A NOTE ON THE SECURITY OF MANETS

Jiazi YI, Polytech'Nantes
March, 2008

ABSTRACT

This note is based on the bibliography of security issues of networks, especially mobile ad hoc networks. The purpose of this note is to give a quick review on the security related problems in MANETs, and to answer the question "at the moment, what do we have to deal with the security". We first introduce the basic concepts of network security, and then come to the security changes and requirements. The key management and secure routing schemes are presented in section 3 and section 4.

1. BASIC OF NETWORK SECURITY

Security is a broad and essential topic in computer networks. Most security problems are intentionally caused by malicious people trying o gain some benefit, get attention or to harm someone. In its simplest form, the security is concerned with making sure that nosy people cannot read, or worse yet, secretly modify messages intended for other recipients. The fundamental elements of the network security are presented in this section, which include cryptography, symmetric/public key algorithms, digital signatures and authentication protocols.

1.1. CRYPTOGRAPHY

In cryptography, the message to be encrypted, known as the plaintext, are transformed by a function that is parameterized by a key. The output of the encryption process, known as the ciphertext, is then transmitted, often by messenger or radio. We assume that the enemy, or intruder, hears and accurately copies down the complete ciphertext. The art of breaking ciphers, called cryptanalysis, and the art devising them is collectively known as cryptology.

A fundamental rule of cryptography is that one must assume that the cryptanalyst know the methods used for encryption and decryption. This idea is called *Kerckhoff's principle*:

Kerckhoff's Principle: All algorithms must be public; only the keys are secret

From the cryptanalyst's point of view, the cryptanalysis problem has three principal variations: *ciphertext-only problem*, *know plaintext problem* and *chosen plaintext problem*.

And the encryption methods are also divided into two categories: *substitution cipher,* in which each letter or group of letters is replaced by another letter or group of letters to disguise it. One of the oldest known ciphers is the *Caesar cipher*. *Transposition ciphers*, in contrast, recorder the letters but do not disguise them.

Although there are many different cryptographic systems, there are two principles underlying all of them:

Redundancy: Messages must contain some redundancy.

Freshness: Some method is needed to foil replay attacks.

For the evaluation of the cryptographic primitives includes:

- Key size;
- Level of security;
- Functionality;
- Methods of operation;
- Ease of implementation;
- Performance.

1.2. SYMMETRIC KEY ALGORITHMS

A basic requirement for the symmetric key scheme is that the parties involved in the communication share a common key. This implies that the shared key must be distributed over a secure communication channel.

The working of a symmetric key cryptographic operation is shown in Figure 1. The plaintext messages are encrypted at the sender using the encryption key E. The resulting ciphertext can then be transmitted over the channel (wireless or wireline or a combination). Intruders are assumed to have access to the ciphertext on the channel as shown in the figure. Passive intruders can just record the ciphertext while active intruders can attempt to

modify it. The ciphertext then reaches the receiver where it is decrypted using the decryption key D as shown. The decryption results in recovery of the original plaintext assuming that the ciphertext has not been modified in transmit by an active intruder. Typically both the E and the D keys are the same and this common key is typically called as the shared key. Such symmetric key schemes can be used to achieve confidentiality, integrity, and authentication.

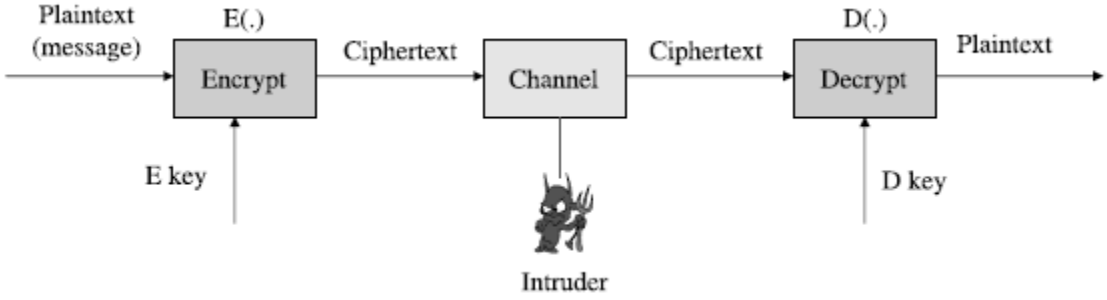

Figure 1 Basic operation of symmetric key cryptography

Two of the most famous symmetric ciphers are Data Encryption Standard (DES) and Advanced Encryption Standard (AES).

