1-1-2009

A Security Enhanced AODV Routing Protocol For Wireless Mesh Networks

Ahuang Wang
Ryerson University

A SECURITY ENHANCED AODV ROUTING PROTOCOL FOR WIRELESS MESH NETWORKS

By Zhuang Wang

B. Sc., University of Central Lancashire, 1999
M.Eng., The University of Sheffield, 2000
England, U.K.

A thesis
presented to Ryerson University
in partial fulfillment of the
requirements for the degree of
Master of Applied Science
in the Program of
Electrical and Computer Engineering

Toronto, Ontario, Canada, 2009

Author's Declaration

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Author's signature: _

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Author's signature: ___

Abstract

A Security Enhanced AODV Routing Protocol

for Wireless Mesh Networks

© Zhuang Wang 2009

Master of Applied Science

Department of Electrical and Computer Engineering

Ryerson University

This thesis introduces a Security Enhanced AODV routing protocol for wireless mesh networks (WMNs).

SEAODV employs Blom's key pre-distribution scheme to compute the pairwise transient key (PTK) through the flooding of enhanced HELLO message and subsequently uses established PTK to distribute the group transient key (GTK). PTK and GTK are used for authenticating unicast and broadcast routing message respectively. In WMNs, a unique PTK is shared by each pair of nodes, while GTK is shared secretly between the node and all its one-hop neighbours. A message authentication code (MAC) is attached as the extension to the original AODV routing message to guarantee the message's authenticity and integrity in a hop-by-hop fashion.

Security analysis and performance evaluation show that SEAODV is more effective in preventing identified routing attacks and outperforms ARAN and SAODV in terms of computation cost and route acquisition latency.

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Dr. Cungang Yang not only for providing me with valuable feedback and support but also for selflessly sharing his knowledge in this field with me.

Also, I am grateful for the financial assistance that I have received through my supervisor and the Electrical & Computer Engineering department at Ryerson University. In addition, I would appreciate Dr. Olivia Das, Dr. Cherie Ding, Dr. Bobby Ma and other reviewers' valuable comments.

Finally, I would like to thank my family, especially my wife, my parents and other friends for their kindness and for supporting me through this and all phases of my life. I sincerely appreciate all that you have done for me and many thanks for believing in me and encouraging me to reach my goals.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Wireless Mesh Networks (WMNs) are an emerging technology for next generation wireless networks. The interconnection of access points using wireless links presents significant potential in addressing the "last mile" connectivity issue. Although a profusion of routing protocols has been proposed for other wireless networks such as Ad hoc network, the unique characteristics of WMNs indicate that WMNs demand its own solution. Routing security in WMNs has rarely been addressed so far. Hybrid routing seems to be one of the promising answers to the question of what is the trend in WMNs' routing. In hybrid routing, proactive routing is specifically used for traffics flow to the mesh portal, where a connection with Internet was created. While for intra-mesh traffic, which means traffics will not bypass the mesh portal, on-demand routing is preferred. In hybrid routing of WMNs, such as HWMP [11], [12] and [13], our proposed scheme can be seen as a secure version of the on-demand part in HWMP and used to securely discover a route between any pair of mesh routers in the network.

Our research is focused on:

• How to make the route discovery process of AODV secure between any two wireless nodes in WMNs?

• How to defend against different attacking scenarios that might have occurred in AODV by using our proposed SEAODV?

• How to establish the Pairwise Transient Key (PTK) and distribute Group Transient Key (GTK) in WMNs?

• How secure is SEAODV in contrast to SAODV and ARAN in terms of how effectively defending against identified attacking scenarios?

• How efficient is SEAODV as compares to SAODV and ARAN in terms of computation cost, communication cost and route acquisition delay?

To solve the above questions, in this thesis:

• We utilize PTK and GTK keys to protect the unicast and broadcast routing message respectively to ensure that the route discovery process between any two nodes in WMNs is secure.

• We identify different attacking scenarios specifically in AODV and present security analysis to prove that our proposed SEAODV is able to effectively defend against most of these attacks while the number of attacks that SAODV and ARAN can defend against is less.

• We apply BLOM's key pre-distribution scheme in conjunction with the enhanced HELLO message to establish the PTK and use the established PTK to distribute GTK to the node's one-hop neighbours throughout the entire network. In WMNs, nodes are normally fixed or tend to experience a less changing environment; therefore, BLOM's scheme is suitable for WMNs due to less overhead incurred in the key generation/exchange process and easy planning of N, which is the total number of fixed mesh routers in the WMNs.

• We perform evaluation on SAODV, ARAN and SEAODV and testify that SEAODV is superior to other two in terms of computation cost and route acquisition delay, which indicates that SEAODV is also energy efficient and the route is expected to survive longer if a certain number of mesh clients on the established route are involved in relaying the packets.

1.2 Contribution

The contribution of this thesis is given as follows.

- We identify and summarize various attacking scenarios in AODV and proposed a Security Enhanced AODV routing protocol, named SEAODV, which is a variant of AODV, can effectively defend against the identified attacks.

- We incorporate Blom's key pre-distribution scheme into our proposed SEAODV. Blom's scheme is secure, lightweight and offers great scalability.

- In SEAODV, every node possesses two types of keys: PTK and GTK, which are used to secure unicast and broadcast routing messages respectively.

- Our Scheme is lightweight and computationally efficient since only symmetric cryptographic operations (in this case, MAC) are involved in SEAODV. SEAODV implements a hop-by-hop authentication.

- Through security analysis, we conclude that SEAODV presents better immunity to the identified attacks as compares to other secure routing protocols such as ARAN, SAODV and LHAP.

- Performance evaluation proved that SEAODV incurs less computation cost and offers better route acquisition latency in contrast to other two secure routing protocols ARAN and SAODV.

- SEAODV can not only offer secure route setup on a hop-by-hop basis, but also guarantee safe delivery of the subsequent data traffics by using the PTK in each node.

1.3 Related Work

So far, there has been tremendous research on layer 3 secure routing for wireless networks. Each secure routing protocol is tailored to a specific type of wireless networks, such as Ad hoc networks, wireless sensor networks and wireless mesh networks. All of them have similar properties, therefore, some secure routing protocols for Ad hoc networks can also be applied to wireless mesh networks. However, some of them do not provide enough security features (such as hop-by-hop authentication) and are still vulnerable to various types of routing attacks such as flooding, route re-direction, spoofing etc.

Depending on when routes are required to be calculated, routing protocols can be divided into two categories: Proactive routing and On-demand routing.

In proactive routing, every node maintains one or more tables containing routing information to every other node in the network. All nodes in the network update their tables to maintain a consistent and up-to-date view of the network whenever the network topology changes or a node's routing table is updated. Example of proactive routing in WMNs is Link Quality Source Routing (LQSR) [1], which giving minimum burden on relaying nodes as the source node calculates the route for a flow and stores the complete path of the flow in its packet headers. Intermediates nodes only need to forward the packets according to the path stored in the packet headers.

On-demand routing originally proposed for Ad hoc network, it is also called reactive routing. Ad hoc On-demand Distance Vector routing (AODV) [2], Dynamic Source Routing (DSR) [3], Load Balancing Aware Routing (LBAR) [4] and Dynamic Load Aware Routing (DLAR) [5] are all belong to on-demand routing protocols. A route is only being created when the source node actually needs to send packets to the destination.

SAODV [6] is a secure variant of AODV and operates in a very similar way to that of AODV does. AODV uses hash chain and digital signature to secure the so called mutable field (hop count) and non-mutable field (the rest of the routing message except hop count field) and successfully defends against some attacks. For instance, impersonation attacks, modification of hop count and sequence number attacks. However, it is not based on hop-by-hop authentication, it is source authentication based. Intermediate nodes on the path cannot verify the authenticity of the messages from their predecessors.

Although SAODV can prevent the hop count field in AODV routing message from decreasing, the adversary can still increase hop count and therefore, affect the routing decision of which node is going to be selected during route discovery process and increase the likelihood of nodes not being chosen on the established route.

SAODV only presents how to secure the routing messages; however, after route has been established between source and destination, it does not guarantee either authentication or integrity of the subsequent data packets. However, our proposed SEAODV can still make use of PTK key to secure the subsequent data packets since we believe that securing data packets is also important under some circumstances.

ARAN (Authenticate Routing for Ad hoc Networks) [7] adopts digital signature to ensure hop-by-hop authentication and routing message integrity; however, it suffers from experiencing significant computation overheads that dramatically affect its routing performance (such as route acquisition latency).

Each node in the network has to verify signatures every single time it receives the signed messages, removes the certificate and signature of its predecessor, use its own private key to sign the message originally broadcast by the source and appends its own certificate before rebroadcasting to its one-hop neighbours. Whenever security is presented in a routing protocol, it is all about the trade-off between the security level achieved and its routing performance. In our protocol, we use pri-distributed shared keys based on symmetric cryptography to make SEAODV lightweight and efficient. And we believe that lightweight should always be considered with the first priority in an energy constrained ad hoc network, which forms the mobile part of the hybrid WMNs.

Both SAODV and ARAN cannot prevent and defend the DoS (Denial of Service) attack. For example, when a node receives a routing message, it has to verify the signature first in order to perform further actions. Adversaries can utilize this flaw to inject a large number of faked or replayed signed messages in a short period and try to intentionally make the designated node repeatedly verify the signatures. As a consequence, the node will soon be dead due to energy depletion. Our proposed SEAODV tends to be more resistant against this type of attack due to efficiency of using the symmetric cryptography, which means it is relatively more time consuming for attackers to effectively bring the node down.

Ariande in [8] is a secure on demand source routing. It uses TESLA [9], digital signatures and standard MAC to ensure the source authentication; however, the route request will not be authenticated until it reaches the destination, which is the drawback of Ariande since adversary could initiate route request flooding attack.

A variant of Ariande named endairA is proposed in [10] with a difference that instead of signing a route request, intermediate nodes are able to sign the route reply. It experiences less cryptographic computation, but are still vulnerable to malicious route request flooding attack.

[11], [12] and [13] introduce a hybrid routing protocol called Hybrid Wireless Mesh Protocol (HWMP). In HWMP, on-demand mode allows two MPs (Mesh Point) to communicate using peer-to-peer paths. This mode is primarily used by nodes that experience a changing environment and when there is no root MP configured. While the proactive tree building mode can be an efficient choice for nodes in a fixed network topology. However, HWMP does not address the security issues and suffers from different type of attacking scenarios described in chapter 4. In addition to this, since HWMP is a hybrid version of the on-demand and proactive mode, each mode needs to be considered separately to make the hybrid routing secured.

LHAP [14] is indeed a lightweight transparent authentication protocol for Ad hoc networks that resides between the data link and the network layer. It uses TESLA to maintain the trust relationship among nodes, which is not a realistic approach because of the delayed key disclosure period in TESLA. Also in LHAP, simply attaching the TRAFFIC key right after the raw message is not secure enough since the traffic key has no relationship at all with the message being transmitted. LHAP also experiences various vulnerabilities as given in chapter 4.

The thesis is composed of eight chapters and the remainder is organized as follows:

Chapter 2	Introduces the background knowledge of cryptography.
Chapter 3	Presents security in Wireless Mesh Networks. Reviews the original standard AODV routing protocol and two well known secure routing protocols, SAODV and ARAN.
Chapter 4	Identifies various attacking scenarios that might have appeared in AODV.
Chapter 5	Outlines background of Blom' key pre-distribution scheme.
Chapter 6	Presents our proposed security enhanced AODV (SEAODV) in details.
Chapter 7	Discusses security analysis and presents performance evaluation among various secure routing protocols.
Chapter 8	Provides conclusions and future work.

Chapter 2

Background of Cryptography

2.1 Introduction

With the emergence of wireless networking technology, it is becoming more and more important for people to protect their own privacy from being corrupting by unauthorized users. Cryptography has become as an enormously effective tool to combat with different types of attacks that could have been launched by malicious "people" in order to gain a certain level of payoff.

Different cryptography techniques are applied on various fields, such as data encryption/decryption and data authentication to achieve the confidentiality, authenticity and integrity of the traffic transmissions over an unsecure wireless channel.

Cryptography has a long and fascinating history; it is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication [15]. Cryptography is not the only means of providing information security, but rather one set of techniques.

Today to most people, cryptography is all about how to keep communications private. This is the truth and has been emphasized all the time throughout the history of cryptography. However, it is only a part of the cryptography.

Encryption transforms original information, called plaintext or cleartext, into transformed information, called ciphertext, codetext or simply cipher, which usually has the appearance of random, unintelligible data. The transformed information, in its encrypted form, is called the cryptogram [16]. Encryption provides confidentiality, integrity and authenticity of information being transferred from A to B over an unsecure channel. It guarantees that the transmitted information has not been tampered and the received information is authentic, which means it indeed comes from A, who is the originator of the received information at B. Just as its name implies, decryption is the reverse process of the encryption and it transforms the encrypted data back into an intelligible form.

In general, there exist two types of encryptions: public-key encryption and secret key (also called symmetric key) encryption.

Both encryption and decryption are required to use a key, which is kept secretly. Some encryption mechanisms apply the same key for performing the encryption and decryption, others require different keys.

Key establishment is any process whereby a shared secret key becomes available to two or more parties, for subsequent cryptographic use. Key management is the set of processes and mechanisms which support key establishment and the maintenance of ongoing keying relationships between parties, including replacing older keys with new keys as necessary [15].

The remainder of this chapter is organized as follows.

Section 2.2 Introduces key management, including public key and secret key cryptography and the comparison between the two.

Section 2.3 Presents RSA Encryption Decryption Algorithm.

Section 2.4 Briefly introduces data authentication techniques such as hash function and message authentication code (MAC).

2.2 Key Management

There are two types of cryptosystems: public-key and secret-key cryptography. Secret-key cryptography is also called symmetric cryptography since the same key is used for both encryption and decryption. Advanced Encryption Standard (AES) is one of the popular symmetric cryptographies used today.

While in public-key cryptography, every user owns a pair of keys, one is public key and the other is private. The public key is known and made public; however, the private key remains secret. Encryption is done by using the public key while decryption is accomplished by applying the private key. The famous and popular public-key cryptosystem is called RSA, which stands for Rivest, Shamir, and Adleman, the inventors of the RSA cryptosystem.

Key management is the provisions made in a cryptography system design that are related to generation, exchange, storage, safeguarding, use, vetting, and replacement of keys. It includes cryptographic protocol design, key servers, user procedures, and other relevant protocols [17].

Key management deals with keys at the user level, either between users or systems. Successful key management is crucial to the security of a cryptosystem. It is the process of generating, certifying, distributing and revoking the keys.

Once a key is generated randomly, it must be kept secret to avoid the misuse by other attackers (for example impersonation). Also a certificate is used to let users be able to legitimately obtain others' public keys. This is based on the fact that the issuer must be impervious to attacks, the issuance of certificates must proceed in a secure way and the individual that obtains the certificate must be authentic.

Successful key management is critical to the security of a cryptosystem. In practice it is arguably the most difficult aspect of cryptography because it involves system policy, user training, organizational and departmental interactions, and coordination between all of these elements [17].

2.2.1 Public Key

The public key cryptography is invented to overcome some drawbacks of secret-key cryptography. In secret key encryption/decryption, keys are shared secretly between the sender and receiver. Therefore, there must be a secure channel between sender and receiver to ensure that the secret key is impossible to be intercepted or eavesdropped by adversary during the transit over the transmission medium. Adversary who obtains the key in transit can subsequently read, modify and forge all messages encrypted/decrypted using that key. To defend against this key management attack, Whitfield Diffie and Martin Hellman introduced the concept of public-key cryptography in 1976.

The great benefit by using public key cryptography is that now both sender and receiver do not need to share secret information, secret keys are never transmitted or shared among nodes in the network. Anyone can use the public key to encrypt a confidential message, but only the sole possession of the intended recipient has the corresponding paired secret key to decrypt the message.

Moreover, public-key cryptography can also be used not only for privacy (encryption), but also for authentication (digital signatures) and other various techniques.

The two main branches of public key cryptography are given below [18]:

- Public key encryption — a message encrypted with a recipient's public key cannot be decrypted by anyone except a possessor of the matching private key -- presumably, this will be the owner of that key and the person associated with the public key used. This is used for confidentiality.

The figure below presents the sketch map of the public key encryption.

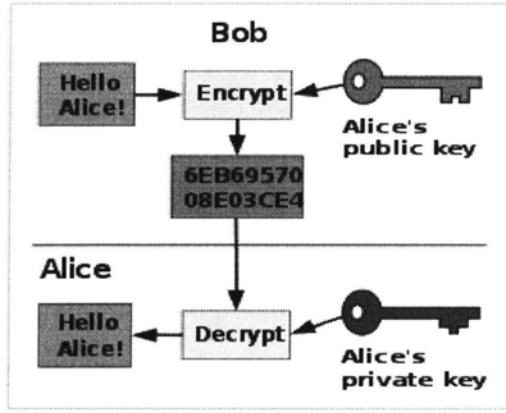


Fig. 2. 1 Public Key Encryption

In the above sketch map [18], anyone can encrypt messages by using the public key, but only the holder of the paired private key can decrypt. Security relies on the secrecy of that private key.

- Digital signatures — a message signed with a sender's private key can be verified by anyone who has access to the sender's public key, thereby proving that the sender had access to the private key (and therefore is likely to be the person associated with the public key used), and the part of the message that has not been tampered with.

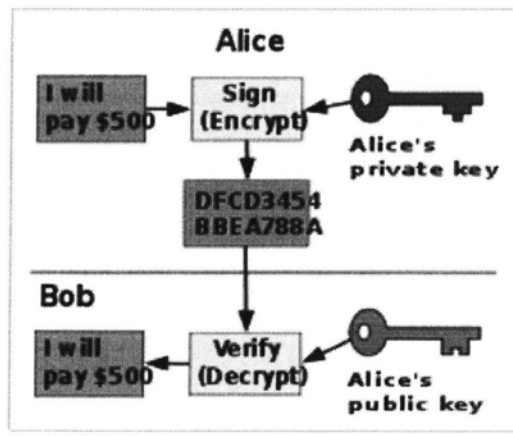


Fig. 2. 2 Digital Signature

The above figure [18] illustrates a sketch of the digital signature, which applies private key to sign a message and anyone who possesses the public key can verify the signature. Validity depends on private key security. In other words, Bob now can use Alice's public key to decrypt the signed message comes from Alice. Since Alice is the only legitimate possessor of the private key, the successful verification indicates that Alice indeed is the originator of the signed message, hence authenticity is guaranteed.

2.2.2 Secret Key

An encryption/decryption system in which the sender and receiver of a message share a single, common key that is used to encrypt and decrypt the message, while the public-key cryptography utilizes two keys - a public key to encrypt messages and a private key to decrypt them.

Symmetric-key cryptography is sometimes referred to as secret-key cryptography. The most popular symmetric-key system is the Data Encryption Standard (DES).

Symmetric-key systems are simpler, faster and easy to use, but the main drawback is that the two parties have to somehow exchange the key in a secure way. Public-key encryption avoids this problem since the public key can be distributed in a non-secure way, and the private key is never transmitted and exposed to adversary. The popular techniques used in secret-key cryptography include block ciphers, stream ciphers, composition of ciphers and MAC (message authentication code).

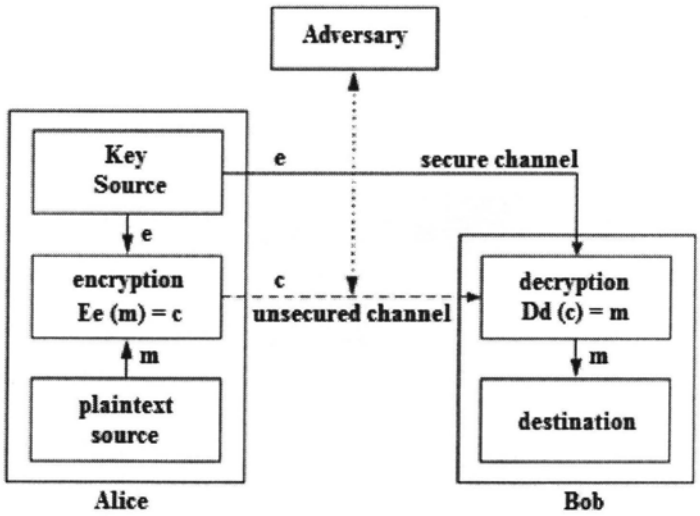


Fig. 2. 3 A Two-party Communication Using Symmetric Key Encryption

The above figure [15] shows an example of two-party communication by using symmetric key encryption. The decryption key d can be efficiently computed from the encryption key e or d may be identical to e. Therefore, Alice and Bob use the same secret key to perform the encryption/decryption on message m.

2.2.3 Public-key vs. Secret-key cryptography

This section highlights various advantages and disadvantages for public-key and secret-key cryptography respectively, although some of which are common to both.

