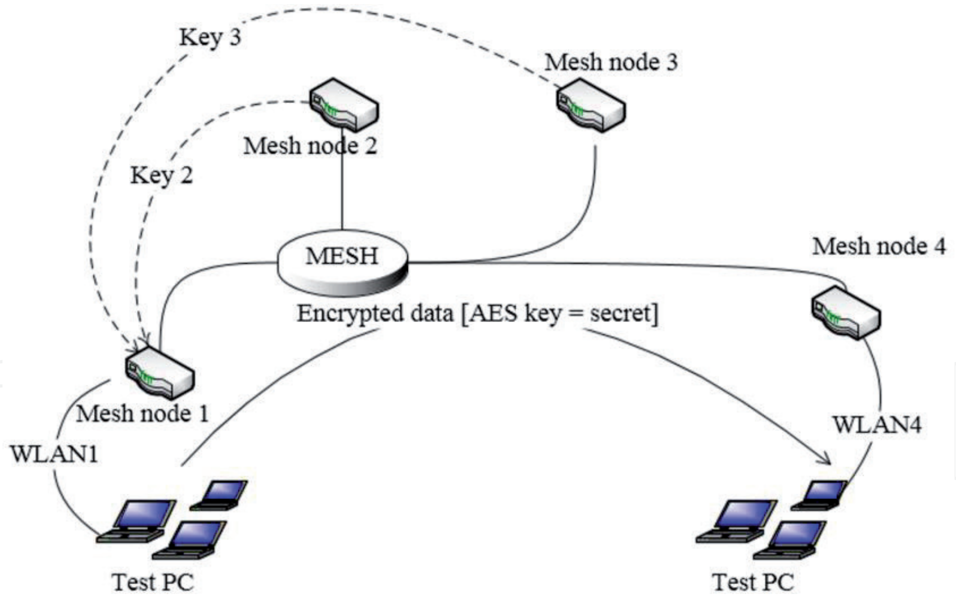


Figure 11.
System model.

Time	Key 1 (Hex)		Key 2 (Hex)		Key 3 (Hex)		Secret(Dec)
19:45	96	C4	85	F4	26	18	64
19:51	EC	C4	FB	F4	E5	18	70
22:06	8C	C4	6C	F4	D5	18	28

Table 1.
Secret reconstruction analysis.

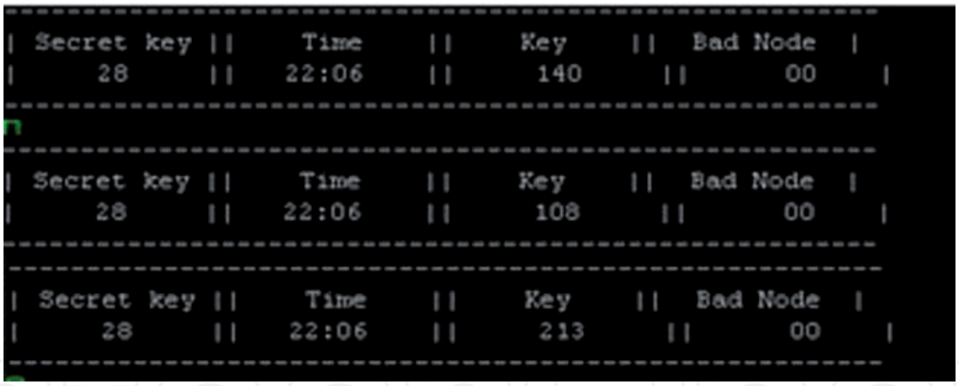
same value. **Table 1** shows the keys (decrypted keys) collected by node 1 needed for secret reconstruction analysis at the different time. We put a simple function for our experiment secret as follows: Secret = Hour value + Minute value. We executed our program on three nodes in this experiment. This program shows us the value of generated key of each node, the number of bad nodes this security detected, and the time and the secret which was reconstructed at that time. **Figure 12** shows all those results of all three nodes at the time 22:06. Each node has a different private key from the others based on its MAC address. But all of them have the same secret at a certain time.