DES was developed by IBM and was adopted by the US Government in Jan 1977 as its official standard. It was widely used by the industry for use in the security products. The plaintext is encrypted in blocks of 64 bits, yielding 64 bits of ciphertext. The algorithm, which is parameterized by a 56-bit key, has 19 distinct stages.

AES is a replacement of DES. It has been designed to known attacks and exhibits simplicity of design. Originally called as Rijndael, this was issued as a Federal Information Processing Standards (FIPS) standard in November 2001. AES uses three different key sizes, namely 128, 192, and 256 bits. The block size is 128 bits. It treats data in four groups of four bytes, each called the state. AES has been designed to have one of 9, 11, or 13 rounds. In each round the state undergoes four operations, namely byte substitution, shifting of rows, mixing of columns and XORing with the subkey of the round. All operations can be combined into XOR and table lookups. Hence these are very fast and efficient.

However, AES and DES is basically a monoalphabetic substitution cipher using big characters. Symmetric ciphers like AES operate on fixed size input messages in blocks of 128 bits. Given this, a question arises as to the mode of operation of these ciphers when messages which have arbitrary length (larger than the size of an input message) have to be encrypted. There are five modes of operation which are typically used in such cases and are applicable to any fixed-length encryption scheme. These are:

- Electronic code book (ECB).
- Cipher block chaining (CBC).
- K-bit cipher feedback (CFB).
- K-bit output feedback (OFB).
- Counter mode (CTR).

1.3. Public Key Algorithms

In 1976, two researchers at Stanford University, Diffie and Hellman, proposed a radically new kind of cryptosystem, one in which the encryption and decryption keys were different, and the decryption key could not feasibly be derived from the encryption key. In

their proposal, the encryption algorithm, *E*, and the decryption algorithm, *D*, had to meet three requirements. These can be stated as follows:

1. *D(E(P)) = P.*
2. It is exceedingly difficult to deduce *D* from *E*.
3. *E* cannot be broken by a chosen plaintext attack.

This is referred as public key, also asymmetric algorithms. The asymmetric approach uses two keys, a public key and a private key. The public key may be known to anybody while the private key is supposed to be known only to the entity creating a message. Every entity in the network wishing to send messages has these two keys, namely a private and a public key. Both the keys are different but related mathematically. Additionally, it is computationally infeasible to derive the private key knowing only the public key and the cryptographic algorithm. Derivation of the private key would require additional information. Note, however, that it should be computationally easy to encrypt or decrypt messages when the relevant key is known.

There are RSA algorithm, ElGamal algorithm and Rabin algorithm which and be used for both encryptions as well as digital signatures. RSA is the best know and widely used public key encryption algorithm. It was devised by Rivest, Shamir and Adleman of MIT in 1977. It is based on exponentiation in a finite field over integers modulo a prime. The security of this scheme is based on the intractability of the integer factorization problem. It is obvious that this problem is trivial when dealing with small numbers but becomes intractable when dealing with large numbers. Hence, the numbers used in the RSA algorithm need to be very large (1024 bits and beyond).

However, given the performance overhead associated with the public key algorithms, these algorithms are typically not used for encrypting data. Rather, they are typically used to transport symmetric keys between the parties that need to exchange data securely. In case of key transport, one of the parties involved decides the key. This party then encrypts the key using the public key of the other party and sends the message to the other party, who can then obtain the key. A problem with this scheme is that the key is decided by a single party. In some cases, it might be preferable to let both the parties contribute equally to the resulting key. This is referred to as key agreement.

The Diffie-Hellman protocol is such kind of key exchange protocol. The objective of DH protocol is to establish a common key between the participants of the protocol. The common key is expected to be used to exchange messages securely using symmetric key ciphers. Further, the common key is known only to the participants of the key exchange protocol. The Diffie-Hellman protocol itself does not ensure entity authentication. This implies that the participants in the protocol do not have guarantees of the identities of each other. Therefore, these participants need to use other techniques in order to verify the authenticity of each other. The DH key exchange protocol is based on exponentiation in a finite (Galois) field modulo a prime number. This requires the ability to find large primes as well as the ability to find primitive roots for large primes. It also requires the ability to carry out efficient modular arithmetic. The security of the protocol depends on a hard problem. This hard problem is the difficulty of computing discrete logarithms.