Table 2.1 given blow summarizes the advantages/disadvantages for both the public and secret cryptography. In practice, public-key and secret-key encryption are complementary to each other. Public-key encryption techniques can be used to accomplish the secure key establishment for a secret-key cryptosystem. Public-key cryptography also provides digital signature, which offers non-repudiation. And secret-key cryptography is efficient for encryption and some applications that provide data integrity.

	Public-key cryptography	Secret-key cryptography
Advantages	Public/private key pair might remain unchanged for a certain period	Keys are relatively short
	The number of key pairs required for a large network could be significantly smaller than that of secret-key cryptography	Low computation cost, fast
	Only private key needs to be kept secret	Simple and straightforward
	Can provide digital signature that guarantees non-repudiation	Keys can be used to construct stronger ciphers
Disadvantages	High computation cost, slow	Keys must be kept secret at both ends
	Larger key size	The larger the network, the more key pairs are required and managed. Key management issue arises
	No public-key scheme has been proven to be secure	Keys need to be changed frequently

	May be vulnerable to impersonation attack due to the compromise of certification authority	Secure key establishment is still an ongoing research topic

Tab. 2. 1 Public-key Vs. Secret-key Cryptography

2.3 RSA Encryption Decryption Algorithm

The RSA algorithm was publicly described in 1978 by Ron Rivest, Adi Shamir, and Leonard Adleman at MIT; the letters RSA stands for the initials of their surnames. RSA offers both encryption and digital signatures.

RSA algorithm includes three steps: key generation, encryption and decryption.

Key generation

1. Take two distinct prime number p and q, make sure both of them are large enough;
2. Compute their product by using $n = pq$; where n is called modulus used for both public and private keys;
3. Compute the totient as $\varphi(n) = (p-1)(q-1)$;
4. Take an integer e such that $1 < e < \varphi(n)$ and e is relatively prime to $\varphi(n)$, which means e and $\varphi(n)$ share no common factors but 1. Where e is defined as the public key exponent;
5. Determine another integer number d such that (ed-1) can be evenly divided by the totient $\varphi(n)$, where d is called the private key exponent and must be kept secret;

After key generation process, two pairs of key have been created. The public key pair (n, e) and the private key pair (n, d) respectively.

The message m can be encrypted and decrypted by using the following equations.

Encryption

$$c \equiv m^e \ (mod\ n)$$

Whenever Bob wants to send a message to Alice, he applies the public key pair (n, e) received from Alice to encrypt the message and sends the ciphertext c to Alice.

Decryption

$$m \equiv c^d \ (mod\ n)$$

Once Alice receives c, she uses her own private key pair (n, d) to decrypt the received ciphertext and recovers message m, which is the plaintext.

The security of RSA replies on the assumption that factoring is reasonably difficult, which means one could not possibly factor n into p and q, therefore, the security of the private key pair (n, d) is guaranteed. This ensures that only the possessor of the private key pair can decrypt the message correctly.

There always has been a mathematical relationship between public key and private key pair. Therefore, adversaries can possibly attack the public-key cryptosystem by deducing the private key pair from the public key pair. How to improve the security of the public-key based cryptosystem turns out to be making the derivation of the private key pair as difficult as possible.

RSA algorithm can also be used for signing message, which is called digital signature as mentioned earlier. Suppose Bob wants to send a signed message to Alice, he can use his own private key pair to do so. He can first produce a hash value of the message, then using the decryption equation defined above to sign the hashed message; finally attach it as a "signature" to the original message. Upon receiving the signed message, Alice applies the same hash function in conjunction with the public key pair to the signed message (as she does when encrypting a message) and compares the resulting message with the actual message. If the two are identical, then Alice knows that the author of the message is the holder of Bob's private key and the message has not been tampered ever since.

2.4 Data Authentication

Informally, data authentication allows a receiver to verify that the data really was sent by the claimed sender.

In the two-party communication case, data authentication can be achieved through a purely symmetric mechanism: The sender and the receiver share a secret key to compute a message authentication code

(MAC) of all communicated data. When a message with a correct MAC arrives, the receiver knows that it must have been sent by the sender.

Data authentication is important for many applications in wireless mesh networks. Adversary could simply inject spurious message into the network, therefore, receiver has to ensure that the data used in received message originates from the correct source and has been unaltered. In a two-party communication case, data authentication can be achieved by using hash functions and message authentication code (MAC) or the combination of both.

The following two sub-sections present more details for hash function and message authentication code.

2.4.1 Hash Function

A hash function is a deterministic procedure that takes arbitrary block of data and returns a fixed-sized bit string, the hash value, such that an accidental or intentional change to the data will change the hash value [19]. A hash function H is said to be one-way if it is hard to invert, where "hard to invert" indicates that given a hash value h, it is computationally infeasible to find some input x such that $H(x) = h$.

Hash function has its unique properties as given below.

1. It is computationally infeasible to find two different messages with the same hash result;
2. It is computationally infeasible to change the hash result without modifying the message;
3. Giving a known hash, it is computationally infeasible to find a message that matches the hash;
4. Giving a raw message, it is easy to compute the hash.

Examples of well known hash functions are MD2, MD5 and SHA-1. Hash function can be used in many security applications such as digital signature, integrity checks, message authentication code (MAC) and others forms of authentication. For example, in order to verify the integrity of a message, a comparison can be made between the hash values calculated before and after to determine whether any changes have been made to a message during transmission.

2.4.2 MAC

Message authentication code (MAC) is sometimes called a keyed hash function since it is constructed by using a secret key in conjunction with a hash function. MAC is also known as a tag and it offers both the

integrity and the authenticity of a massage simultaneously by allowing the receiver who possesses the shared secret key to detect whether any changes have been made to the message.

Hash function can be used to create MAC functions; a typical example of this is called HMAC. Unlike digital signatures, MAC is computed and verified by applying the same secret key at both ends so that only recipient who shares the secret with the sender can successfully verify the MAC. However, MAC does not provide non-repudiation offered by digital signature since any user who can verify a MAC is also able to generate MAC for other messages. While a digital signature ensures that no one other than the private key holder can sign the message because the private key is only accessible to the key holder. Hence, non-repudiation is guaranteed by digital signature.

The figure [20] below shows an example of MAC process. The sender applies the secret key in conjunction with its MAC algorithm to compute the MAC, which in turn has been attached to the message. At the other end, the receiver runs the message portion of the received packet by applying the same MAC algorithm and secret key in order to produce a second MAC. The receiver then can compare the first MAC it obtained from the sender and the second MAC it computes from itself. If the two MACs are identical, the receiver now can assure that the message has not been compromised and altered during transmission and it indeed originates from the sender whoever it claims to be.

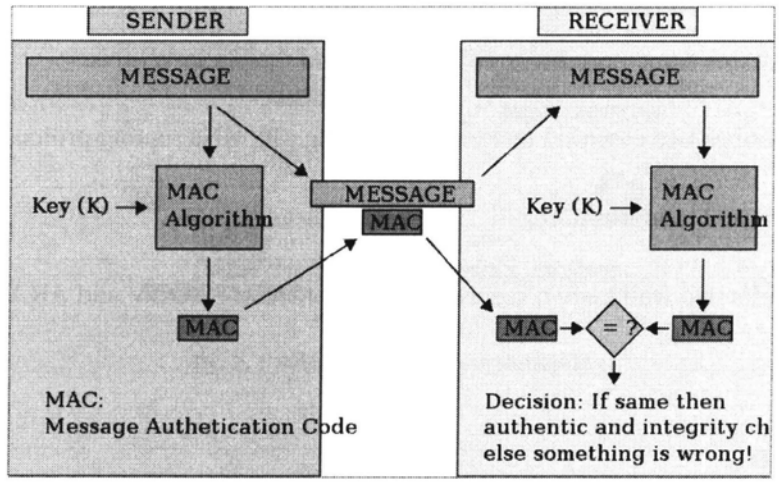


Fig. 2. 4 An Example of MAC Process

Chapter 3

Security in Wireless Mesh Networks

Nowadays, security has been considered as one of the most critical parameters for the acceptance of any wireless networking technology. However, security in wireless mesh networks (WMNs) is still an interesting and promising research topic as little attention and research has been done to this field by the research community.

The remainder of this chapter is structured as follows.

Section 3.1 Briefly introduces wireless mesh networks (WMNs).

Section 3.2 Outlines the security requirements of WMNs.

Section 3.3 Presents some routing protocols developed particularly for WMNs.

Section 3.4 Discusses some possible routing attacks and threats in WMNs.

Section 3.5 Describes two essential factors in designing a WMNs routing protocol.

Section 3.6 Reviews Standard AODV.

Section 3.7 Recalls two well known secure routing protocols: SAODV and ARAN.

3.1 Introduction

Wireless Mesh Networks (WMNs) [21] [22] [23] are a wireless multi-hop technology that has much in common with the mobile ad hoc networks (MANETs).

Wireless Mesh Network can be considered as a superset of the ad hoc networks. Unlike the traditional wireless LAN, where each node is connected to the AP (access point) via a single hop wireless link and the AP interconnects with the wired backbone network, the WMN does not rely on such wired backbone and constructs the wireless network through multi-hop wireless links. The low cost, fast deployment, self

configuring, self organizing features make WMNs a very potentially appealing candidate for a wide range of applications, such as military tasks, emergency response etc.

There are two types of nodes in the WMNs. One is called the MR (Mesh Router); the other is named MC (Mesh Client). Conventionally, mesh routers are the nodes with no power constrain and tend to be static with even zero mobility in the network. On the other hand, mesh clients only present limited battery power and are normally mobile client devices such as Wi-Fi enabled PDA.

An infrastructure WMNs is one that is entirely comprised of mesh routers, whereas a client WMNs is a network purely made up of mobile client devices. In most cases, a typical WMN is a hybrid network where both mesh routers and mesh clients exist simultaneously. The following figure [11] shows a typical hybrid WMN.

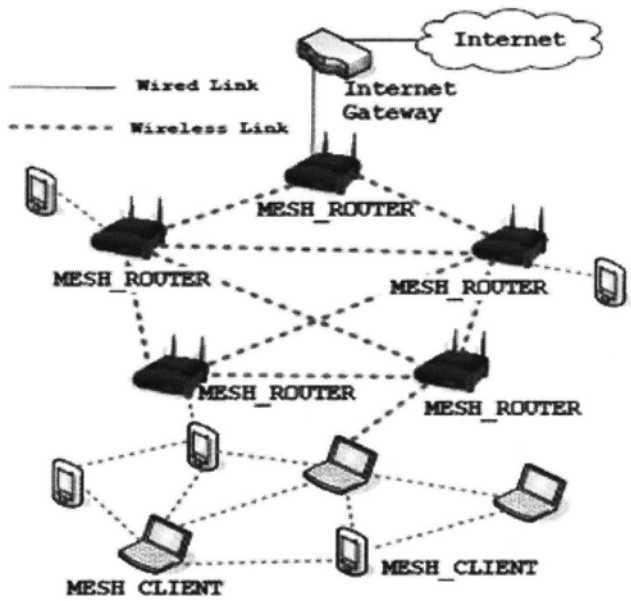


Fig. 3. 1 Hybrid Wireless Mesh Network

3.2 Security Requirements of WMNs

To ensure the security of WMNs, the following security objectives must be met.

a. Confidentiality

   It indicates that certain amount of information is only accessible to those who have been authorized to access it. Certain information such as secret keys is never disclosed to unauthorized users.

b. Integrity

Integrity ensures that information being transmitted from one end to the other end has remained unchanged. Information cannot be corrupted during transmission.

c. Availability

Availability guarantees the survivability of network services. This security requirement is mainly challenged by the so called DoS (Denial of Service) attacks.

d. Authenticity

It ensures that the participants in communication are genuine and not the impersonators. Security techniques such as digital signature can guarantee authenticity.

e. Non-repudiation

Non-repudiation simply means that both the sender and the receiver of a message cannot deny that they have ever sent or received such a message. It is useful to identify and isolate the misbehaved nodes in the network.

f. Authorization

It is simply a process in which an entity is given a credential by a trusted certificate authority. It is mainly used to assign different access rights to different level of users.

3.3 Routing in WMNs

WMNs and Ad hoc network share some common characteristics and features. Therefore, some of the routing protocols can also apply to WMNs. However, these routing protocols might not fit well in terms of WMNs' own characteristics. Only a few routing protocols have been developed specifically for WMNs. MIT [24] (SrcRR) and Meshnetworks Scalable Routing [25] (MSR) are routing protocols specially designed for WMNs. MSR is a hybrid routing protocol that can accommodate highly mobile users and dynamically adapt to network conditions.

802.11s is a HWMP (Hybrid Wireless Mesh Protocol) designed for WMNs; its routing function consists of two parts, proactive and reactive respectively. It is a hybrid version of both the proactive and reactive routing. For the mesh routers in WMN, they tend to be relative static; hence proactive routing protocols are the best candidates for them. On the other hand, reactive routing protocols are more suitable for highly mobile mesh clients in the WMNs. However, this will certainly bring additional complexity of implementing the routing protocol.

3.4 Possible Routing Attacks and Threats in WMNs

The following lists some of the possible attacks that could have been launched by adversaries in WMNs.

a. DoS attack

   There are so many types of DoS attack and it can happen at any layer of the network. On the physical layer, an adversary could simply apply jamming signal to interfere with communications on physical channel. At the network layer, an attacker can create routes to nonexistent nodes with the intention to create enough routes in order to prevent new routes from being created in the routing table. This is also called routing table overflow attack;

b. Wormhole attack

   Normally this type of attack requires two colluding nodes to sit at two different locations in the network. Upon receiving packets at one location, the node tunnels them to the other colluding node who locates elsewhere in the network. The subsequent packets are tunnelled between the two colluding nodes and the tunnel is referred to as a wormhole;

c. Sinkhole/blackhole attack

   Malicious node uses the routing protocol to advertize that it has shorter path or whatever better metrics used in the routing protocol to lure subsequent traffics through it;

d. Location disclosure attack

   This attack reveals information about network topology or even location of nodes in the network.

There might have some other possible routing attacks in WMNs and our SEAODV builds on the original version of AODV, in this case, the pivot of discussion focuses on the security flaws and relevant attacking scenarios AODV could have possibly faced, which are discussed in chapter 5.

3.5 Two essential Factors in Designing a WMNs Routing Protocol

When designing a routing protocol for WMNs, the following two essential factors in designing must be considered.

1. Which performance metrics?

   Which performance metric is going to be used in the routing protocol? The routing protocol can focus on optimizing one or even more performance metrics based on different characteristics of the network. For an instance, our SEAODV uses hop count (the number of hops between the source and destination) in making routing decisions. Some routing protocols [26] and [27] use expected transmission count (ETX) as its routing metric, where the data loss rate due to the

medium contention and environmental factors are considered. Based on the data loss rate, the number of retransmissions needed to successfully deliver a packet is calculated. While [28] uses expected transmission time (ETT) to further include the link bandwidth in its computation.

2. Proactive, reactive or hybrid routing protocol?

   In general, routing protocol for WMN can be classified into two categories. The first category consists of traditional routing protocols for wired networks such as RIP (Routing Information Protocol) or OSPF (Open Shortest Path First). These protocols do not aim at handling highly dynamic networks with high node mobility; therefore, their applications are limited to relatively static infrastructure WMNs.

   The other category consists of protocols that are based on MANET routing protocols. These protocols have been designed for highly dynamic networks made up of relatively high mobility and power constrained nodes. In some cases, the number of updates may not be distributed to its neighbours fast enough due to the channel contentions between control traffic and data traffic, hence leading to non-optimal routing decisions. As a consequence, a hybrid routing protocol seems to be better satisfied the requirements of the hybrid WMNs and has drawn so much attentions by the researchers.

3.6 Overview of Standard AODV

The Ad Hoc On-Demand Distance Vector (AODV) algorithm is an on-demand reactive routing protocol, which means it seeks for routes only when required. AODV makes use of sequence numbers to avoid forming routing loops; it also enables mobile nodes to find routes quickly for new destinations and nodes are only required to maintain the routes to those destination that are currently in active communication.

The standard operations of AODV are described as follows.

Whenever a source node wants to communicate with a new destination, it broadcasts a RREQ (Route Request) message to all its one-hop neighbours under the circumstance that the node could not find an active route in its routing table. Upon receiving the broadcasted messages, its neighbours then check their routing tables to see whether there exists a fresh enough route to the destination node. If not, they then will continue to broadcast the RREQ message to their neighbours until the RREQ reaches the designated destination or an intermediate node that already has a fresh enough route to the destination. If yes, which means either the broadcast routing message reaches the desired destination or an intermediate node has a

fresh enough route to the destination. In that case, the destination node or the intermediate node updates its reverse route to the source node from which they received the RREQ, generates a RREP (Route Reply) message and unicasts it back to the source node. When the source node or the intermediate node receives a RREP message, they will update their forward route to the destination, using the neighbours from which they receive their RREP and update their routing tables accordingly.

To maintain connectivity information, each node executing AODV can use periodical HELLO message to detect any link breakage to its immediate neighbours. In the case where a broken link is detected for the next hop of an active route, a RERR (route error message) is sent to all its neighbours that were using that specific route. Details of AODV can be found in [2].

3.7 Two Known Secure Routing Protocols

ARAN (Authenticated Routing for Ad hoc Networks) and SAODV (Secure Ad hoc On-demand Distance Vector) are two well known secure routing protocols for Ad hoc networks. Both of them are based on AODV, although ARAN presents different message format in route discovery process, it still applies the basic methodology of AODV.

3.7.1 SOADV

SAODV is a secure version of AODV and it protects the routing message of the original AODV. SAODV uses hash chains to secure hop count field and digital signatures to protect the non-mutable fields in both RREQ and RREP messages. The following explains how SAODV works in details.

During route discovery process, a random $seed$ number is generated by the source node and the maximum hop count (MHC) value is also set to be the TimeToLive (TTL) value from the IP header. Source node then computes the hash value as $Hash = h(seed)$ and Top_{Hash} as $h^{MHC}(seed)$.

Upon receiving an RREQ message, an intermediate node verifies whether the value of Top_{Hash} equals to $h^{MHC-Hop_{Count}}(Hash)$. If the two values are completely identical, the hop count is presumed to be unaltered. Before rebroadcasting the RREQ to its one-hop neighbours, the intermediate node increases the hop count by one and computes the new Hash value as $h(Hash)$. Except the hop count field, all other fields in the RREQ are considered to be non-mutable and secured by using digital signature. When RREQ meets the destination, destination generates a RREP in the same way towards to the source.

3.7.2 ARAN

ARAN uses digital signatures only in discovering routes. The entire routing message is signed both by the source and the intermediate node who forwards the RREQ or RREP. ARAN offers authentication, message integrity and non-repudiation.

ARAN requires the use of a trusted certificate server T, whose public key is known to all legitimate nodes in the network. Each valid node receives a certificate from T as described below [7].

$$T \rightarrow A : cert_A = [IP_A, K_{A+}, t, e]K_{T-} \quad (1)$$

Where a certificate includes A's IP address, A's public key, a timestamp t (when the certificate was generated) and a time e which indicates when the certificate expires.

Whenever a source (in this case, node A) needs to seek a route, it generates a RREQ as follows [7].

$$A \rightarrow brdcast : [RDP, IP_X, cert_A, N_A, t]K_{A-} \quad (2)$$

The complete routing message (RREQ) has been signed by A's private key. The signed RREQ contains the route discovery packet (RDP), destination address IPx, certificate A, a monotonically increased nonce and a timestamp at which the RREQ was signed.

Upon receiving the RREQ by an intermediate node (in this case, node B), node B uses the pubic key of certificate server T to extract A's public key and applies this key to verify the received signed message. If the certificate is verified to be still valid and the signature's verification is successful, the received message is considered to be authentic and unaltered. Node B then updates the routing table accordingly, signs it and appends its own certificate before rebroadcasting the signed message to its one-hop neighbours. The verification and signing process of node B is as follows [7].

$$B \rightarrow brdcast : \left[[RDP, IP_X, cert_A, N_A, t]K_{A-}\right]K_{B-}, cert_B \quad (3)$$

Assuming node C who is one of node B's one-hop neighbours receives the RREQ from node B. Node C validates the corresponding signature with the given certificate. Upon successfully verifying the signature, C then removes B's certificate and signature, updates routing table, signs the entire message originally broadcast by node A and appends its own certificate before rebroadcasting. The broadcast message is given below [7].

$$C \rightarrow brdcast : \left[[RDP, IP_X, cert_A, N_A, t]K_{A-}\right]K_{C-}, cert_C \quad (4)$$

During the route discovery, each intermediate node repeats the steps mentioned above. These include verifying the previous node's signature, removing previous node's certificate and signature, updating routing table, signing the original contents of the message and appends its own certificate at the end of the message before rebroadcasting.

When the destination node receives the RREQ, it creates an RREP and unicasts it back along the reverse path to the source in the same way.

Chapter 4

Attacking Scenarios in AODV

The following sections present possible attacks that could have been launched in AODV during a route discovery process.

The remainder of this chapter is structured as follows.