Secret is reconstructed exactly as the origin with at least three private keys of WMN. Therefore, we can run to the next step—data encryption—and original and decrypted data are the same in both transmitter and receiver if it works correctly, in our case are clients of nodes 1 and 3.

After reconstructing the secret completely, data which are sent from node 1 will be encrypted by this secret. Both encrypted data and decrypted data are shown. This data is captured at one of the non-mesh clients of wireless local area network node 4. We had also checked if the keys are secured when they are transmitted in our model (four nodes with maximum hops equal to 2). Then, we used an external node which worked as a monitor node to capture the raw package to check if the keys are encrypted.

We tested our methods to face types of attacks mainly in general wireless network and in particular wireless mesh network.

Firstly, we test our network model with eavesdropping attacks. We use ESP8266 kits to collect all the data from our network; the entire data was encrypted; we also tried to collect private keys from authenticated nodes in this network to reconstruct



```
---------------------------------------------------------------
| Secret key ||     Time     ||     Key     ||  Bad Node  |
|     28     ||    22:06     ||     140     ||     00      |
---------------------------------------------------------------
---------------------------------------------------------------
| Secret key ||     Time     ||     Key     ||  Bad Node  |
|     28     ||    22:06     ||     108     ||     00      |
---------------------------------------------------------------
---------------------------------------------------------------
| Secret key ||     Time     ||     Key     ||  Bad Node  |
|     28     ||    22:06     ||     213     ||     00      |
---------------------------------------------------------------
```

Figure 12.
Experimental figure.

the secret key, but all keys which are transferred in this network had been encrypted with the hop count parameter we had discussed before. To resolve this issue, an attacker needs to decrypt those keys with the pairs of MAC address and hop count parameter, respectively, in this kind of network. Even if attackers can decrypt all keys, they will face with the problem that the keys of our network model are constantly changed over time.

Secondly, we tested our model with many kinds of active attacks, because the nature of the connection on the layer 2 of the original BATMAN protocol has already been pretty tight so almost the active attacks up to this model are neutralized, so the impacts of them are only small impacts on single node and easily detected by our protocol when there are abnormal signs from any nodes in our model.

Next, we tried to use jamming attack to our model. Unfortunately, we have not handled this kind of attack. Therefore, in the near future, we will develop our model to overcome this drawback.

Finally, let see how our model handle the man-in-middle attacks. Because we use real time mainly in our protocol for generating keys, reconstructing the secret, and also detecting abnormal nodes. Therefore, any man-in-middle attacks without being synchronized in real time or do not have the ability to interact with the other authenticated nodes in the specified period that we mentioned in the previous section are defined as abnormal node.

To sum up, attackers only can strike this network model if they know how the protocol works. However, it requires a process to collect, decrypt, synchronize, and analyze accurately complex data from the attacked nodes.

Table 2 shows a comparison between our algorithm and the others over the security reliability criteria. Our proposed algorithm can prevent many types of attacks which we have discussed before—some of them cannot be prevented by the other algorithms.

The original Shamir's algorithm [5] has the weakest security reliability in this table because it only prevents attacks focusing on eavesdropping data. Similarly, SSKM [7] and HSTMK [8] are capable of defending eavesdropping data attacks, but they use two different methods to keep the key materials confidential. SSKM is an improvement of SSSS [5] by using a discrete logarithm algorithm to exchange the keys in a secure channel, while HSTMK takes advantages of separated sub-phases to anticipate the deletion of master secrets. The more resources are consumed by the network models, the more the number of nodes is increased. Consequently, the scale of the model deployed by this method is limited. Therefore, in order to avoid this problem, our algorithm mainly focuses on extending the scale of the network model with the custom protocol using minimal buffer on each node that we mentioned in sections above.