1.4. DIGITAL SIGNATURES

The authenticity of many legal, financial, and other documents is determined by the presence or absence of an authorized handwritten signature. For computerized message systems to replace the physical transport of paper and link documents, a method must be

found to allow documents to e signed in an unforgivable way. The following requirements are needed:

- The receiver can verify the claimed identity of the sender.
- The sender cannot later repudiate the contents of the message.
- The receiver cannot possibly have concocted the message himself.

Symmetric-Key Signatures: Use a central authority that knows everything and whom everyone trusts to distribute the symmetric key.

Public-Key Signatures.

Message Digests: It can be used for the situation in which authentication is needed but secrecy is not. It is based on the idea of a one-way hash function that takes an arbitrarily long piece of plaintext and from it computes a fixed-length bit string. It has four important properties:

- Given *P*, it is easy to compute *MD(P)*.
- Given *MD(P)*, it is effectively impossible to find *P*.
- Given *P* no one can find *P*' such that *MD(P') = MD(P)*.
- A change to the input of even 1 bit produces a very different output.

To meet the 3rd criterion, the hash should be at least 128 bits long, preferably more. To meet the 4th criterion, the hash must mangle the bits very thoroughly, not unlike the symmetric-key encryption algorithms we have seen. Computing a message digest from a piece of plaintext is much faster than encrypting that plaintext with a public-key algorithm, so message digests can be used to speed up digital signature algorithms. There are two kinds of hashing algorithms: unkeyed hash functions, which do not require any secret key; and keyed hash functions, which are referred as message authentication codes (MAC).

MD5 is one of the most widely used ones of message digest functions. It is the fifth in a series of message digests designed by Ronald Rivest. It operates by mangling bits in a sufficiently complicated way that every output bit is affected by every input bit.

The other major message digest function is ***SHA-1***(Secure Hash Algorithm 1), developed by NSA and blessed by NIST in FIPS 180-1. Like MD5, SHA-1 processes input data in 512-bits blocks, only unlike MD5, it generates a 160-bit message digest.

2. SECURITY CHALLENGES AND REQUIREMENTS

2.1. VULNERABILITIES

While ad hoc networks provide a great flexibility in establishing on-the-fly communications links, they also bring a lot of research challenges at the same time. The truth is that the underlying characteristics of wireless ad hoc networks make them highly vulnerable. First, the use of wireless communication and the freedom of mobility place the network nodes at a greater risk of being captured, compromised or hijacked physically than their wired counterparts. Second, lack of a predefined infrastructure means that there is no centralized control for the network services. The network functions in a distributed fashion by cooperative participation of all nodes. The decentralized decision making is prone to attacks that are designed to break the cooperative algorithms. Third, dynamically changing topology aids the attackers to update routing information maliciously by pretending to be legitimate topological changes. Fourth, a constrained network bandwidth may limit the number and size of the message transmitted during a protocol execution. Fifth, energy- and

resource-constrained mobile nodes are sensitive to the use of computationally complex algorithms. Finally, node selfishness is a security issue specific to ad hoc network. Since routing and network management are carried by all the available nodes in ad hoc networks, some nodes may selfishly deny the routing request from other nodes to save their own resources (e.g., battery power, memory, CPU).

Based on the fundamental vulnerabilities of wireless hoc networks, various potential attacks have been widely exploited and can be classified into passive attacks or active attacks, depending on the nature of the attack. Figure 2 shows a categorization of the attacks.

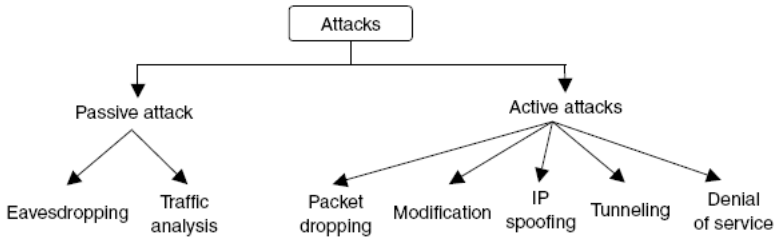

Figure 2 Classification of attacks

All the attacks can be generated either by an outside intruder or by an inside intruder. The attacks caused by the nodes that do not belong to the networks are considered as *external attacks*, while the attacks created by the compromised internal nodes are called *internal attacks*. Internal attacks usually have a more serious effect in degrading network performance, since the compromised node would have access to all the cryptographic keys of the network and it may also cooperate with other adversaries or compromised nodes.