Section 4.1 Outlines RREQ flooding attack.

Section 4.2 Discusses RREP routing loop attack.

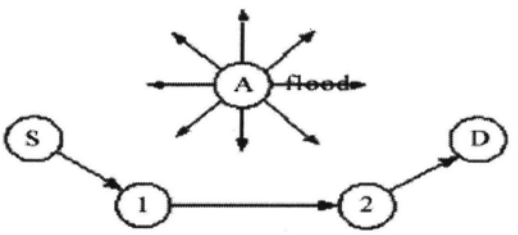
Section 4.3 Presents route re-direction attack.

Section 4.4 Explains formation of routing loops attack.

Section 4.5 Depicts fabrication attack.

Section 4.6 Describes Tunnelling Attack.

4.1 RREQ Flooding Attack



a) flood attack by malicious node.

Fig. 4. 1 RREQ Flooding Attack

Flooding is one of the simplest attacks that a malicious node could have launched. An attacker tries to flood the entire network with RREQ message destines to a known or an unknown address as shown in Fig. 4.1 [29]. As a consequence, this causes a mass of unnecessary broadcasts and force the neighbours to process these flooding route requests, the aim is to consume the energy of the nodes in the network and the network bandwidth. Therefore, the whole network communication may be breakdown and the throughput is dropped dramatically.

4.2 RREP Routing Loop Attack

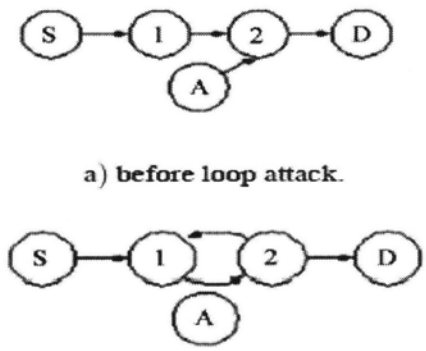


Fig. 4. 2 RREP Routing Loop Attack

The figure [29] above shows the RREP routing loop attack. A routing loop is a path that goes through the same nodes over and over again. As a consequence, this kind of attack will deplete the resources of every node in the loop and cause the isolation of the destination and few packets can eventually reach the destination.

From the above figure, node1 and node2 are the intermediate nodes between the source and the destination. The attacker A could have launched a RREP routing loop attack to form a loop between node1 and node2 by forwarding a RREP message with a higher RREP sequence number to node2 on behalf of node1. When node2 receives this falsified message, it updates its next hop from node D to node1 as well as the destination sequence number in its routing table. Therefore, a routing loop is formed and the packets are sent back and forth between node1 and node2.

Both attacks described in 4.1 and 4.2 are also called Denial-of-Service (DoS) attacks. DoS attacks do not intend to destroy the data message, but try to consume and compromise the scarce resource that available to nodes in the network. They can even disrupt the usability of the network.

DoS (Denial of Service) attacks

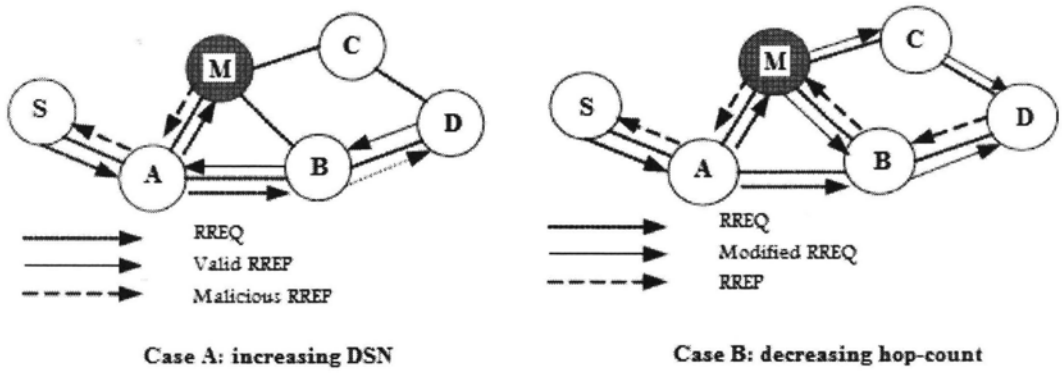
This kind of attack causes the entire network failed by depleting the energy of nodes in the network. For example, adversary may intercept a large number of signed messages and later send theses intercepted packets to a designated node in the network for a very short period, hence causing the node to be eventually exhausted due to constantly verifying signatures.

4.3 Route Re-direction Attack

The following figure [30] indicates two cases where a route re-direction attack could have been launched by malicious node M (circle with red color).

In the first case (Case A), malicious node M tries to initiate this attack by modifying the mutable fields in the routing message. These mutable fields may include hop count, sequence numbers and other metric related fields.

A malicious node M could divert the traffic through itself by advertising a route to the destination with a larger destination sequence number (DSN) than the one it received from the destination. For an instance, when M receives the RREQ from node A which was originated from source S for a route towards to destination D. Since AODV has an option that allows intermediate nodes to reply RREQ on behalf of destination if these nodes already have a fresh enough route to the destination, M will unicast a RREP back to node A with a larger DSN than the actual value last advertised by node D. Same thing happens to node D, when D receives the RREQ originated from node A, it also unicasts a RREP back to A. As a consequence, eventually A will get more than one RREP, which one should A choose? The answer is that A picks the RREP from malicious node M as a valid RREP since it simply has larger DSN. Therefore, A will re-direct all the subsequent traffics destined to D to malicious node M instead of node E.



Case A: increasing DSN **Case B: decreasing hop-count**

Fig. 4. 3 Route Re-direction Attack

In the second case (Case B), route re-direction attack may also be launched by modifying the metric field in the AODV routing message, which is the hop count field in this case. In AODV, hop count is a cumulative field contributed by each node in the path, it is also the only metric field used to decide the route between a source and a destination. A malicious node M simply modifies the hop count field to zero in order to claim that it has a better path to the destination. Therefore, when the modified RREQ reaches destination, malicious node M will be included on the reserve path that used to send RREP back to the source. And all traffics to the destination will be re-directed through the malicious node M.

Both Case A and Case B belong to the category of modification attacks in AODV, which is summarized below.

Modification attacks

1. Modification of Sequence Number (including both source and destination sequence No.)
   AODV uses sequence number in the routing message to distinguish the freshness of the route; a compromised node could increase this number to attract other nodes to select itself as one stop of the final route to the destination;
2. Modification of hop count number
   As hop count is the only metric in the original version of AODV to be used to decide the metric of a route, changing it on purpose would cause the routing protocol failed running properly. Decreasing the hop-count will lure the network traffic from other nodes to go through the malicious node; increasing the hop count could enhance the possibility of deselecting the node to be one of the stops on the final established route;

4.4 Formation of Routing Loops Attack

A malicious node may use other legitimate nodes' IP address and modify the value of the metric field to achieve the goal of creating routing loop. The figure [30] given below shows how this type of attack could be launched through Step a to Step c.

Assume that there exists a path between source S and destination X that goes through node B, C and E. There is also a known path from node A to C that goes through D. Suppose there is a malicious node M sitting in the vicinity where it can listen to the RREQs/RREPs that go through node A, B, C and D during the route discovery phase. Then M can create a routing loop among nodes A, B, C and D by launching both impersonation and modification of metric field in RREP.

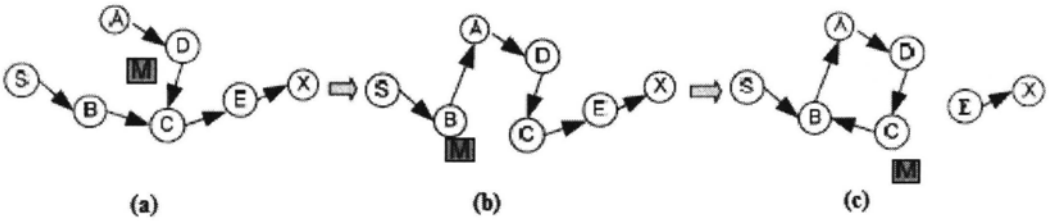


Fig. 4. 4 Formation of Routing Loops Attack

The following three steps explain how the routing loop is formed.

Step a)

Malicious node M impersonates node A's IP address, and then it moves closer to node B where node A cannot hear from M.

Step b)

M sends a falsified RREP to node B indicating a better metric (in terms of hop-count in AODV) than that of the value received from node C. As a result, now node B will re-direct all the traffics destined to X towards to node A

Step c)

Again similar actions have been taken by malicious node M. Now M gets closer to node C and sends a RREP to C on behalf of node B indicating a better metric than the one node C receives from node E. Therefore, now C will choose B as its next hop for the route to destination X, all traffics destined to X

will be routed to B. As a consequence, a routing loop now has been created and destination X is no longer reachable to node A, B, C and D.

Malicious nodes have to impersonate other legitimate nodes and modify the related routing metrics in order to create the routing loop attack. We have mentioned modification attacks in section 4.3, now giving the definition of impersonation attack as follows

Impersonation attacks

Attackers may use spoofed node's ID or IP address to impersonate other nodes in the network to gain traffics that were supposed to be designated to the authentic nodes or be involved in creating routing loops.

4.5 Fabrication Attack

Adversary may also fabricate the routing messages to disorder the routing decisions. For instance, a malicious node could simply fabricate a route error message in AODV protocol, this will put all the upstream nodes in the network into a very embarrassment situation since all of them now believe that a certain number of destination are unable to reach. This may result these upstream nodes to take further actions by re-initiating a route request to those unreachable destinations in order to discover and build another possible route to them. Once again, this brings the energy consuming issue on the table and significantly degrades the performance of the routing protocol such as network throughput and packet delivery latency.

4.6 Tunnelling Attack

In Ad hoc network, any node can be located adjacent to any other nodes. A tunnelling attack is referred to two or more malicious nodes in the network may collude and cooperate with each other to encapsulate and exchange routing messages between them by either using the existing data routes or potentially high power transceiver. The purpose is to prevent the honest intermediate nodes from correctly incrementing the metric field that will be used to measure the path length. In AODV, the only metric field used to make routing decision is the hop count field. The routing protocol always chooses the route with least hop-count. In another word, tunnelling attack may lead to the misrepresentation of the actual path length;

therefore, the attackers raise the chances of being included on the final established route between the source and destination and lured subsequent data traffics through them.

The following figure [7] illustrates such an attack where M1 and M2 are malicious nodes that are cooperating with each other in order to mispresent the actual path length by tunnelling the route request (RREQ) in AODV. The actual path is represented by the solid line in black color. The blue solid line denotes the tunnel and the dotted line in black color refers to the path that has been falsely claimed by M1 and M2.

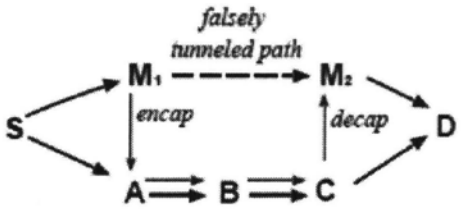


Fig. 4. 5 Tunnelling Attack

Upon receiving the RREQ from source S, M1 encapsulates the RREQ and tunnels it to M2 through the available existing data route {A→B→C}. Therefore, the RREQ will be tunnelled from M1 to M2 through the path {M1→ A→B→C→M2}. When M2 receives the tunnelled RREQ, it forwards it to destination node D as if it had only travelled {S→M1→M2→D}. Neither M1 nor M2 updates its routing table (such as hop count metric, previous hop, next hop) to reflect the fact that the RREQ also travelled through the path {A→B→C}. Once the route discovery process ends, it appears to destination D that there are two distinct routes available: {S→M1→M2→D} and {S→A→B→C→D}. Since AODV only picks the path with least hop counts, route {S→M1→M2→D} will be chosen to unicast the RREP back to source S through M2 and M1. When S receives the tunnelled RREP from M1, it falsely considers that the best available path to D would be via node M1 instead of going through node A in terms of path length. As a consequence, routing metrics, in this case hop count has been misrepresented.

Chapter 5

Background of Blom's Key Pre-distribution Scheme

Our propose idea is inspired by [31], which is a scheme builds on Blom's key pre-distribution scheme [32]. Base on Blom's scheme, [31] further improves the network resilience by introducing multiple-space key pre-distribution scheme, while Blom's scheme only offers single key space. In [31], two nodes have to share a common key space in order to deduce their pairwise secret keys. In the presence of two nodes share no common key space, a random key K is generated from one node and forwarded over the existing secure links to the other node, and two nodes would be able to start secure communication.

Since Blom's key pre-distribution scheme merely uses single key space, every two nodes can find and compute the shared secret pairwise keys between them. Blom's scheme also saves the space of memory storage since it only needs $\lambda+1$ memory spaces compared to the N-1 pairwise key pre-distribution scheme, where each node carries N-1 secret pairwise keys, each of which is only known to this node and one of the N-1 nodes (suppose N is the total number of nodes in the network). Here $\lambda+1$ is much smaller than N. However, Blom's scheme is not perfectly secure; it has to follow the λ-secure property. As long as the attacker compromises less than or equal to λ nodes in the network, the rest of the uncompromised nodes are perfectly secure; when there are more λ nodes are compromised, all pairwise keys of the entire network are compromised.

The threshold λ can be treated as a critical security parameter in determining how secure the network wants to be since the adversary has to attack a significant portion of the entire network in order to gain high payoff. As increasing the network security simply means to enlarge λ, this inevitable leads to higher memory usage.

The following briefly explains how Blom's λ secure key pre-distribution scheme works.

During the pre-deployment phase, first let the computer construct a $(\lambda+1) \times N$ matrix G over a finite field GF(q), where N is the network size and G is public information and can be known by any node in the network, even the adversary.

Secondly, a random $(\lambda+1)\times(\lambda+1)$ symmetric matrix D over GF(q) needs to be produced. Finally the last matrix $A = (D \cdot G)^T$ is created, where $(D \cdot G)^T$ is the transpose of $D \cdot G$ and A is a $N \times (\lambda+1)$ matrix. Matrix D cannot be disclosed to any node in the network and must be kept secret. Since matrix D is symmetric, it can be proved that $A \cdot G = (D \cdot G)^T \cdot G = G^T \cdot D^T \cdot G = G^T \cdot D \cdot G = (A \cdot G)^T$. This indicates that $A \cdot G$ is a symmetric matrix; therefore, we have the following conclusions.

The following key pre-distribution scheme can be used to compute the pairwise keys, for k = 1, . . . ,N:

1. Store the k^{th} row of matrix A at node k, and
2. Store the k^{th} column of matrix G at node k.

When nodes i and j need to find the pairwise key between them, they first exchange their columns of G, and then they can compute K_{ij} and K_{ji} respectively, using their private rows of A because $K_{ij} = K_{ji}$. Since G is public information, its columns can be transmitted in plaintext.

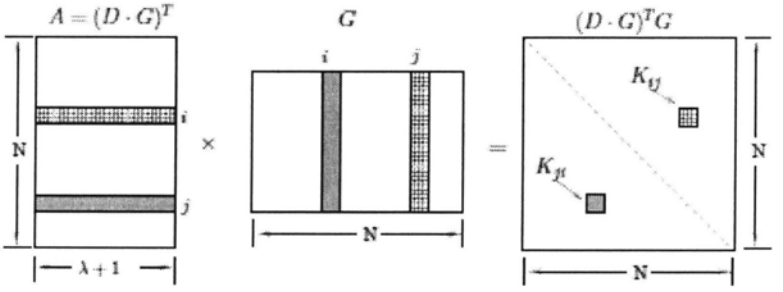


Fig. 5. 1 Generating Keys in Blom's Scheme

The above figure [31] illustrates how the pairwise key Kij = Kji is generated.

Chapter 6

Proposed SEAODV

This chapter presents our proposed security enhanced AODV routing protocol in details. SEAODV employs Blom's key pre-distribution scheme and enhanced HELLO message to compute the pairwise transient key (PTK), which subsequently being used to distribute the group transient key (GTK). PTK and GTK are used to secure the unicast and broadcast routing message respectively.

The remainder of this chapter is structured as follows.

Section 6.1 Identifies the mutable and non-mutable fields in AODV routing messages.

Section 6.2 Explains the use of two types of keys: PTK and GTK in SEAODV.

Section 6.3 Presents formats of the enhanced HELLO messages: HELLO RREQ and HELLO RREP.

Section 6.4 Explains how public information and GTK are exchanged among nodes by using the enhanced HELLO messages in conjunction with Blom's scheme.

Section 6.5-6.7 Describes how to secure route discovery, route setup and route maintenance.

6.1 Identification of Mutable and Non-mutable fields

In AODV routing messages, some information elements that can be modified by the intermediate nodes are considered as mutable fields. Those cannot be modified arbitrarily are termed as non-mutable field.

The following figures [2] show different standard routing message formats used in AODV. We consider the Hop Count field to be the mutable information field in both RREQ and RREP messages since this field provides information on the total number of links in the path and will be incremented by each intermediate node. The hop count field is also the only metric field in AODV routing message for making routing decision. As a consequence, protection of the hop count field in both RREQ and RREP is extremely important and necessary.

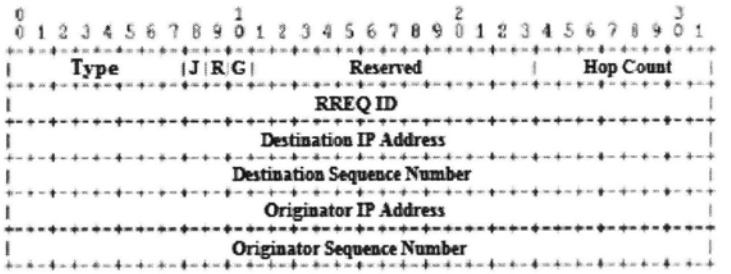


Fig. 6. 1 RREQ Message Format

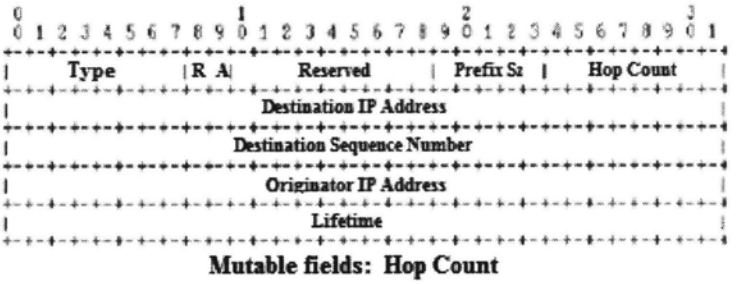


Fig. 6. 2 RREP Message Format

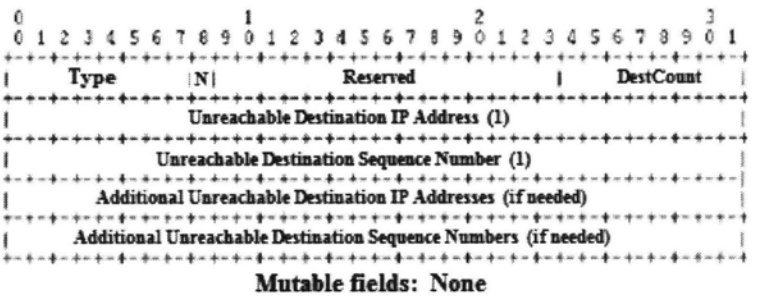


Fig. 6. 3 RERR Message Format

In RERR message, none of the information fields are considered as mutable fields. However, sometimes securing the non-mutable fields is also necessary since the adversary can simply change these fields in order to disrupt the operation of the routing protocol. For example, arbitrarily modifying the IP address of unreachable destination or adding excessive number of unreachable destination leads to additional re-initiation process of route discovery. As a result, data flows are suspended and network is going to be congested by the flooding of excessive route request message because every single node in the network is seeking a new route to the so called "unreachable destinations".

6.2 Use of Keys

Our proposed SEAODV builds on existing AODV. Choosing AODV as our protocol's footstone is due to its simplicity, maturity, popularity and availability in the research over the past few years.

SEAODV requires each node in the network maintains two key hierarchies. One is the broadcast key hierarchy, which includes all the broadcast keys from its active one hop neighbours. The other hierarchy is called unicast hierarchy and it stores all the secret pairwise keys that this node shares with its one hop neighbours. Every node uses keys in its broadcast hierarchy to authenticate the incoming broadcast routing messages (e.g., RREQ) from its one hop neighbours and applies secret pairwise keys in the unicast hierarchy for verifying the incoming unicast messages such as RREP.

From now we define the broadcast key and the pairwise key in the two key hierarchies to be called Group Transient Key (GTK) and Pairwise Transient Key (PTK) respectively. Therefore, GTK is used for encrypting/decrypting broadcast message and PTK is used for encrypting/decrypting unicast message.

Our scheme implements a hop-by-hop authentication routing since every node in the network has its own GTK and PTK. All routing messages can be thoroughly protected at every single hop during the entire route setup process, hence delivering complete end-to-end secure routing solution.