	Proposed algorithm	SSSS [5]	SSKM [7]	HSTMK [8]
Prevent attacks	- Eavesdropping keys and data - De-authentication attack - DoS attack - Replay attack - Man-in-middle attacks	- Eavesdropping data	- Eavesdropping keys and data	- Eavesdropping keys and data

Table 2.
Security reliability comparison.

6. Conclusion

This chapter presented four key management schemes and also security protocol for WMNs. First, Shamir's scheme was a popular method which was used for distributing the keys. The second scheme called SSKM was an improvement of Shamir's scheme by generating different keys in different period as well as using discrete logarithm algorithm to transport the key in a secret way. The third method called HSTMK utilized a key negotiation routine to solve the problem of compromised node. This one also divides initialization phase into four sub-phases to reduce the time requirement and increase the security level. Finally, in our scheme, we establish secured communication sessions between nodes so they can hide their private keys from the other except the requester. That means not only data but also keys were encrypted by combining our scheme with AES encryption. We also use the real-time value to constantly change each node's private key. This has caused great difficulty for anyone who wants to find out private keys of WMN. Comparing with existing security protocols and schemes shows that our scheme is simple to deploy, and it has a better security.

There remain some problems that should be addressed for this security protocol. We need to reduce the amount of the calculations for the proposed protocol which is deployed on routers with small flash memory. Besides, the WMN structure needs to be improved in order to make the system model work efficiently. Thus, these considerations would be developed in the future work.

Author details

Vinh Truong Quang* and Hoa Le Viet
Ho Chi Minh City University of Technology (HCMUT), Vietnam National University, Ho Chi Minh (VNU-HCM), Vietnam

*Address all correspondence to: tqvinh@hcmut.edu.vn

References

[1] Akyildiz F, Wang X, Wang W. Wireless mesh networks: A survey. Elsevier Journal of Computer Networks. 2005;**47**(4):445-487

[2] Siddiqui MS, Hong CS. Security issues in wireless mesh networks. In: Proceedings of the International Conference on Multimedia and Ubiquitous Engineering (MUE'07). New York: IEEE Press; 2007. pp. 41-47

[3] Agarrwal S, Gupta N. Authentication and key management in wireless mesh network. MIT International Journal of Computer Science & Information Technology. Aug. 2012;**2**(2):70-74

[4] Gao L, Chang E, Parvin S, Han S, Dillon T. A secure key management model for wireless mesh networks. IEEE AINA. 2010:655-660

[5] Shamir A. How to share a secret. Communications of the ACM. 1979;**22**(11):612-613

[6] Li F, Xin X, Hu Y. Key management in ad hoc networks using self-certified public key system. International Journal of Mobile Communications. 2007;**5**(1):94-106

[7] Lan Y, Wu C, Zhang Y. A secret-sharing based key management in wireless sensor network. 4th IEEE International Conference on Software Engineering and Service Science (ICSESS). 2013

[8] Gandino F, Ferrero R, Montrucchio B, Rebaudengo M. Fast hierarchical key management scheme with transitory master key for wireless sensor networks. IEEE Internet of Things Journal. 2016

[9] Singh A, Awasthi AK, Singh K. Lightweight multilevel key management scheme for large scale wireless sensor network. In: International Conference on Computing for Sustainable Global Development (INDIACom). 2016

[10] Zhu S, Setia S, Jajodia S. Leap+: Efficient security mechanisms for large-scale distributed sensor networks. ACM Transactions on Sensor Networks. Nov. 2006;**2**(4):500-528

[11] Hoa LV, Vinh TQ. Real-time key Management for Wireless Mesh Network. Journal of Telecommunication, Electronic and Computer Engineering. 2018;**10**(2-6):13-18

[12] Whittaker ET, Robinson G. Lagrange's formula of interpolation. In: The Calculus of Observations: A Treatise on Numerical Mathematics. 4th ed. New York: Dover; 1976. pp. 28-30