2.2. GOALS AND REQUIREMENTS

Similar to traditional networks, the goals of securing an ad hoc network can be characterized by the following attributes: availability, confidentiality, integrity, authentication, and non-repudiation:

Availability ensures that the intended network services are available to the intended parties whenever needed.

Confidentiality ensures that the transmitted information can be accessed only by the intended receivers and is never disclosed to unauthorized entities. It protects the information from threats based on eavesdropping.

Authentication allows a node to validate the identity of the peer node it is communicating with. It guards messages against impersonation, substitution, or spoofing. Without authentication, an adversary can masquerade as a node, thus gaining unauthorized access to resource and sensitive information and interfering with the operation of other nodes.

Integrity guarantees that information is never corrupted during transmission. Only the authorized parties are able to modify the information. It ensures that any modification of the messages will be detected. Message modification includes writing, changing, deleting, creating, and delaying, or replaying of a transmitted message.

Non-repudiation ensures that an entity can prove the transmission or reception of information by another entity, that is, a sender/receiver cannot falsely deny having received or sent certain data.

On addition to the basic security goals, there are also some new challenges for the ad hoc networks.

First, the critical issue concerning the design of security service in ad hoc networks is not to rely on any centralized entity, and the security management should be implemented in a distributed fashion. The two commonly used architectures in regular wired and wireless networks — centralized and hierarchical architectures — are vulnerable to the loss of the central controlling node and are not suitable for ad hoc networks. Instead, distributed strategies are often preferred and have shown significance in recent research.

Second, the low resource availability necessitates its efficient utilization and prevents the use of complex authentication and encryption algorithms. Light-weight authentication and encryption schemes with resource awareness are required.

Third, an integrated security scheme that combines intrusion prevention and intrusion detection mechanism is necessary. Intrusion prevention approaches can efficiently deal with the attacks from outsiders by constraining the network access control, and intrusion detection can efficiently detect some active attacks from inside the network. Therefore, a security scheme combining these two mechanisms is better suited to secure ad hoc networks.

Fourth, various ad hoc network applications exhibit numerous inconsistencies, and designing an efficient security scheme for a generalized ad hoc network makes the problem complex and more difficult to address. The level of security should depend on the type of network application under consideration. There is a tradeoff between the network capabilities and security, as a very secure system may become too complex, slow, and difficult to administer. If the information transmitted is not critical and the network is used in a friendly environment, a lower level of security can be achieved using some simple security mechanisms.

Therefore, an efficient security scheme for ad hoc networks should be an integration of intrusion prevention and intrusion detection mechanisms, which must be implemented in a distributed fashion with resource awareness and a reasonable degree of security.

3. KEY MANAGEMENT

Nodes in a network need to be able to communicate securely with each other. The existence of secure communication channels is especially crucial in ad hoc networks on account of the use of wireless links and other characteristics of such networks. These channels are required for many operations such as exchanging data or exchanging control packets in the case of functions like routing. To make such secure communication possible, it is necessary for nodes to have access to the proper keying material. This is the objective of the key management process. This has lately been a very active area of research in ad hoc networks.

The importance of key management cannot be overemphasized for both traditional and ad hoc networks. When employing cryptographic schemes, such as encryption or digital signatures, to protect both control and data traffic, a key management service is always required. For secure communication between any two entities, both the entities should possess a secret value or key. The possible ways in which secure communication can be established are for the entities concerned to share a key (symmetric-key system) or for the entities concerned to possess different keys (asymmetric-key system). Key management is the process by which those keys are distributed to nodes on the network and how they are further updated if required, erased, and so on. There are several steps that key management

has to be concerned with for both symmetric key systems as well as asymmetric key systems. These include

1 Initializing the system users.
2 Creating, distributing and installing the keying material.
3 Organizing the use of the keying material.
4 Updating, revoking, and destroying the keying material.
5 Archiving the keying material.