6.3 Enhanced HELLO Message (HELLO RREQ, HELLO RREP)

6.3.1 Notation

We use the variables and notations given in the following table to describe the enhanced HELLO messages, our proposed secure routing protocol and relevant cryptography operations.

EN_{HRREQ}	Enhanced hello RREQ message	OSN_A	Originator sequence number of node A
EN_{HRREP}	Enhanced hello RREP message	DSN_B	Destination sequence number of node B
GTK_A	Group Transient Key of Node A	IP_A	IP address of node A
GTK_B	Group Transient Key of node B	IP_B	IP address of node B
PTK_A	Pairwise Transient Key of Node A	IP*	Broadcast IP address

PTK_B	Pairwise Transient Key of Node B	NC_A	Nonce issued by node A
$\{d\}GTK_A$	Encryption of data d with key GTK_A	NC_B	Nonce issued by node B
$\{d\}GTK_B$	Encryption of data d with key GTK_B	$Seed_{G\_A}$	Seed of column of Public Matrix G, Node A
$\{d\}PTK_A$	Encryption of data d with key PTK_A	$P_{Row\_A\_A[\,]}$	Row of Private Matrix A, Node A
$\{d\}PTK_B$	Encryption of data d with key PTK_B	M_{type}	Message type
MAC(K,M)	Computation of MAC over message M with key K	M_{ID}	Message ID
t	Timestamp	M	All the elements before the MAC field in SEAODV routing message

Tab. 6. 1 Variables and Notations

6.3.2 HELLO RREQ

In AODV, HELLO message [33] is broadcast only to its one-hop neighbours in order to maintain updated local connections. In our proposed SEAODV, we define another two types of Enhanced HELLO messages by using the idea inspired from [34]. Each node embeds its column of the public G matrix into its enhanced HELLO RREQ message. Since each column of the public known matrix G can be regenerated by applying the seed from each node, where the seed is a primitive element of GF(q), every node only needs to store the seed in order to exchange the public information of matrix G at a later stage.

To guarantee bi-directional links, enhanced HELLO RREQ message format is used and neighbouring nodes receive it will reply with an enhanced HELLO RREP to acknowledge this link. The new enhanced HELLO RREQ is given below, from which modifications are made based on standard format of AODV RREQ message.

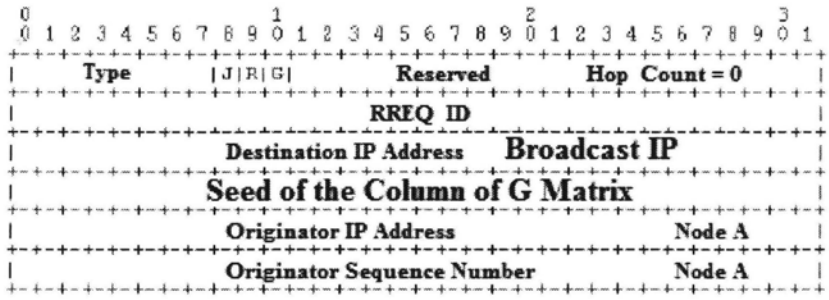


Fig. 6. 4 Enhance HELLO RREQ Message Format

Assume node A is the originator of the enhanced HELLO RREQ; this RREQ can also be represented in the following format by using notations given in Tab. 6.1

$$EN_{HRREQ}: [M_{type}, M_{ID}, IP *, Seed_G, IP_A, OSN_A] \quad (1)$$

Compare the enhanced HELLO RREQ message in Fig. 6.4 against the original RREQ, the original 32-bit destination sequence number field is now replaced with the seed of the column of public G Matrix, which is obtained from the pre-generated G matrix. This is because the destination address is now the local broadcast IP address (e.g., 255.255.255.255); the destination sequence number does not perform any meaningful functions in the enhanced HELLO RREQ message. The seed must be 32-bits aligned. Message length and other fields in the enhanced HELLO RREQ remain unchanged.

There are three causes that lead to node A to broadcast the enhanced HELLO RREQ to its one-hop neighbours periodically.

1. To announce its one-hop neighbours that it is in active mode;
2. To let its one-hop neighbours know its seed of the column of public Matrix G
3. To trigger its one-hop neighbours to send their encrypted GTK back to itself

6.3.3 HELLO RREP

The following figure presents the message format of the enhanced HELLO RREP. The hop count field now is replaced with zero and the lifetime field is equal to the value of ALLOWED_HELLO_LOSS × HELLO_INTERVAL.

In AODV, HELLO_INTERVAL is defined as the maximum time interval between the transmissions of Hello messages. ALLOWED_HELLO_LOSS is considered as the maximum number of allowed hello message loss before a neighbour link is recognized to be broken. After receiving an enhanced HELLO

RREP message from a neighbour, if no packet is received by a node from that neighbour for more than the ALLOWED_HELLO LOSS × HELLO_INTERVAL time milliseconds, the node should assume that the link is currently broken. By default, ALLOWED_HELLO_LOSS is equal to 2 and HELLO_INTERVAL is equal to 1000 ms [2].

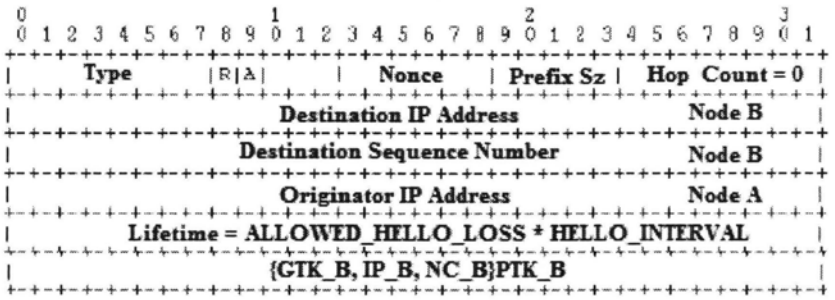


Fig. 6. 5 Enhance HELLO RREP Message Format

Assume that node B is one of the one-hop neighbours of node A and receives the HELLO RREQ from node A. In order to respond to node A, node B uincasts an enhanced HELLO RREP back to node A.

The following presents three reasons of why node B needs to send HELLO RREP back to node A.

1. To acknowledge node A that it already receives the public key of node A;
2. Once obtained node A's public key, node B can derive the secret pairwise key it shares with node A and use this shared key to encrypt its own group key and unicasts it back to node A;
3. Upon receiving the HELLO RREP from node B, now node A confirms that there is a bi-directional wireless link between itself and node B. Node A also knows the GTK of node B by using its PTK to decrypt the received HELLO RREP.

The enhanced HELLO RREP message can also be expressed in the following way.

$$EN_{HRREP}: [M_{type}, NC_B, IP_B, DSN_B, IP_A, Lifetime, \{GTK_B, IP_B, NC_B\}PTK_B] \quad (2)$$

Detailed explanation of how public Seed_G and GTK keys are exchanged among nodes in the network can be found in the following sections.

6.4 Exchange Public Seed_G and GTK by Using Enhanced HELLO Message

During the key pre-distribution phase, every legitimate node in the wireless mesh network knows and stores its public known Seed_G (seed of the column of public G matrix) and the corresponding private

row of the generated A matrix. Now the question is how the public Seed_G and GTK key in each node can be exchanged among nodes in the WMN by using the enhanced HELLO RREQ and HELLO RREP message?

By considering the following figure, the entire exchange process can be depicted in the following three major steps.

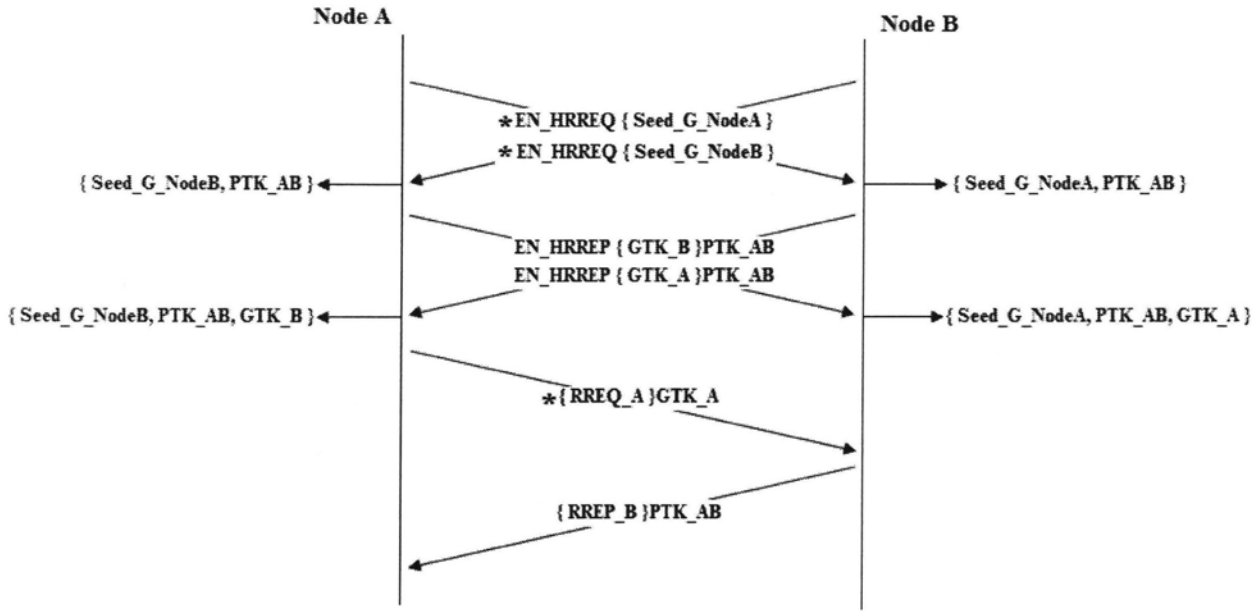


Fig. 6. 6 Public Seed and GTK Key Exchange Process

Step 1: Exchange of Seed_G of public G matrix

From the above Fig. 6.6, when node A wants to exchange its Seed_G with all its one-hop neighbours, node A first looks for its own public Seed_G from its key pool, then it broadcasts the enhanced HELLO RREQ to all its one-hop neighbours while node A itself is receiving the Seed_G from all its one-hop neighbours simultaneously. Assume node B is one of node A's one-hop neighbours and is expecting to experience the same situation as node A. In other words, node B also broadcasts its own Seed_G and looks forward to receiving other Seed_Gs from all its one-hop neighbours. Eventually, every node in the network gains a group of Seed_G that it obtains from its one-hop neighbours.

Node A and node B broadcast their HELLO RREQ by using the message format given below.

$$A \rightarrow * : EN_{HRREQ} : [M_{type}, M_{ID}, IP *, Seed_{G_A}, IP_A, OSN_A] \quad (3)$$

$$B \rightarrow * : EN_{HRREQ} : [M_{type}, M_{ID}, IP *, Seed_{G_B}, IP_B, OSN_B] \quad (4)$$

As a consequence, after a certain amount of time, node A gets node B's Seed_G_B and so does for node B.

Under the situation of re-assigning different public and private key pairs from matrix G, node A and node B can re-announce different Seed_G, this gives extra security of the routing protocol since every node in the network can now update its Seed_G anytime whenever the node senses the uncertainty of the key pair.

Upon finishing step 1, every node in the network possesses the public Seed_G of all its one-hop neighbours, so does its one-hop neighbours.

Step 2: Derivation of PTK (Pairwise Transient Key)

On receiving the public Seed_G of its one-hop neighbours, the node now can use both Seed_G it received from its one-hop neighbour and its corresponding private row of matrix A to compute the unique PTK that it shares with every one-hop neighbour by using Blom's scheme as discussed earlier in chapter 6.

Initially node A has A(i) and seed for G(i), and node B has A(j) and seed for G(j). After exchanging the seeds, node A can regenerate G(j) and node B can regenerate G(i). By using Fig. 6.1 as reference, the pairwise secret key between nodes A and B, Kij =Kji, can be computed by these two nodes independently through using the following equation.

$$K_{ij} = K_{ji} = A_i \times G_j = A_j \times G_i \quad (5)$$

Recalls the variables and notations given in Tab. 6.1, the figure below describes how PTK can be computed by applying Blom's key pre-distrbution scheme.

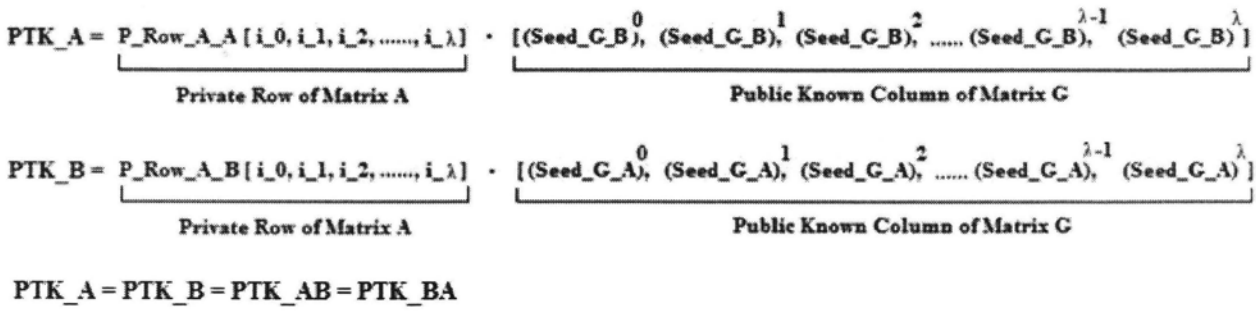


Fig. 6. 7 Computation of PTK by Using Blom's Scheme

From Fig. 6.7, we can conclude that both the private row of matrix A and the public known column of matrix G are comprised of λ+1 different elements. Each node has its own key pair: the private row of matrix A and public known column of matrix G respectively. The key pair is generated and stored in the phase of key pre-distribution.

Upon finishing step 2, every node has stored the public known Seed_G of all its one-hop neighbours and derived every single unique PTK secret pairwise key that it shares with its one-hop neighbours. Now every node can apply its PTK key to encrypt its GTK key and unicast it back to the originator of the HELLO RREQ message by using the HELLO RREP.

Step 3: Exchange of GTK (Group Transient Key) through HELLO RREP

The unicast HELLO RREP message from node A and B can be expressed in the following format

$$A \rightarrow : EN_{HRREP}: [M_{type}, NC_A, IP_A, DSN_A, IP_B, Lifetime, \{GTK_A, IP_A, NC_A\}PTK_A] \quad (6)$$

$$B \rightarrow : EN_{HRREP}: [M_{type}, NC_B, IP_B, DSN_B, IP_A, Lifetime, \{GTK_B, IP_B, NC_B\}PTK_B] \quad (7)$$

In Fig. 6.6, upon receiving HELLO RREQ from node A, node B starts unicasting a HELLO RREP back to node A. The encrypted GTK_B of node B is also attached within the unicasted HELLO RREP message. Once node A receives HELLO RREP from node B, node A applies its private PTK_A it computes from step 2 to decrypt the GTK_B and stores it in its database. Similarly, the exactly same process applies on node B, which uses its PTK_B to decrypt the GTK_A and stores it. Since PTK_A is identical to PTK_B, node A and B can use their own PTK key that it shares with the originator of the GTK to decrypt the appropriate GTK key.

Whenever step 3 is finished, every node in the network now also has all the GTK keys from all its one-hop neighbours. Therefore, now each node possesses all the public known Seed_G and GTK key of all its

one-hop neighbours as well as all its self-derived PTK keys, and so does the node's one-hop neighbours. This implies the completion of the key exchange process and indicates that from now on every node in wireless network can start initiating secure routing.

In the next context, we use pseudo code and flow chart to take a closer look in the key exchange process between node A and B. It consists of two parts, the first part explains the HELLO RREQ message and the second part discusses the HELLO RREP message.

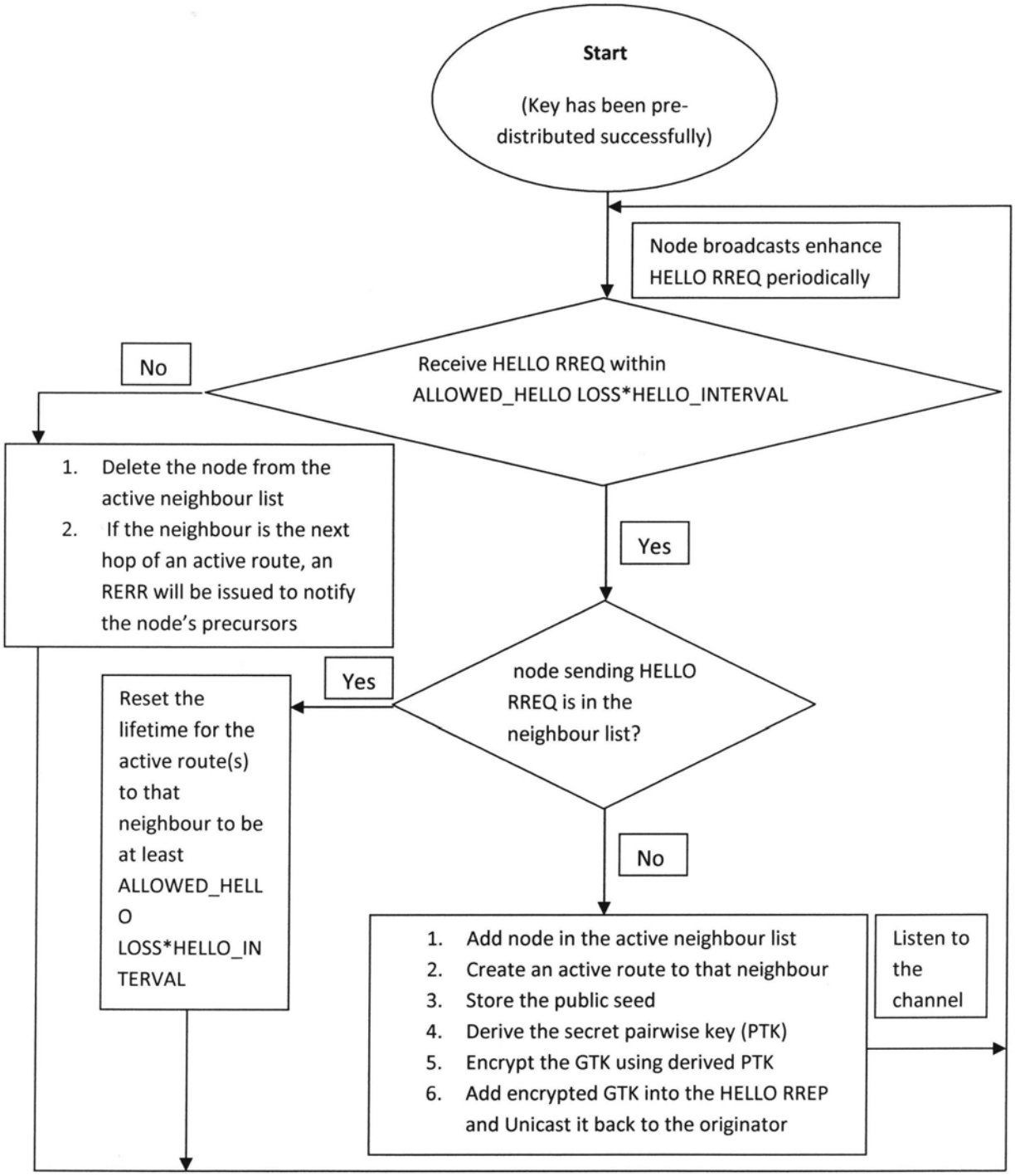


Fig. 6. 8 Key Exchange Process_RREQ_A (Node B's reaction, Flow Chart)

The above Fig. 6.8 gives node B's reaction when it receives the broadcast HELLO RREQ from node A based on the assumption that node A is the originator of the HELLO RREQ and node B is one of node A's one-hop neighbours.

The Key Exchange Process_RREQ_A (Node B's reaction) can also be interpreted by the pseudocode given in Fig. 6.9.

```
Begin {
   If ((NodeB.Receive_Time – NodeB.Last_Receive_Time) <= Allowed_Hello Loss * Hello Interval)
   {    If (NodeA == NodeB.OneHop_NeighbourList)
        NodeB.ResetTimeToNodeA = CurrentTime + Allowed_Hello Loss * Hello Interval
        Else  { NodeB.AddNodeToNeighbourList(NodeA);
                NodeB.AddActiveRouteTo(NodeA);
                NodeB.StoreSeed(Seed_G_A);
                PTK_B = NodeB.CreatPTK(P_Row_A_B, Seed_G_A);
                GTK_B_Encrypted = NodeB.GTK_Encrypt(PTK_B);
                NodeB.AddGTK_To_Hello_RREP(GTK_B_Encrypted);
                NodeB.SendTo(HELLO_RREP, NodeA);
              }
        Endif
     }
     Else  { NodeB.Remove_Onehop_neighbour(NodeA);
             If (NodeB.Routingtable_Activeroute(Nexthop, NodeA) == True)
               NodeB.InitiateRouteError(Nexthop, NodeA.Precursors);
           }
    Endif
} End
```

Fig. 6. 9 Key Exchange Process_RREQ_A (Node B's reaction, Pseudocode)

Once node A receives the unicast message from node B, node A follows the reactions presented in the flow diagram as given below in Fig. 6.10.

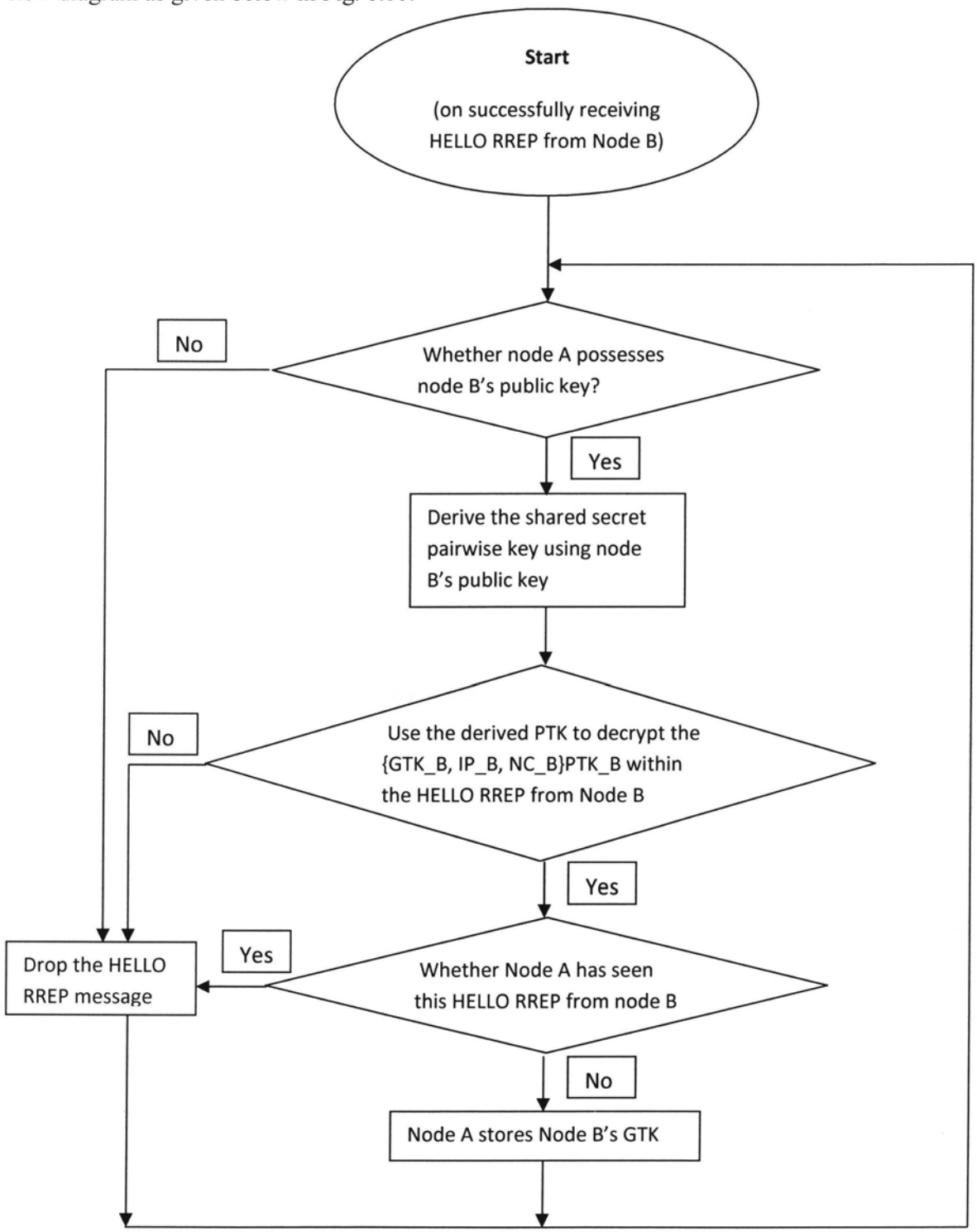


Fig. 6. 10 Key Exchange Process_RREP_B (Node A's reaction, Flow Chart)

Similarly, the pseudocode for Key Exchange Process_HELLO RREP_B (Node A's reaction) is presented hereinafter.

```
Begin {

    If (NodeA.Keypool_Seed_G_B == True)

    {     PTK_A = NodeA.Derive_PTK_A(Seed_G_B);

          If (NodeA.Decrypt(PTK_A, {GTK_B, IP_B, NC_B}PTK_B) == True)

          {       If  (NC_B <= NC_B_PreviouslyStored)

                              NodeA.Drop(HELLO_RREP)

                  Else    {    NodeA.StoreGTK(GTK_B, Node B) }

                  Endif

          }

          Else NodeA.Drop(HELLO_RREP);

    }   Else  NodeA.Drop(HELLO_RREP);

} End
```

Fig. 6. 11 Key Exchange Process_RREP_B (Node A's reaction, Pseudocode)

Upon accomplishing the key exchange process, every node stores its one-hop neighbours' public seeds (Seed_G), the group of secret pairwise PTK keys that it share with its every single one-hop neighbour and a group of GTK keys from its one-hop neighbours. To ensure the security of route discovery, route setup and route maintenance, node in the network applies its own GTK key to secure the broadcast routing message and uses PTK key to secure the unicast routing message. All of these will be discussed in the following sections 6.5, 6.6 and 6.7 respectively.

6.5 Securing Route Discovery

In order to implement a hop-by-hop authentication, each node must verify the incoming message from its one-hop neighbours before re-broadcasting or unicasting the message. The trust relationship between each pair of nodes relies on their shared GTK and PTK keys, which have been obtained during the key exchange process as mentioned earlier.

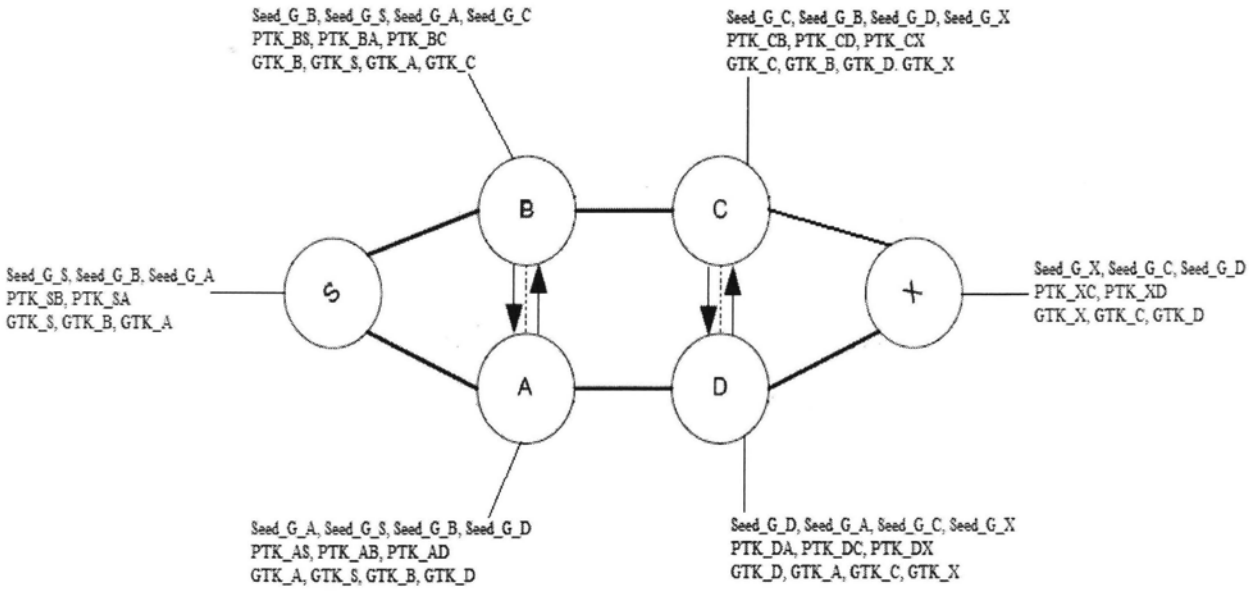


Fig. 6. 12 Distribution of Keys {Seed_G, PTK and GTK}

The above figure shows the key distribution in a six-node wireless mesh network, with node S and X are the source and destination node respectively, others are intermediate nodes between node S and X. In the figure, every node has its own group of Seed_G, GTK and PTK. Seed_G is publicly known, while GTK and PTK are privately shared between the node itself and its one-hop neighbours. Therefore, a compromised node in the network does not affect communications among other nodes as long as the total number of compromised node is less or equal to λ, which mean is has to follow the λ-secure property of Blom's key pre-distribution scheme: as long as an attacker compromises less than or equal to λ nodes, the remaining uncompromised nodes in the network are perfectly secure; when the attacker compromises more than λ nodes, all secret shared pairwise keys (PTK in our case) of every node in the entire network are compromised.

Since PTK keys in our proposed SEAODV routing are used to secure delivery of GTK keys and the subsequent unicast routing messages, and GTK keys are used to secure broadcast routing messages, the compromise of PTK keys means the defeat of GTK keys and the security of the routing protocol. However, how to improve the security of the Blom's key pre-distribution scheme is out of the scope in this thesis. The thesis is focused on the security improvement of AODV routing, which adopts Blom's scheme to implement key pre-distribution.

Route discovery process is similar to that of standard AODV, but a MAC extension is appended to the end of the AODV routing message. The new format of the RREQ in SEAODV is given below.

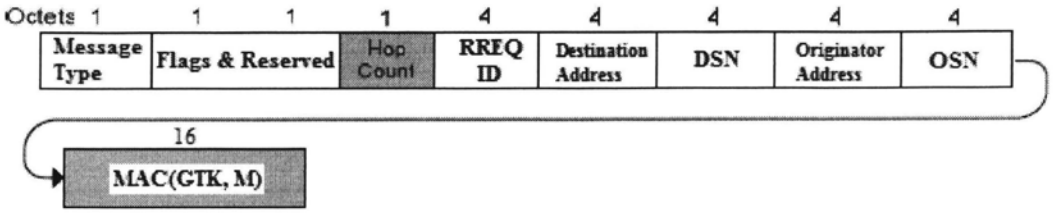


Fig. 6. 13 Format of Modified RREQ

The first shadowed box is the mutable field, which has been identified in section 6.1. The second shadowed box is the MAC field that has been appended at the end of the RREQ routing message.

The MAC is computed over message M by using the key GTK of node who needs to broadcast a RREQ to its one-hop neighbours. Message M refers to all the elements before the MAC field in this RREQ. Therefore, M includes both the non-mutable fields and the mutable field, which is the hop-count in this case.

Securing the non-mutable fields in the RREQ is as much important as securing mutable fields. Sometimes adversary can intentionally modify the content of the non-mutable fields to gain additional pay off or to disorder the routing protocol. For an instance, DSN (Destination Sequence Number) in RREQ is considered as non-mutable field; however, attacker may give a higher (fresher) number to this field in order to submerge the originally useful routing messages since node only processes "fresh enough" RREQ and discards the stale ones. For example, considering a node A that wants to broadcast a modified RREQ, it sends the following message

$$A \rightarrow * : RREQ : [M, MAC(GTK,\ M)] \tag{8}$$

Whenever a node needs to discover a route to a designated destination, it broadcasts the modified RREQ message to all its neighbours. Upon receiving the broadcast RREQ, the neighbour checks whether it possesses the GTK key of the sender by checking its group of GTK. If there is a match found, the receiving node computes the corresponding MAC by using the entire received message and the found GTK. The receiving node then compares the new computed MAC with the one it just received from the sender. If there is a perfect match, the received RREQ from the sender is considered to be authentic and unaltered. Only will that happen, the receiving node updates the mutable field (hop-count in RREQ) and its routing table, and subsequently sets up the reverse path back to the source by recording the neighbour from which it received the RREQ. Finally, the receiving node examines whether or not it is the destination. In the case that the node itself is destination, the node responds a RREP with a new

MAC(PTK, M) affixed to the end of the RREP and unicasts this RREP back to its next hop of its reverse path towards to the source. The appended MAC(PTK, M) is computed on the entire RREP message by using the PTK key the node secretly shares with its next hop, to which the RREP is going to be forwarded. On the other hand, in the case that the node is an intermediate node between the source and destination, the node applies its own GTK key on the updated RREQ to compute the new MAC(GTK, M) and attaches it to the end of the RREQ before it re-broadcasts the new RREQ to its one-hop neighbours.

If the receiving node of the broadcast RREQ does not have the GTK key of the sender of RREQ, the receiving node simply discards the RREQ since it does not have the appropriate GTK to compute the MAC to be compared against the one it received. The receiving node also discards the received RREQ under the situation that there is a mismatch between the new computed MAC'(GTK, M) and the MAC it receives from the sender of the RREQ. In both cases mentioned hereinbefore, the RREQ message is considered as being non-authentic and already tampered, hence it will be discarded.

Assume node A broadcasts a RREQ, node B is one of its one-hop neighbours that receives the RREQ. Hereinafter (Fig. 6. 14) presents the pseudocode for the route discovery process (Node B's reaction).

```
Begin {
    If (NodeB.GTK_List includes NodeA.GTK)
    {    Compute MAC'(GTK, M);
         If (MAC'(GTK, M) == MAC(GTK, M))
         {       Update Hop-count in RREQ;
                 Setup reverse path back to the source;
                 Update routing table;
                 If  (NodeB is Destination node)
                    {      Look for PTK that NodeB shares with NodeA in NodeB.PTK_List;
                           Compute MAC(PTK, M);
                           Attach MAC(PTK, M) to the end of the RREP;
                           Unicast RREP back to NodeA towards to the source;
                     } Else   {  GTK_NodeB = NodeB.GetGTK;
                                 Compute MAC(GTK_NodeB, M);
                                 Attach MAC(GTK_NodeB, M) to the end of the RREQ;
                                 Re-broadcast RREQ to NodeB's one-hop neighbours;
                              }
                  Endif
         } Else  Drop RREQ
         Endif
    } Else Drop RREQ
    Endif
} End
```

Fig. 6. 14 Route Discovery Process_RREQ_A (Node B's Reaction, Pseudocode)

The following figure depicts the entire route discovery process based on the pseudocode presented in Fig. 6.14.

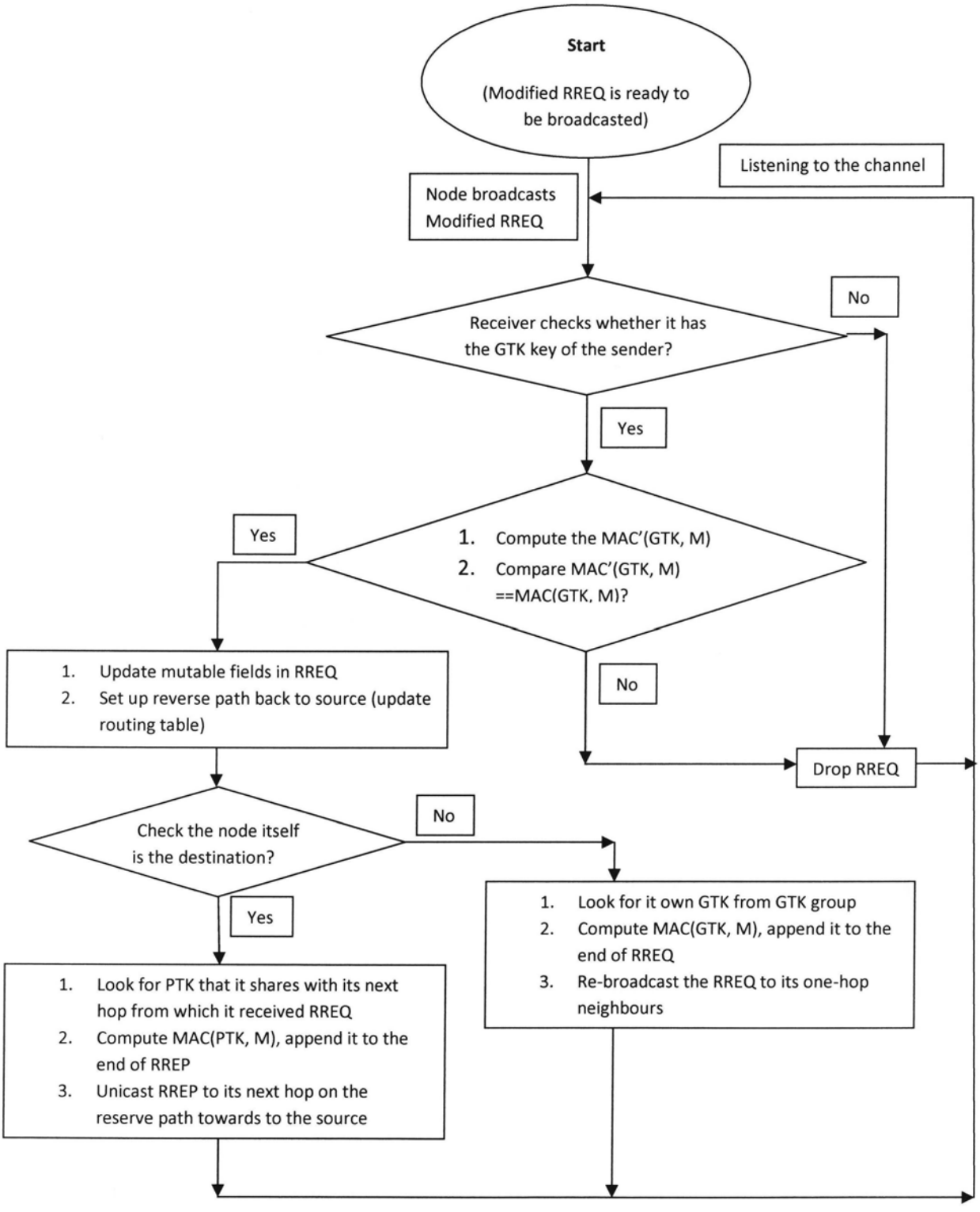


Fig. 6. 15 Route Discovery Process_RREQ_A (Node B's reaction, Flow Chart)

6.6 Securing Route Setup

Eventually, the RREQ message will reach the destination or an intermediate node that already has a fresh enough route to the destination. Therefore, a destination itself or an intermediate node can generate a modified RREP and unicast it back to the next hop from which it received the RREQ towards to the originator of the RREQ (in this case, the source). Since RREP routing message is authenticated at each hop due to the use of PTK keys, adversary has no opportunity to re-direct the traffic. Detailed discussions are presented in chapter 7.

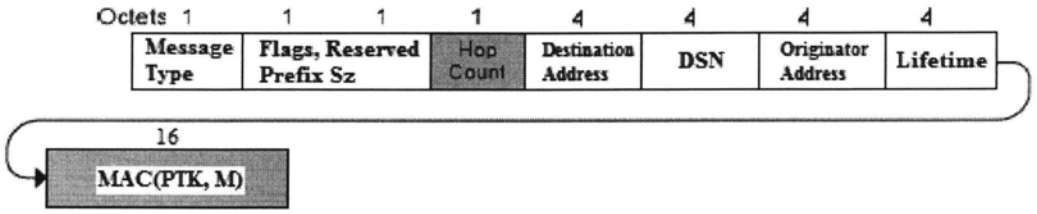


Fig. 6. 16 Format of Modified RREP

Similar to secure the RREQ message, there is only one more field is attached to the end of the stand AODV RREP, which is a MAC(PTK, M). This MAC is computed on the entire RREP message by using the PTK key that the node secretly shares with the one to which the RREP is going to be forwarded.

The format of a modified RREP is shown in Fig. 6.16. The mutable field in the RREP is identified as hop-count, which is indicated in the first shadowed box. The MAC field is given in the second shadowed box and appended at the end of the RREP.

Before unicasting the modified RREP back to the originator of the RREQ, the node first needs to check its routing table to identify the next hop from which it received the broadcast RREQ; then the node applies the PTK key it privately shares with the identified next hop to compute the MAC(PTK, M) and affixes this MAC to the end of RREP before unicasting the modified RREP back to the identified next hop towards to the source.

For instance, suppose a node B who needs to unicast a modified RREP back to Node A, it sends the following message.

$$B \rightarrow * : RREP : [M, MAC(PTK_{BA}, M)] \quad (9)$$

As mentioned earlier, $PTK_{AB} = PTK_{BA}$

Upon receiving RREP from node B, node A checks whether PTK_BA is in its group of PTK. When there is a match, node A computes MAC'(PTK_AB, M) and compares MAC'(PTK_AB, M) against the

MAC(PTK_BA, M) it received from node B. M represents the entire message of RREP before the MAC field as indicated in the format of modified RREP.