Key management in ad hoc networks, however, is more difficult than in traditional networks. This is because of several factors, such as the vagaries of wireless links, lack of a central authority, constraints on resources such as power, memory, and bandwidth availability, and the inability to predetermine the neighbors of a node after deployment, which is further worsened on account of the mobility of nodes in such networks.

3.1. ASYMMETRIC KEY-BASED APPROACH

The traditional approach towards developing an asymmetric key-based system is based on the use of a CA, as discussed earlier. However, such an approach is not practical in ad hoc networks for several reasons. Firstly, a CA will be a vulnerable point in the network, especially if it is not distributed. Compromise of a CA will allow an adversary to sign any certificate, thereby paving the way for impersonation of any node or for revocation of any certificate. More importantly, in order to carry out the key management operations, the CA will have to be accessible all the time. If the CA is unavailable, then the nodes in the system might be unable to update/change keys. New nodes will also not be able to obtain certificates. An approach to improving the availability would be to replicate the services of the CA, but a naive replication of the CA could lead to more problems. Compromise of any single replica could lead to collapse of the entire system. An approach to solving this problem is to distribute the trust reposed in a single CA over a set of nodes, thereby letting the nodes share the responsibility of key management.

3.1.1. PARTIALLY DISTRIBUTED AUTHORITY

The partially distributed authority is based on the concept of threshold cryptography (TC). A TC scheme makes it possible for n parties to share the ability to perform a cryptographic operation. For example, consider the digital signature on a message. We have seen techniques whereby a single user creates the digital signature. A problem occurs, however, when this user is compromised or cannot be trusted. A better approach then is to distribute the trust placed on a single user among multiple users. This indeed is what threshold cryptography strives to achieve.

The objective of threshold cryptography is to protect information by distributing it among a set of n entities. In addition, there is a threshold t associated with the TC schemes such that any t of the n parties can execute the cryptographic operation. Such schemes are referred to as (n,t) TC schemes. In case of an (n,t) TC scheme, fewer than t parties will not be able to execute the cryptographic operation successfully. Thus, C can be considered to be an approach for secure sharing of a secret. We see from here that, even when some number of entities (less than the threshold t) in the network is compromised, the system is not at risk. Nonavailability of certain number of nodes (at most n 2t nodes, to be precise) in the network will also not have an impact on the working of the system. Note that the TC schemes perform the cryptographic operation in a distributed manner.

In [5], the authors propose using a scheme based on the technique of threshold cryptography to distribute the private key of the certification authority. Knowledge of this

key is distributed over a subset of the nodes in the network. The system, made up of the nodes in the network, is expected to have a public – private key pair. This key pair is created initially by a trusted authority before deployment of the nodes. Following that, the private key is divided into n shares using an (*n, t+1*) threshold cryptography scheme. These n shares are then allocated to n arbitrarily chosen nodes by the authority that created the public – private key pair. These chosen nodes are called servers. Following this distribution of the shares of the private key to the servers, the central authority is no longer needed. Thus, the central authority is only needed during the bootstrapping phase. Each server also has its own key pair and stores the public keys of all the nodes in the network. In particular, each server (chosen node) knows the public keys of other servers. As a result, the servers can establish secure links among themselves. We show the initial configuration of such a service in. The service as a whole has a public – private key pair K – k. The public key K is known to all nodes while the private k is divided into shares $s_1, \ldots, s_n$, with each server having one share. Each server also has a public – private key pair *Ki – ki.*

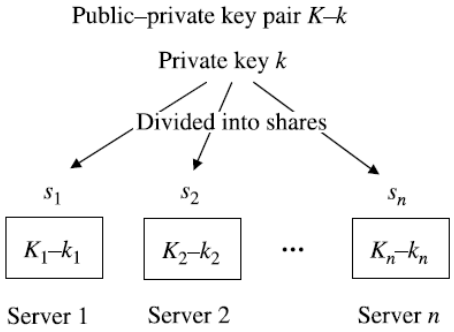

Figure 3 Configuration of partially distributed authority

Whenever a certificate has to be signed using the private key of the system, the servers are contacted. Each server generates a partial signature for the certificate using the share of the private key that the server has. The partial signature is then submitted to a combiner that computes the overall signature from the partial signatures. Note that the combiner will not be able to create the overall signature without the partial signatures.