When considering MAC'(PTK_AB, M) matches MAC(PTK_BA, M), the received RREP from node B is considered to be authentic and integrated. In this case, node A updates the hop-count field in the RREP and its own routing table, sets up the forwarding path towards to the destination. Apart from this, node A also searches the appropriate PTK key that it shares with its next hop to which the new RREP is going to be forwarded towards to the source. Node A then uses this found PTK key to construct the new MAC(PTK, M) to be attached at the end of the new RREP. Intuitive illustration can be found in the flow diagram presented in Fig. 6.17, while corresponding pseudocode is given in Fig. 6.18.

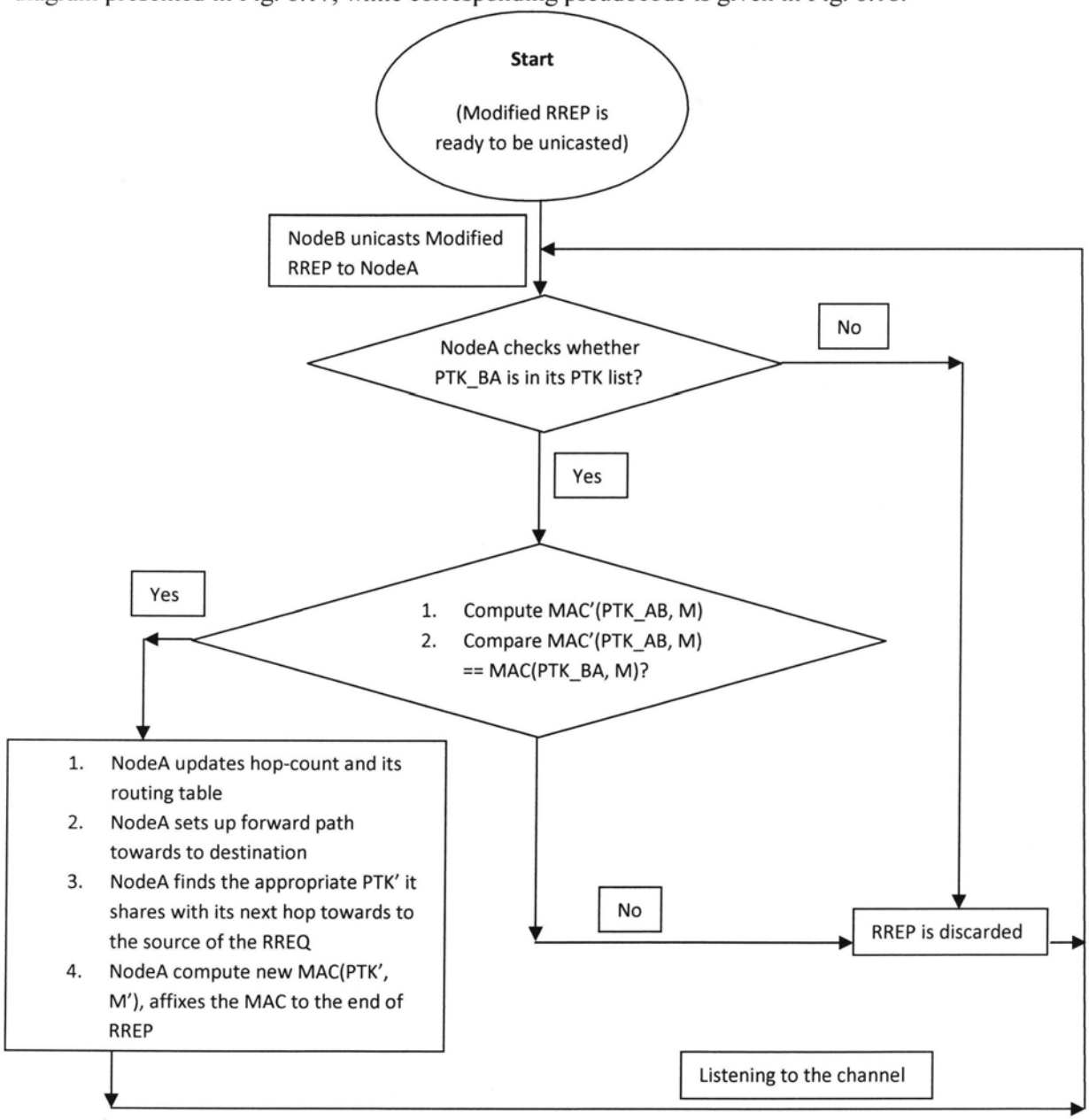


Fig. 6. 17 Route Setup Process_RREP_B (Node A's reaction, Flow Chart)

```
Begin {
    If (NodeA.PTK_List includes NodeB.PTK)
    {    Compute MAC'(PTK_AB, M);
         If (MAC'(PTK_AB, M) == MAC(PTK_BA, M))
         {      NodeA updates hop-count field in RREP;
                NodeA sets up forwarding path towards to destination;
                NodeA updates its routing table;
                NodeA searches appropriate PTK it shares with its next-hop to which the new RREP
                Is going to be forwarded;
                NodeA uses the found PTK' to compute new MAC(PTK', M) and affixes the MAC to
                the end of the new RREP;
                NodeA unicasts the new RREP to its next-hop;   }
          Else  NodeA discards received RREP
          Endif
    } Else  NodeA discards received RREP
    Endif
} End
```

Fig. 6. 18 Route Setup Process_RREP_B (Node A's reaction, Pseudocode)

6.7 Securing Route Maintenance

AODV is an on-demand routing protocol. That means when no traffic has occurred on an existing route for the entire lifetime of that particular route, the route will be de-activated from its routing table.

A node generates a RERR message under the following three situations:

1. If a node receives data packet destined to another node for which it does not have an active route in its routing table, the node will generate a route error (RERR) message.
2. Whenever there is a broken link for the next hop of an active route has been detected by a node, the node will initiate a RERR message to all its precursors that may use the broken next hop towards to their destinations.
3. On receiving a RERR from a neighbour for one or more active routes, node also generates a RERR.

The following figure shows the format of a modified RERR message.

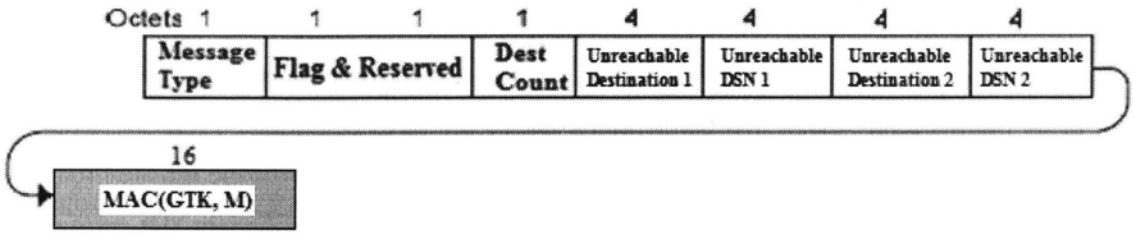


Fig. 6. 19 Format of Modified RERR

From the above figure, once again a MAC(GTK, M) has been appended to the end of the RERR. M is the entire message before this MAC field and has the same format as the standard AODV RERR. The field of "DestCount" represents the number of unreachable destinations. This number must be at least equal or greater than one. The modified RERR is broadcast to all its one-hop neighbours in order to deliver the notification of the unreachable destinations.

Let's suppose a node B that needs to broadcast a modified RERR to all its one-hop neighbours. Node A is one of node B's one-hop neighbours. Node B broadcasts the following message.

$$B \rightarrow * : RERR : [M, MAC(GTK_B,\ M)] \tag{10}$$

GTK_B is the GTK key that node B secretly shares with all its one-hop neighbours.

The MAC field in modified RERR is actually the extension part and it is created by using node B's GTK_B on the entire RERR message before this field.

Upon receiving the broadcast RERR message from node B, node A first checks whether it has GTK_B stored in its group of GTK. If GTK_B is found, then node A computes the MAC'(GTK_B, M') by using the found GTK_B in its database and the received RERR. The MAC'(GTK_B, M') is going to be

compared against the received MAC(GTK_B, M), which has been attached to the end of the received RERR. If the comparison results between these two MACs are positive, node A searches its routing table and identifies the affected routes (in this case, a new group of unreachable destinations) that use node B as its next-hop based on the unreachable destination list received from node B. If there is no affected routes found in node A's routing table, node A simply drops the RERR and starts listening to the channel again.

Node A also discards the RERR under another two situations. First, node A fails to find the GTK_B in its group of GTK key; second, the MAC'(GTK_B, M') computed by node A is not consistent with the one received from RERR, which is MAC(GTK_B, M).

Now moving to another scenario, node A has identified that at least one or more routes are going to be affected in its routing table, given that these routes satisfy the three conditions. First, the route must be active; second, the route uses node B as its next-hop; third, the destinations of the affected route in node A's routing table are members of the unreachable destinations list specified in the received RERR message from node B.

Node A takes the following actions once it successfully detects one or more routes in its routing table are going to be affected.

1. Node A marks the affected routes as invalid in its routing table;
2. Node A updates the RERR message. For example, the number of affected destinations in Node A's routing table might not be the same as that of the received one from node B. The number of affected destination has to be smaller or identical to that of in RERR obtained from node B;
3. Node A computes the new MAC(GTK_A, M) by using its own GTK_A and the updated RERR obtained from step 2 and attaches it to the end of the updated RERR;
4. Node A broadcasts the new RERR to all its one-hop neighbours and starts listening to the channel.

Assume node B broadcasts a RERR, node A is one of node B's one-hop neighbours, the following flow chart given in Fig. 6.20 shows the processing procedure of the modified RERR upon receiving the RERR from node B by Node A. The corresponding pseudocode is given right after the flow chart.

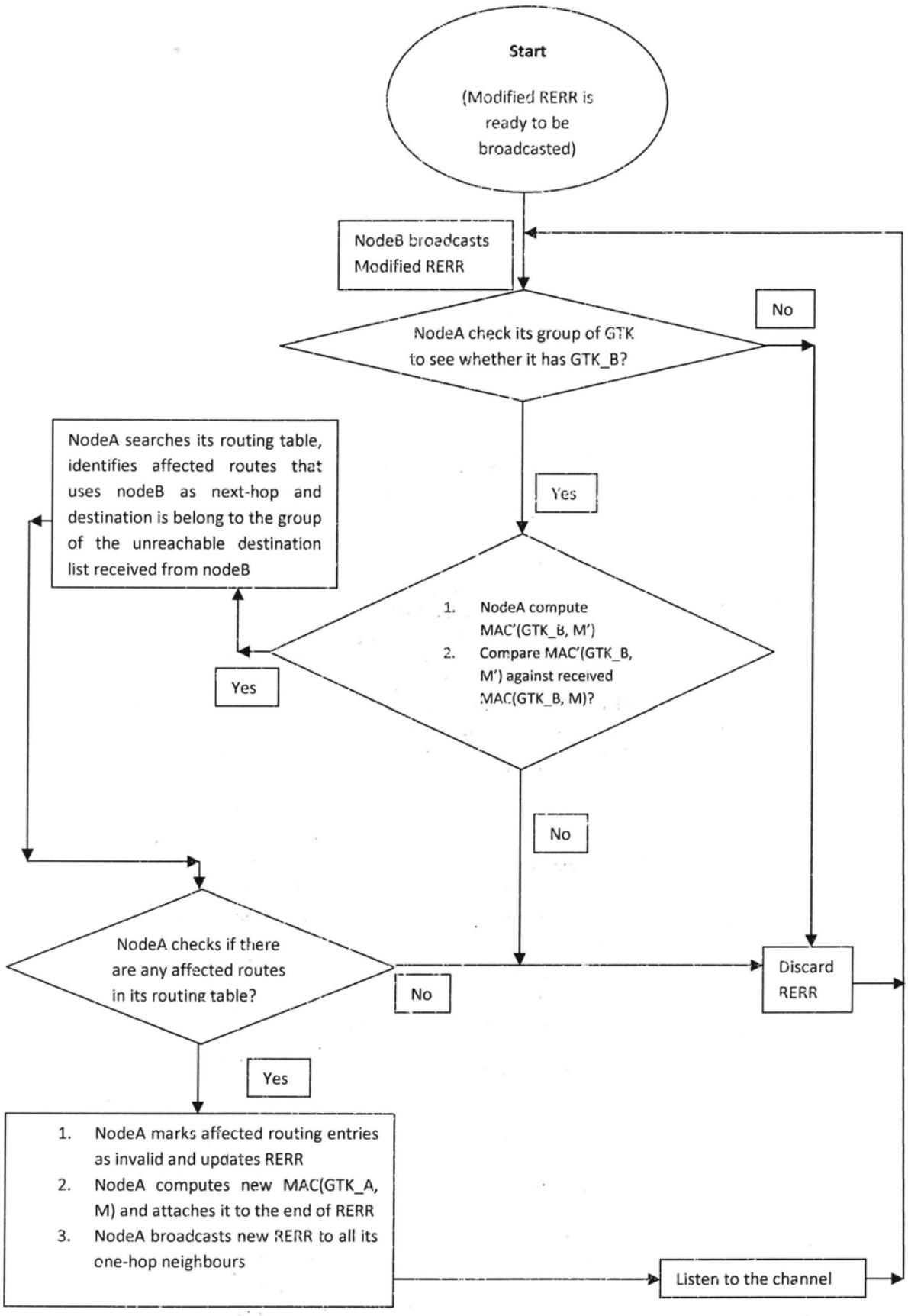


Fig. 6. 20 Route Maintenance Process_RERR_B (Node A's reaction, Flow Chart)

```
Begin {
    If (NodeA has GTK_B in GTK_List of NodeA)
    {    NodeA computes MAC'(GTK_B, M');
         If (MAC'(GTK_B, M') matches MAC(GTK_B, M))
         {    NodeA searches its routing table and finds out affected routes based on
              the information provided by the received RERR;
              If NodeA finds any affected routes in its routing table
              {    NodeA marks the affected route entries as invalid;
                   NodeA updates RERR;
                   NodeA computes new MAC(GTK_A, M) and attaches it to the end of RERR;
                   NodeA broadcasts new RERR to all its one-hop neighbours;
                   NodeA starts listening to the channel;
              } Else  NodeA discards RERR and starts listening to the channel
              Endif
         } Else  NodeA discards RERR and starts listening to the channel
         Endif
    } Else NodeA discards RERR and starts listening to the channel
    Endif
} End
```

Fig. 6. 21 Route Maintenance Process_RERR_B (Node A's reaction, Pesudocode)

Chapter 7

Security and Performance Analysis

In this chapter, we analyze our proposed SEAODV in defending against the attacking scenarios presented in Chapter 4 and compare the security analysis results against other three secure routing protocols: ARAN, SAODV and LHAP. Performance evaluation is also presented to prove that SEAOV is superior to ARAN and SAODV in terms of computation cost and route acquisition latency.

The remainder of this chapter is structured as follows.

Section 7.1 Depicts how SEAODV effectively defends against those identified attacks described in chapter 4.

Section 7.2 Explains why choosing routing protocols like AODV, ARAN, SAODV and LHAP as potential candidates to be compared against SEAODV.

Section 7.3 Summarizes vulnerabilities of various routing protocols based on the identified attacks given in chapter 4 and draws some conclusions in terms of superiorities SEAODV offers but others not.

Section 7.4 Presents performance evaluation in terms of computation cost and communication cost among ARAN, SAODV and SEAODV.

7.1 Security Analysis

1) RREQ Flooding Attack

In our proposed SEAODV, a node can participate in the route discovery process only if it has successfully established shared pairwise keys with its neighbours through key pre-distribution and key exchange phase. Hence it would not be possible for a malicious node to initiate a route discovery process with a destination address that is not in the network. Since every single RREQ is encrypted during transmission,

a malicious node cannot insert or modify any elements in these routing messages. However, malicious node can still hear the RREQ from other nodes and try to flood the heard RREQ to the network. Since nodes in the network only deal with incoming RREQs that originate from its one-hop neighbours, those flooded RREQs that are not in the active neighbour list of the receiving node will not be processed. That limits the scope of flooding RREQ in the entire network. In addition to this, pairwise key authentication somehow alleviate the side effect of the DoS attack due to its symmetric cryptography nature.

2) RREP Routing Loop Attack

This attack is formed due to the effect of both impersonation and modification of DSN (Destination Sequence Number). SEAODV is based on hop-by-hop authentication and all the fields of RREP have been encrypted with secret pairwise key (PTK key), hence eliminating the possibility of impersonation and modification attack. The only possible attack would be forwarding replayed routing messages by the attackers. Once again, this type of attack is only effective under the condition that the originator of the replayed message is in the active neighbour list of the receiving node; hence the effectiveness of this attack is restricted.

3) Route Re-direction Attack

The cause of route re-direction attacks is modification of mutable fields in routing messages such as DSN and hop count. These mutable fields will be authenticated at each hop. If any malicious node modifies the value of a field in transit, it will be readily detected by the next hop while comparing the new created MAC value with the received one. Whenever there is a miss-match in comparison, the tampered packet will be discarded. Therefore, the possibility of this type of attack has been eliminated.

4) Formation of Routing Loops Attack

Formation of routing loops requires gaining information in terms of network topology, spoofing and alteration of routing message. As all the routing information is encrypted between every two nodes in the network by either applying GTK or PTK, an adversary would be unable to spoof other nodes' IP and alter the fields in routing message. And simply forwarding replayed routing messages only benefits the adversary a little in terms of depleting energy of nodes and learning network topology.

5) Fabrication Attack

In AODV, a typical example of this attack is the fabrication of RERR messages. Again, arbitrarily fabricating RERR will be detected at the receiver by using the GTK key. Non-authentic routing messages will be dropped and the effectiveness of energy consumption is inconspicuous because of the one-hop neighbour effect, which means node only authenticates routing messages that come from its one-hop neighbours. Since SEAODV adopts shared pairwise key between every pair of nodes, only simple symmetric cryptographic operations are required for the authentication process, energy depletion issue and the corresponding incurred overhead would not be a critical issue in our scheme. That indicates our proposed scheme offers better resistance against the DoS attack and less computational overhead compared to those applied asymmetric cryptography operations in their schemes.

6) Tunnelling Attack

Tunnelling attacks often refer to two or more malicious nodes collaborate with each other to encapsulate and exchange messages between them through relatively high power transceivers or existing data routes in the network. The aim is to misrepresent the routing metrics, in this case hop count for AODV, and attract the subsequent data traffics to go through them. The more traffic they capture, the more information in terms of network topology, traffic patterns they gain. It is pretty tough to completely eliminate the tunnelling attacks. However, using an unalterable physical metric such as time delay can offer a dependable measure of path length. Using dependable metrics such as time delay can somehow alleviate the effect of tunnelling attack, but still cannot completely defend against this attack since two malicious nodes may use high-gain powerful transceiver to launch the tunnelling attack, which incurs shorter time delay. Therefore, fully preventing tunnelling attack is a tough task and it could have occurred in any secure routing protocol.

7.2 Why chooses routing protocols like AODV, ARAN, SAODV and LHAP to be compared against SEAODV?

The reasons are summarized as follows based on the characteristics of SEAODV.

• Our secure scheme is based on AODV routing protocol, therefore, comparison with AODV and its related secure version such as SAODV is desired. AODV offers no security features of the routing protocol itself end hence suffers from all different types of attacks as mentioned in the subsequent section 8.3. SAODV is a secure version of AODV, is uses hash chain to secure the hop count field of the routing message and applies digital signature to authenticate the origination of the routing messages that authentically come from either the source or the destination. This gives the adversary a chance to spoof the identity of the intermediate nodes between the source and destination, hence in this case, SAODV does not implement a hop-by-hop based secure routing mechanism and it is rather source-destination based. In the contrary, SEAODV is hop-by-hop and offers the best immunity to all various attacks given in section 8.3.

• We use symmetric cryptographic operation, in this case MAC to authenticate the received routing messages. Hence our scheme is lightweight compares to other routing protocols such as ARAN, which applies digital signatures to ensure the security of the routing message in a hop-by-hop fashion and ARAN incurs a large amount of overhead in terms of generation and verification of digital signatures. Although LHAP is indeed a lightweight transparent authentication protocol that resides between the data link and the network layer, it uses TESLA to maintain the trust relationship among nodes, which is not a realistic approach because of the delayed key disclosure period in TESLA. Also in LHAP, simply attaching the TRAFFIC key right after the raw message is not secure enough and the attacks it suffers are more severe than SEAODV. Our scheme is not only lightweight compares to ARAN, but also presents better security than LHAP.

• Our proposed idea is more suitable for applying for wireless mesh networks, where a WMN normally includes mesh routes and mesh clients. Mesh routers are referred to those nodes that do not involve any mobility; their locations in the WMNs are fixed. On the other hand, nodes that move arbitrarily in the WMNs are called mesh clients. Our SEAODV adopts Blom's key pre-distribution scheme to guarantee that each pair of nodes in WNNs can compute their unique shared secret pairwise key. Others secure routing protocols such as ARAN and SOADV do require the use of a third trusted certificate server T in order to distribute the certificate. The public key of the trusted server T is known to all available legitimate nodes in the network. The occurrence of the trusted certificate sever T means that asymmetric cryptography operations are performed and more computational energy are going to be consumed by each node that expects to be involved in the routing process. In the presence of the failure or malfunction of the trusted server T, a second backup trusted server T' needs to be added to enhance the security level of the entire network. As a consequence, this certainly brings additional complexity of the implementation of the

routing protocol and adds more cost to the actual system deployment. Surprisingly, the pre-distribution scheme used in SEAODV is simple, straightforward and secure as long as the total number of compromised nodes in the network is below to a certain threshold.

7.3 Vulnerabilities of various routing protocols

Attack	AODV	ARAN	SAODV	LHAP	SEAODV
RREQ Flooding	Yes	Yes	Yes	Yes	Yes
RREP Routing Loop	Yes	No	Yes	Yes	No
Route Re-direction	Yes	No	Yes	Yes	No
Formation of Routing Loops	Yes	No	No	Yes	No
RERR Fabrication	Yes	Yes	Yes	Yes	Yes
Tunnelling	Yes	Yes	Yes	Yes	Yes

Tab. 7. 1 Vulnerabilities of Various Routing Protocols

Summaries for each of the identified attacks from Tab. 7.1 are presented as follows.

RREQ Flooding

ARAN suffers badly from continuously verifying digital signatures, while SAODV also incurs massive overhead in signature verification process. Contrarily, LHAP offers better immunity due to its light-weight nature by using one-way hash chain and only authenticates RREQ from its one-hop neighbours. The number of hash operations required to verify the authenticity of a message is from single hash operation up to maximum number of tolerance in terms of packet loss. SEAODV only authenticates RREQ from nodes that are in the list of its active one-hop neighbours. Hash operations are required in SEAODV and re-creation of MAC is simple, fast and one time only.

RREP Routing Loop

In ARAN, every transmission of signed routing message makes impersonation and modification of sequence numbers impossible. SAODV is not based on a hop-by-hop authentication, but rather being a source-destination authentication and it means that any intermediate nodes could have been impersonated by any chance during the flying of RREP. LHAP uses one-way hash chain to protect the message by simply appending traffic key right after the raw message. Malicious node can simply block the wireless transmission between two neighbouring nodes, modifies the messages, put the corresponding intercepted traffic keys right after the messages and send them back to the wireless channel. SEAODV is a hop-by-hop authentication. GTK and PTK keys are used to secure the broadcast and unicast routing messages respectively. However, identification of the mutable and non-mutable fields in the routing message is not considered. The entire routing message is MACed either by applying GTK or PTK key. Therefore, possibilities of impersonation and modification are eliminated.

Route Re-direction

Both ARAN and SEAODV can defeat this type of attack. ARAN employs digital signature to sign every single routing message in a hop-by-hop fashion, while GTK and PTK keys are used in SEAODV to compute the MAC, which secures all the fields in the entire routing message. SAODV cannot effectively prevent the metric field (in this case hop count) from being increased by malicious nodes. This increases the chances of this route being de-selected from the potential candidate routes, which is another form of route re-direction attack. In LHAP, again malicious nodes could have blocked the wireless communication between any pair of neighbouring nodes for a short period, intercept the flying routing messages, modify them and attach the appropriate intercepted traffic keys after the modified routing messages before putting them back to the channel.

Formation of Routing Loops

Two conditions need to be satisfied in order to launch this attack. The malicious node has to impersonate a legitimate node in the network and is able to modify the metric such as hop count to be a better value in terms of less hop count in this case. SAODV is able to prevent the hop-count from decreasing, hence avoiding this attack. ARAN and SEAODV can also defeat this type of attack due to its hop-by-hop authentication. However, in LHAP, as long as the malicious node gets a chance to intercept the effective traffic keys and re-use them in a timely manner, there is a possibility to launch this type of attack.

RERR Fabrication

In ARAN, messages can only be fabricated by nodes with valid certificates and ARAN offers non-repudiation. Nodes keep sending fabricated routing messages might get excluded from the future route computation. While in SAODV, malicious node may simply impersonate nodes other than the one initiates the original RERR and forward the signed RERR to other nodes in the network. By doing do, malicious nodes can not only defeat the routing protocol, but also successfully deplete the energy of the nodes. LHAP also suffers from this type of attack; again malicious node could use the captured traffic key to be attached after the modified RERR as long as the captured traffic keys are still "fresh" enough to be authenticated by the receivers. SEAODV experiences least negative impact due to this attack since a receiving node only authenticates the RERR that comes from its active one-hop neighbours. This forces malicious node can only forward the replayed RERRs come from the receiving nodes' one-hop neighbours in order to launch this type of attack.

Tunnelling

ARAN uses physical metric, in this case the total time consumed in seeking a route. In other words, ARAN does not guarantee the shortest path in terms of hop count, but does offer the quickest path, which means whoever reaches the destination first in the route discovery process. This is still not enough to defeat the tunnelling attack because malicious nodes can simply adopt high-power gain transceiver to tunnel the routing messages such as RREP in order to make the source believe that the "tunnelled path" is the quickest path. As a consequence, malicious nodes would have been included on the final route towards destination and gained all the subsequent data packets passed through them. Similar methodology would be taken by malicious nodes to launch this attack on SAODV and SEAODV with the difference that now the actual routing metric is misrepresented in terms of hop counts.

LHAP only authenticates messages from its one-hop neighbours, it makes tunnelling attack become more tougher to be launched since malicious nodes now have to intercept the "fresh enough" traffic keys at both ends of the tunnel.

From Tab. 7.1, some conclusions in terms of benefits or superiorities of our SEAODV against other routing protocols can be drawn and given below.

1. Our proposed SEAODV is based on AODV
   AODV is a pretty mature on-demand routing protocol that uses hop count as the only routing metric to make routing decisions. The final established route between source and destination is

the one presents the least number of hop counts. There are plenty of resources that are available online for AODV and its essential implementation is also available on some of the network simulators such as NS2 [35], which means modification of the original version of AODV can be made and built directly on its implementation from these available network simulators. To the best of our knowledge, currently AODV has been considered one of the best candidates to be applied for Ad hoc and wireless mesh networks, although some modifications of the routing protocol itself need to be made in order to best accommodate the nature of the networks in terms of node mobility, energy constrain, network topology etc. Therefore, using AODV as the foundation of our routing protocol is implicitly a wise decision;

2. After successfully performing Blom's key pre-distribution scheme, which means every node in the network now possesses the secret PTK and GTK key. Our SEAODV is considered to be lightweight due to the sole involvement of MAC. Every node is only required to generate the MAC twice upon receiving a routing message from its one-hop neighbours. However, in ARAN and SAODV, every time a node receives a routing message, it needs to verify the signature first, only if the signature is verified, will it store the route and possibly generate its own signature before re-broadcasting or forwarding. As a result, our scheme incurs less overhead in terms of computation cost and offers better immunity to the attacks such as RREQ flooding attack, RERR fabrication attack compared to those secure routing protocols that rely on signature generation and verification;

3. In contrast to LHAP, which is also a lightweight authentication protocol, our proposed scheme is more practical and realistic. In LHAP, every node needs to maintain two different key chains. One is called traffic key chain to be used for authenticating the incoming packets; the other is TESLA key chain, which is used to maintain the trust relationship among nodes by authenticating keyupdate message. Keyupdate message is sent periodically to ensure that the current released traffic key is valid so that an obsolete traffic key will not be utilized by malicious nodes to forge a packet. Although LHAP is proved to be computationally efficient, but it requires each node to maintain two key chains, which adds more complexity in the authentication process. Compares to our scheme, LHAP is impractical since it does not eliminate the drawback of the delayed authentication in TESLA, which means the authenticity of the incoming packets and the traffic keys cannot be verified until TESLA key is verified to be authentic;

4. Although SEAODV cannot defeat tunnelling attack as others listed in the above Tab. 8.1, nodes only authenticate routing messages that come from their one-hop neighbours. Every node maintains a list of its active one-hop neighbours; routing messages are discarded if they come from nodes other than those from the active one-hop neighbour list. As a consequence, this constrains the scope of the flooding and fabrication attacks to be as far as the one-hop neighbours of a node and hence improving the resistance to the DoS (Denial of Service) attack;

5. The Blom's key pre-distribution scheme applied in our SEAODV is simple, straightforward, flexible and secure in contrast to those routing protocols where a trusted certificate server T is required in order to issue and distribute the valid certificate to every node in the network. There are two evident disadvantages of using trusted server. First, each node in the network must request a certificate from server T and each node only receives exactly one certificate from T. In order to communicate with certificate server T safely, different methods for secure authentication between these nodes and the server T are required. Blom's scheme does not rely on any secure authentication methods during the process of public key exchange. Second, by simply using single certificate server T can cause problem such as single-point failure attack, which means malicious nodes can pay great attentions to attack the certificate server T and try to make it failed to function properly; therefore, this brings to the failure of the routing protocol eventually. One of the possible solutions is to add another certificate server T' so that the reliability of the server and the routing protocol can be improved, but this also certainly comes with the price of adding more complexity and cost into the system's implementation and deployment. All these concerns are not applicable to Blom's scheme as long as the λ-secure property is followed, that is no nodes other than node i and j can compute the secret shared pairwise key Kij or Kji if no more than λ nodes in the network are compromised.

7.4 Performance Evaluation

This section presents performance evaluations among ARAN, SAODV and SEAODV routing protocols in terms of their computation cost and communication cost.

7.4.1 Computation cost

Computational cost is measured and computed at every node in the network. Since every node in the wireless mesh network can be looked upon as both the sender and the receiver, the total computation cost incurred at each node is going to be the cost of this node being as a sender plus the cost of the node being as a receiver.

The methodology mentioned above is applied to the evaluation of the computation cost for the following three routing protocols, ARAN, SAODV and SEAODV.

$Signature_{Gen}$	Signature generation cost
$Signature_{Ver}$	Signature verification cost
H	Hash operation cost
MAC	Cost for computing a MAC
$Max_{Hop_{Count}}$	Maximum hop count
Hop_{Count}	Number of hop count
N	Total number of nodes on the established route
$Broadcast$	Broadcast routing message
$Unicast$	Unicast routing message

Tab. 7. 2 Variables and Notations

Variables and notations used for computing computation and communication cost can be found in the above table.

a. ARAN

ARAN is a hop-by-by secure routing protocol for Ad hoc networks; it uses public key cryptography to guarantee integrity of routing message. However, a major drawback of ARAN is that it requires extensive signature generation and verification during the routing discovery process.

At each hop, every single routing message is signed before being re-broadcast or unicast. Upon being received, every signed message is verified in order to prove its authenticity. Only messages successfully verified remain, unauthentic routing messages are discarded.

In ARAN, all computation cost experienced at each hop comes from the extensive signature generation and verification. To be more detailed, during the routing discovery process, each sender generates its own signature and uses it to sign the entire routing message before sending it back to the wireless channel. Once the message is received on its fly to the destination, the receiver has to verify the signature(s) first before updating its routing table.

According to the operation of ARAN, receivers are required to be classified into two different categories. Receivers that are only one-hop away from the originator of the RREQ or RREP fall into the first category and those are more than one-hop away are referred to the second category. The reason is that receivers in different category incur different computational cost in terms of number of signature verifications.

Being a receiver in the first category, which means node is only one-hop away from the originator of RREQ or RREP, node is required to do two times of signature verifications if the routing message comes from the originator of the RREQ or RREP. The first signature verification is used for verifying the certificate of the originator of RREQ or RREP and obtaining the public key of the originator. The second one is required to verify the signature of the originator by using the public key of the originator. However, the node still needs to perform four times of signature verifications should the routing message come from node other than the originator of RREQ or RREP.

In contrast to the node being a receiver in the first category, node in the second category experiences four times of signature verifications when receives a RREQ or RREP from its one-hop neighbour. In addition to two times of signature verifications described in the last paragraph, another extra two times of signature verifications are a must due to the verification of both certificate and signature of the node from which it receives the RREQ or RREP.

Now the computation cost in terms of number of signature generation and verification can be derived and given below.

$Signature_{Gen}$	(sender)
$2 \times Signature_{Ver}$	(receivers that are ONLY one-hop away from the originator of RREQ or RREP)
$4 \times Signature_{Ver}$	(receivers that are more than one-hop away from the originator of RREQ

or RREP)

The following figure depicts a route setup process between a source S and a destination D. The route is comprised of five nodes. Node A and C are belong to the first category, which is only one-hop away from either the originator of RREQ (in the case, node S) or the originator of RREP (in this case, node D). Node B falls into the second category indicates that it is more than one-hop away from the originator of both the RREQ and RREP.

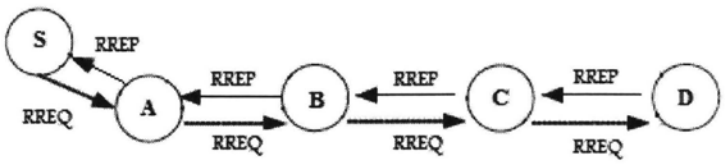


Fig. 7. 1 Route Setup Process

By using the above Fig. 7.1 as an example, the total computation cost for a final established route with N nodes between source S and destination D can be expressed in the equation given below.

$$2 \times (N-4) \times (Signature_{Gen} + 4 \times Signature_{Ver}) + 2 \times [(Signature_{Gen} + 2 \times Signature_{Ver}) + (Signature_{Gen} + 4 \times Signature_{Ver})] + 2 \times (Signature_{Gen} + 4 \times Signature_{Ver}) \quad (1)$$

Giving N equals to five and by using the above figure, we can take a one step closer look on equation 1 as it can be divided into three parts and each of them defines the computational cost for a certain number of nodes on the established route. The first part $2 \times (N-4) \times (Signature_{Gen} + 4 \times Signature_{Ver})$ gives the computation cost for node B. The second part $2 \times [(Signature_{Gen} + 2 \times Signature_{Ver}) + (Signature_{Gen} + 4 \times Signature_{Ver})]$ presents the computation cost for node A and node C. While the last part $2 \times (Signature_{Gen} + 4 \times Signature_{Ver})$ represents the computation cost for node S and node D.

Equation (1) indicates that as the number of nodes on the established final route increases, the number of intermediate nodes who are at least two hops away from the originator of RREQ or

RREP also rises, hence the total computational cost of all the nodes on the final route are going to boost up.

b. SAODV

SAODV is a secure variant of AODV. Routing operations are similar to that of AODV; however, it applies cryptographic extensions to provide authenticity and integrity of routing message. It uses hash chains to prevent manipulation of hop count field and digital signature to secure the rest of the routing message. However, an adversary can still increase the hop count.

Since SAODV offers two types of signature extensions, named single signature and double signature extensions. In the evaluation of computation cost for SAODV, we only consider the single signature extension due to its simplicity. By using single signature extension, intermediate nodes cannot reply to a RREQ message simply because it cannot properly sign its RREP message. Alternatively, it just behaves as if it did not have the route and forwards the RREQ message. The only node that can reply to a RREQ is the destination itself.

Before rebroadcasting a RREQ or forwarding a RREP, a node needs to apply the hash function to the Hash value in the signature extension in order to account for the new hop. If the node itself is the originator of RREQ or RREP, then it is not required to perform the hash operation, but the cost of generating digital signature should be included.

Upon receiving the RREQ or RREP, the receiver is required to apply the hash function h ($Max_{Hop_{Count}} - Hop_{Count}$) times to the value in the hash field in order to secure the hop count. Apart from that, the receiving node also needs to verify the signature generated by the originator of the RREQ or RREP.

The computation cost of the node being as a sender or a receiver in SAODV has been given below.

$Signature_{Gen}$	(sender, originator of RREQ or RREP)
H	(sender, intermediate node)
$H \times (Max_{Hop_{Count}} - Hop_{Count}) + 2 \times Signature_{Ver}$	(receiver)

By using the computation cost described above and Fig. 7.1, the total computation cost can be derived and presented in the following equation (2).

$$2 \times [\sum_{i=0}^{N-3} H + H \times (Max_{Hop_{Count}} - i) + 2 \times Signature_{Ver}] + 2 \times \{Signature_{Gen} + H \times Max_{Hop_{Count}} + H \times [Max_{Hop_{Count}} - (N-2)] + 2 \times Signature_{Ver}\} \quad (2)$$

Equation (2) can also be interpreted in more detail by dividing it into two different parts. The first part $2 \times [\sum_{i=0}^{N-3} H + H \times (Max_{Hop_{Count}} - i) + 2 \times Signature_{Ver}]$ describes the computation cost for node A, node B and node C; while the second part $2 \times \{Signature_{Gen} + H \times Max_{Hop_{Count}} + H \times [Max_{Hop_{Count}} - (N-2)] + 2 \times Signature_{Ver}\}$ presents the computation cost for node S and node D.

Equation (2) indicates that as the total number of nodes on the finalized route increase, more hash operations and signature verifications are required to be performed during the route set up process.

c. SEAODV

The computation cost of SEAODV is simple and straightforward in contrast to that of ARAN and SAODV. In SEAODV, the computation cost involved to every node on the route is exactly the same whenever the node acts to be a sender or a receiver.

The following presents the computation cost for a node as being a sender or a receiver.

MAC (sender)
MAC (receiver)

MAC stands for the cost of computing MAC of the entire routing message by either using the GTK or PTK key.

By using Fig. 7.1 and the computation cost given above, the total computation cost for a finalized route with N nodes can be deduced in equation (3).

$$2 \times (N-2) \times 2 \times MAC + 2 \times 2 \times MAC \quad (3)$$

From equation (3), our scheme only involves the operation of MAC and with no signature at all. In the next following parts, evaluation results proved that SEAODV does offer superior performance in terms of computation cost and route acquisition latency as compares to other two secure routing protocols, ARAN and SAODV.

d. Evaluation of Computation Cost

In this part, three secure routing protocols (ARAN, SAODV and SEAODV) are evaluated in terms of computation cost by using equation (1), (2) and (3).

The computation cost for computing signature, hash operation and MAC are listed in the following Tab. 7.3, which is obtained from [36] and the timing results in the table are based on the experimental results. In [36], different cryptographic primitives have been implemented in based on the open source Crypto++ library in the PDA platform with an Intel Xscale 400 MHz CPU, 64 MB SDRAM, and 32 MB Flash ROM. These implemented primitives include RSA, SHA-1 and HMAC.

The RSA in the Tab. 7.3 listed below is 1024 Bits with exponent being 17; SHA-1 is one input block; HMAC is one input block.

Abbreviation	Definition	Computational time (ms)
$Signature_{Gen}$	RSA Signature generation	33.3
$Signature_{Ver}$	RSA Signature verification	1.42
H	SHA-1	0.009
MAC	HMAC	0.015

Tab. 7. 3 Computational Cost of Cryptographic Operations

By applying the cryptographic costs from the above table in equation (1), (2) and (3) and setting N to be 10, 30, 50, 70 and 100 respectively, the computation cost for secure routing protocols ARAN, SAODV and SEAODV can be calculated. The following tables list the corresponding computation cost for each routing protocol.

ARAN:

N	Computational cost from equation 1 (ms)
10	695.96
30	2255.16
50	3814.36
70	5373.56
100	7712.36

Tab. 7. 4 Computation Cost for ARAN

SAODV:

N	Computational cost from equation 2 (ms)
10	122.976
30	241.796
50	353.416
70	457.836
100	600.966

Tab. 7. 5 Computation Cost for SAODV

SEAODV:

N	Computational cost from equation 3 (ms)
10	0.54
30	1.74
50	2.94
70	4.14
100	5.94

Tab. 7. 6 Computation Cost for SEAODV

By using the Tab. 7.4, Tab. 7.5 and Tab. 7.6, the comparison among these three secure routing protocols (ARAN, SAODV and SEAODV) can be conducted in terms of computation cost in millisecond.

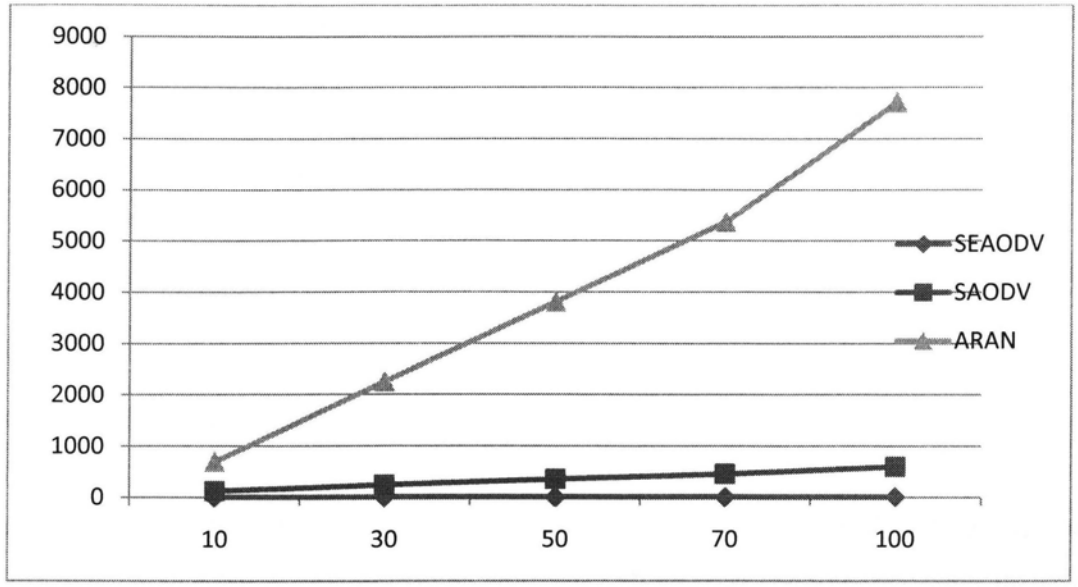


Fig. 7. 2 Computation Cost for ARAN, SAODV and SEAODV

From the above Fig. 7.2, it can be seen that the computation cost almost can be negligible for our SEAODV in contrast to SAODV and ARAN. Ours is much faster (1000 folds on average) than ARAN and almost 100 folds quicker on average than SAODV.

The superiorities of our SEAODV are summarized below.

- Extreme low computation cost as the number of nodes on the selected route increases;
- Better immunity against to DoS attack in terms of energy consumption;
- Extends entire lifetime of the selected route under the condition that a certain number of nodes on the selected route are classified as mesh client, which tends to be power constrained;
- As being a mesh client, low computation cost simply means longer lifetime the node can enjoy and share valuable information with other nodes in the network;
- Computation cost is computed in terms of timing expense in millisecond, indicates that SEAODV performs excellent in authentication latency due to its efficient cryptographic operation.

7.4.2 Communication cost

ARAN, SAODV and our SEAODV are all similar in the way of discovering routes. RREQ is broadcast by the originator of the on-demand node towards to the destination. Upon receiving the RREQ,

destination unicasts the RREP back to the source from which the RREQ is generated by using the reverse path which has been set up during the flooding of the RREQ. All of these routing protocols apply the same methodology in their routing mechanisms. Therefore, the communication cost involved in ARAN, SAODV and SEAODV are the same. In other words, the number of routing messages required to establish a secure path, which is defined as the definition of communication cost are the same.

By using the Fig. 7.1, the communication cost is given below in equation (4).

$$(N-1) \times Broadcast + Hop_{Count} \times Unicast \tag{4}$$

However, ARAN, SAODV and SEAODV experience various number of control bytes within every routing message such as RREQ and RREP. The more number of control bytes incurred in a single routing message, the larger the entire routing message. Therefore, routing message with bigger size in terms of bytes tends to have a lower probability of successful reception at the destination and suffer longer delay.

Before computing the latency produced by the communication overhead for each of the routing protocols mentioned above, the following assumptions are made:

1. Network topology used is the same as the one given in the route setup process in Fig. 7.1;
2. Network throughput is defined to be 400kbps for a single flow (in this case, a single pair of source and destination) by using relevant simulation results and parameters given in [37];
3. Every node in the network only processes RREQ and RREP once and no other types of routing messages are processed;
4. Signature used in ARAN and SAODV are based on 1024 bit RSA algorithm with exponent being 17 and the signed message is 1024 bits;
5. HMAC is used in our SEAODV with output being 128 bits;
6. In ARAN, the size of RREQ or RREP generated by the source or destination is smaller than those forwarded by intermediate nodes, which include two signatures and two certificates. While the RREQ or RREP originates by either the source or destination is only comprised of one signature and one certificate. Presume that the route discovery packet (RDP) in ARAN is the same size as that of used in AODV, which is 24 bytes. Therefore, for RREQ and RREP with single signature and certificate, the total size is 312 bytes which is the same as that of in SAODV given below. For RREQ or RREP with double signatures and double certificates, the total size is extended to 568 bytes;

7. In SAODV, 312 bytes in total for both RREQ and RREP, which include original AODV message (24 bytes), signature (128 bytes), top hash (16 bytes), hash (16 bytes) and certificate (128 bytes);
8. In SEAODV, there are totally 40 bytes for either RREQ or RREP. The AODV message costs 24 bytes and the HMAC is 16 bytes.

Once again, by using Fig. 7.1, the total number of RREQ and RREP incurred during the entire route setup process can be derived for ARAN, SAODV and SEAODV.

For an established route with N nodes on it, the total number of RREQ and RREP is given in equation (5).

$$(N-2)\times 2+2 \tag{5}$$

Therefore, the total number of bytes that are required to be transmitted in order to safely setup a route for ARAN, SAODV and SEAODV can be computed. The corresponding latency for transmitting those required bits can be calculated by using equation (6).

$$Communication\ Cost = Tansmission\ Latency = \frac{Total\ No.of\ bits\ need\ to\ be\ transmitted\ (bits)}{Network\ throughput\ (\frac{bits}{s})} \tag{6}$$

ARAN:
There are only two routing messages with single signature, others are double signatures.

The total number of bytes required to be transmitted in order to ensure a secure route set up is given in equation (7).

$$568\times(N-2)\times 2+312\times 2\ \text{Bytes} \tag{7}$$

N	Total number of bits required (Bits)	Latency required for transmitting number of bits given in column two (ms)
10	9712	24.28
30	32432	81.08
50	55152	137.88

70	77872	194.68
100	111952	279.88

Tab. 7. 7 Total Number of Bytes and Latency Required for ARAN

SAODV:

In SAODV, all routing messages are the same size, hence the total number of bytes required to be transmitted is computed in equation (8).

$$312 \times (N - 2) \times 2 + 312 \times 2 \text{ Bytes} \tag{8}$$

N	Total number of bits required (Bits)	Latency required for transmitting number of bits given in column two (ms)
10	5616	14.04
30	18096	45.24
50	30576	76.44
70	43056	107.64
100	61776	154.44

Tab. 7. 8 Total Number of Bytes and Latency Required for SAODV

SEAODV:

Similarly, total number of bytes incurred during the route set up process is presented in equation (9).

$$40 \times (N - 2) \times 2 + 40 \times 2 \text{ Bytes} \tag{9}$$

N	Total number of bits required (Bits)	Latency required for transmitting number of bits given in column two (ms)
10	720	1.8

30	2320	5.8
50	3920	9.8
70	5520	13.8
100	7920	19.8

Tab. 7. 9 Total Number of Bytes and Latency Required for SEAODV

Now the average route acquisition latency can be derived by using the following equation (10).

$$Average\ Route\ Acquisition\ Latency = Computation\ Cost + Comunication\ Cost \quad (10)$$

Average route acquisition latency is the average delay between the sending of a RREQ packet by a source for discovering a route to a destination and the receipt of the first corresponding RREP.

The following Tab. 7.10, 7.11 and 7.12 summarize the total cost required to safely setup a route between a source and a destination for the three routing protocols in terms of route acquisition latency in millisecond.

ARAN:

N	Computation cost from equation 1 (ms)	Communication Cost (ms)	Total Latency (ms)
10	695.96	24.28	720.24
30	2255.16	81.08	2336.24
50	3814.36	137.88	3952.24
70	5373.56	194.68	5568.24
100	7712.36	279.88	7992.24

Tab. 7. 10 Average Route Acquisition Latency for ARAN

SAODV:

N	Computation cost from equation 2 (ms)	Communication Cost (ms)	Total Latency (ms)

10	122.976	14.04	137.016
30	241.796	45.24	287.036
50	353.416	76.44	429.856
70	457.836	107.64	565.476
100	600.966	154.44	755.406

Tab. 7. 11 Average Route Acquisition Latency for SAODV

SEAODV:

N	Computation cost from equation 3 (ms)	Communication Cost (ms)	Total Latency (ms)
10	0.54	1.8	2.34
30	1.74	5.8	7.54
50	2.94	9.8	12.74
70	4.14	13.8	17.94
100	5.94	19.8	25.74

Tab. 7. 12 Average Route Acquisition Latency for SEAODV

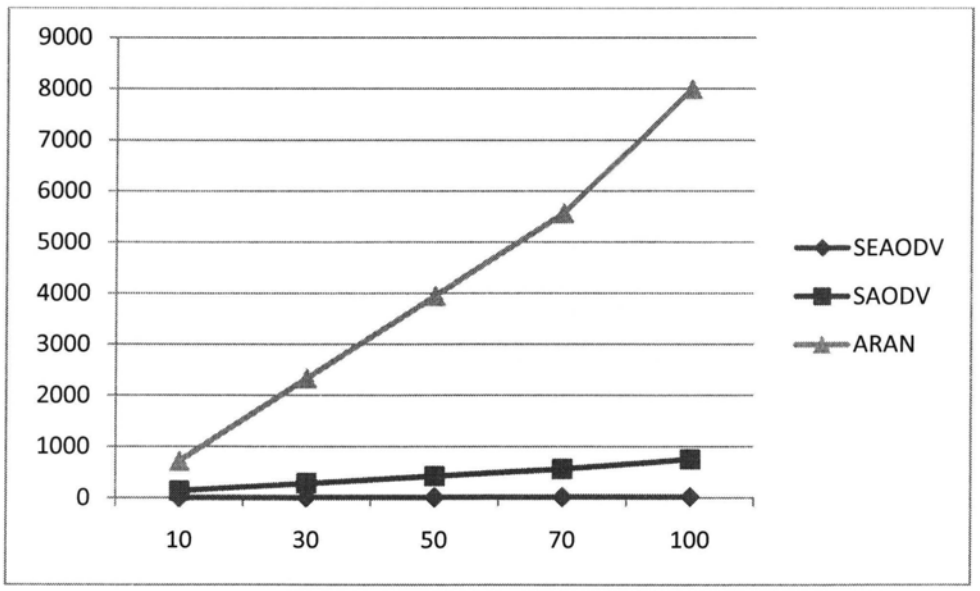


Fig. 7. 3 Average Route Acquisition Latency for ARAN, SAODV and SEAODV

Fig. 7.3 indicates that the total cost (latency in millisecond) of our SEAODV can be ignored in contrast to SAODV and ARAN due to the use of MAC and the smaller size of routing message. In SEAODV, communication cost plays a significant role in contributing to the total cost. While in ARAN, communication cost has almost no impact on the computation of the total cost. For SAODV, the impact of communication cost is sitting between that of ARAN and SEAODV.

Chapter 8

Conclusions and Future Works

This chapter summarizes what we have done and presents the possible future work.

In our proposed SEAODV, Blom's key pre-distribution scheme is used to establish keys to ensure that every two nodes in the network uniquely share the pairwise keys. Our secure routing scheme is based on AODV, but adding secure AODV extensions (in this case, Message Authentication Code) to the original AODV routing messages.

Each node in the network possesses two types of keys: PTK (Pairwise Transient Key) and GTK (Group Transient Key), where PTK is computed by using Blom's scheme, node makes use of PTK to accomplish the distribution of GTK. PTK is used for protecting the unicast routing message and every pair of nodes secretly shares their own PTK. While GTK is used for securing the broadcast routing message and shared privately between the node and its one-hop neighbours. MAC is computed by using either PTK or GTK depending on the type of the routing message (in this case, unicast or broadcast).

So far, the research work have been done are as follows.

- We use Blom's key pre-distribution scheme to establish the PTK between each pair of nodes in the network and employ created PTK to distribute GTK.

- We identify and summarize various attacking scenarios AODV could have possibly faced, present security analysis for ARAN, SAODV, LHAP and SEAODV and conclude that our SEAODV outperforms other three in terms of defending against those identified attacking scenarios.

- We proposed a secure variant of AODV called SEAODV (Security Enhanced AODV), in which only symmetric cryptographic primitives (such as PTK, GTK) and operations (in this case, MAC) are employed to secure the routing messages. SEAODV implements a hop-by-hop authentication.

- We proved that SEAODV presents less computation cost and better route acquisition latency as compares to other two secure routing protocols SAODV and ARAN through performance evaluation.

- Once a route is safely established, PTK can also be used to encrypt the subsequent data traffic in a hop-by-hop fashion towards to the destination. This indicates that both the routing message and the subsequent data can be secured in SEAODV while ARAN and SAODV do not consider protection of the subsequent data after successfully establishing the route.

In contrast to other secure routing protocols (ARAN, SAODV) that use asymmetric cryptography, our SEAODV only employs symmetric cryptographic primitives and implements a hop-by-hop authentication mechanism that secures both the routing messages during route discovery process and the data traffics subsequently.

For future work, some aspects given below might be worth of pursuing.

- Using network simulator such as ns2 [35] to implement SEAODV, SAODV and ARAN respectively.

- Under condition that absolutely no attacks in the network, performing simulations for the three secure routing protocols in terms of packet delivery ratio, network throughput, average route acquisition latency and other possible routing performances.

- Making some nodes in the network be attackers who can launch some of the identified attacks described in Chapter 4 and run the simulation of the above three secure routing protocols to observe whether SEAODV still outperforms other two routing protocols under a certain number of attacks.

- Enrich SEAODV by incorporating proactive routing methodology so that the hybrid version of SEAODV can be introduced. Future simulations could be done on the hybrid SEAODV to see whether it is better fit for WMNs.

Appendix

Acronyms

LQSR:	Link Quality Source Routing
DSR:	Dynamic Source Routing
LBAR:	Load Balancing Aware Routing
DLAR:	Dynamic Load Aware Routing
TESLA:	Timed Efficient Stream Loss tolerant Authentication
HWMP:	Hybrid Wireless Mesh Protocol
MP:	Mesh Point
MR:	Mesh Router
MC:	Mesh Client
AP:	Access Point
MSR:	Mesh networks Scalable Routing
ETX:	Expected Transmission count
ETT:	Expected Transmission Time
MHC:	Maximum Hop Count
TTL:	Time To Live
IP:	Internet Protocol
RDP:	Route Discovery Packet
DSN:	Destination Sequence Number
OSN:	Originator Sequence Number
AES:	Advanced Encryption Standard
DES:	Data Encryption Standard
RSA:	Rivest, Shamir and Adleman cryptosystem
MANETs:	Mobile Ad hoc NETworks
AODV:	Ad hoc On-demand Distance Vector routing
SAODV:	Secure Ad hoc On-demand Distance Vector routing
ARAN:	Authenticate Routing for Ad hoc Networks
SEAODV:	Security Enhanced Ad hoc On-demand Distance Vector routing
MAC:	Message Authentication Code
HMAC:	Hash Message Authentication Code
H:	Hash Function

WMNs:	Wireless Mesh Networks
LHAP:	Lightweight Hop-by-hop Authentication Protocol for ad hoc networks
RREQ:	Route REQuest message
RREP:	Route REPly message
RERR:	Route ERRor message
PTK:	Pairwise Transient Key
GTK:	Group Transient Key
DoS:	Denial of Service

Bibliography

[1] Richard Draves, Jitendra Padhye, Brian Zill. 2004. Routing in multi-radio, multi-hop wireless mesh networks. In ACM Mobicom.

[2] C.E. Perkins, E. Belding Royer, S.R. Das. July 2003. Ad hoc on-demand distance vector routing. IETF RFC 3561

[3] David B Johnson and David A Maltz. 1996. Dynamic source routing in Ad hoc wireless networks. In Mobile Computing. Volume 353.

[4] Hossam Hassanein, Audrey Zhou. 2001. Routing with load balancing in wireless Ad hoc networks. In ACM MSWiM.

[5] Sung Ju Lee, Mario Gerla. 2001. Dynamic load-aware routing in Ad hoc networks. In IEEE ICC.

[6] M. Zapata, N. Asokan. September 2002. Securing ad-hoc routing protocols. In Proceedings of ACM Workshop on Wireless Security (WiSe).

[7] Sangiri, K., Dahil, B. 2002. A Secure Routing Protocol for Ad Hoc Networks. In Proceedings of 10th IEEE International Conference on Network Protocols.

[8] Hu, Y.-C., Perrig, A., Johnson, D.B. September 2002. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In Proceedings of MobiCom, Atlanta, GA.

[9] Perrig, A., Canetti, R., Tygar, J.D., Song, D. 2000. Efficient authentication and signing of multicast streams over lossy channels. In Proceedings of IEEE Symposium on Security and Privacy. 56-73.

[10] Gergely, L.B., Vajda, I. 2006. Provably secure on-demand routing in Mobile Adhoc Networks. In IEEE transactions on Mobile Computing. 1533-1546.

[11] IEEE 802.11s Task Group. July 2007. Draft Amendment to Standard for Information Technology Telecommunications and Information Exchange Between Systems – LAN/MAN Specific Requirements –

Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: Amendment: ESS Mesh Networking, IEEE P802.11s/D1.06.

[12] Bahr, M. August 2006. Proposed Routing for IEEE 802.11s WLAN Mesh Networks. In 2nd Annual International Wireless Internet Conference (WICON), Boston, MA, USA.

[13] Bahr, M. 2007. Update on the Hybrid Wireless Mesh protocol of 802.11s. In Proceedings of IEEE International Conference on Mobile Adhoc and Sensor Systems, MASS. 1-6.

[14] S. Zhu, S. Xu, S. Setia, and S. Jajodia. May 2003. LHAP: A Lightweight Hop-by-Hop Authentication Protocol for Ad-Hoc Networks. In ICDCS International Workshop on Mobile and Wireless Network, Providence, Rodhe Island.

[15] A. Menezes, P. VanOorschot, S. Vanstone. October 1996. Handbook of Applied Cryptography.

[16] Deborah Russell, G.T. Gangemi, Sr. 1991.Computer Security Basics. O'Reilly & Associates, Inc.

[17] http://en.wikipedia.org/wiki/Key_management

[18] http://en.wikipedia.org/wiki/Public-key_cryptography

[19] http://en.wikipedia.org/wiki/Cryptographic_hash_function

[20] http://en.wikipedia.org/wiki/Message_authentication_codes

[21] M. L. Sichitiu. 2005. Wireless mesh networks: Opportunities and challenges. In proceedings of the Wireless World Congress.

[22] I. F. Akyildiz, X. Wang, and W. Wang. 2005. Wireless mesh networks: A survey. Computer Networks.

[23] R. Bruno, M. Conti, and E. Gregori. 2005. Mesh networks: Commodity multihop ad hoc networks. IEEE Communications Magazine. Volume 43, 123-131.

[24] MIT Roofnet. http://www.pdos.lcs.mit.edu/roofnet/

[25] Meshnetworks. http://www.meshnetworks.com

[26] De Couto DSJ, Aguayo D, Bicket J, Morris R. 2003. A high-throughput path metric for multi-hop wireless routing. In Proceedings of the 9th Annual International Conference on Mobile Computing and Networking. 134-146.

[27] Draves R, Padhye J, Zill B. August 2004. Comparison of routing metrics for static multi-hop wireless networks. In Proceedings of the 2004 Conference on Apllications, Technologies, Architectures, and Protocols for Computer Communications.

[28] Draves R, Padhye J, Zill B. 2004. Routing in multi-radio, multi-hop wireless mesh networks. In Proceedings of the 10th Annual International Conference on Mobile Computing and Networking. 114-128.

[29] Chen Hongsong, Fu Zhongchuan, Wang Chengyao, Ji Zhenzhou, Hu Mingzeng. 2007. Using Network Processor to Establish Security Agent for AODV Routing Protocol. In Journal of Computing and Information Technology – CIT. 61-70.

[30] Md. Shariful Islam, Young Jig Yoon, Md. Abdul Hamid, and Choong Seon Hong. 2008. A secure hybrid wireless mesh protocol for 802.11s Mesh Network. In ICCSA 2008, Part I, LNCS 5072. 972–985.

[31] Wenliang Du, Jing Deng, Y.S. Han, P.K. Varshney. October 2003. A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks. ACM.

[32] R. Blom. 1985. An optimal class of symmetric key generation systems. Advances in Cryptology: Proceedings of EUROCRYPT 84 (Thomas Beth, Norbert Cot, and IngemarIngemarsson, eds.), Lecture Notes in Computer Science, Springer-Verlag, 209. 335–338.

[33] Ian D. Chakeres, Elizabeth M. BeldingRoyer. 2002. The Utility of Hello Messages for Determining Link Connectivity. In Proceedings of the 5th International Symposium on Wireless Personal Multimedia Communications (WPMC).

[34] Xiangpeng Jing, Myung J. Lee. January 2004. Energy-Aware Algorithms for AODV in Ad Hoc Networks. In Proceedings of Mobile Computing and Ubiquitous Networking. Yokosuka, Japan.

[35] The network Simulator, http://www.isi.edu/nsnam/ns

[36] M. Long, and John Wu. February 2006. Energy-Efficient and Intrusion-Resilient Authentication for Ubiquitous Access to Factory Floor Information. IEEE Transactions on Industrial Informatics. Volume 2, 40-47.

[37] Yuxia Lin, A. H. M. Rad, Vincent W.S. Wong, Joo-Han Song. October 2005. Experimental Comparisons between SAODV and AODV Routing Protocols. WMuNeP'05.