Since this scheme is based on the concepts of threshold cryptography, the system can tolerate a certain number of compromised servers. Thus, *t* or fewer than t compromised servers will not be able to derive the private key of the system. This is because compromised servers (assuming there are at most *t* of them) cannot generate correctly signed certificates by themselves, since they can generate at most t partial signatures. Further, not all *n* servers are needed to generate the overall signature, but any *t+1* servers will suffice. Figure 4 shows how servers generate a signature using a (3, 2) threshold signature scheme. Each server generates a partial signature PS(m,s_i) for message *m* using its share of the key. The combiner *c* can generate the signature $<m>_k$ even though server 2 does not provide a partial signature.

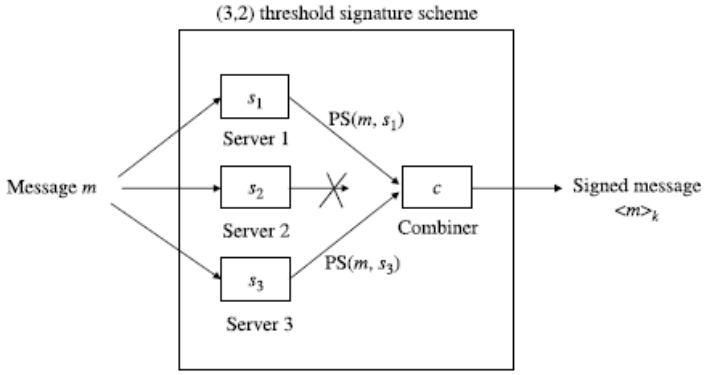

Figure 4 Example of signature generation with threshold cryptography

The authors also propose to refresh the keys proactively in order to tolerate mobile adversaries, because the adversary will have to capture multiple shares during the same interval. Thus, if the old keys under the control of the adversary are refreshed, then they will be of no use with the new keys that the adversary might compromise. Additionally, the system can also be designed to adapt its configuration to the changes in the network. For example, a key management service might start with a (7, 3) configuration but then modify this later to a (4, 2) combination since some servers were detected to be compromised while other servers were unavailable.

3.1.2. FULLY DISTRIBUTED AUTHORITY

The study [6] describes an approach for distributing the functions of a certification authority. The difference compared to partially distributed scheme, however, is that in this case any node instead of a chosen few can contain a share of the private key of the service. A new node that does not have a certificate will have to contact at least t+1 servers. These servers can issue a certificate to such a node after establishing the identity of the node. Any t+1 nodes can also renew a certificate. In addition, a node that does not possess a share can obtain one by contacting any group of at least t+1 nodes that already possess the share. The bootstrapping of the system, however, needs a trusted authority that provides the initial shares to the first t+1 nodes.

3.1.3. SELF-ISSUED CERTIFICATES

The self-organized approach that allows users to create, store, distribute, and revoke their own public keys without the help of any trusted authority. Such a self-organized public key management approach for ad hoc networks has been considered in [7].

Each user decides their own public-private key pair. Other users issue certificates to each other based on other factors such as personal acquaintances. Each user then maintains in their personal repository the list of valid certificates. When a user A needs to authenticate a certificate belonging to user B, then user A checks if a certificate has been issued by itself to B earlier. If not, user A checks if the repository contains a certificate to any other user C who in turn has B’ s certificate. Thus, a chain of valid public key certificates is obtained by user A such that the first certificate of the chain is one issued by A. Further, each remaining certificate in the chain can be verified using the public key contained in the previous certificate of the chain. The last certificate of the chain should then contain the public key of user B. Thus, certificate chaining is used in order to authenticate public keys of users. If A

cannot form a certificate chain to B, then A cannot authenticate B’ s public key. In addition to this, another problem is related to bootstrapping the certificate repository of a node without depending on a separate authentication channel.

There are also a lot of asymmetric schemes introduced in [2].

3.2. SYMMETRIC KEY-BASED APPROACH

The symmetric key-based approach is mainly focused on the sensor networks. A practical solution given these constraints is to load the keys on the nodes before the nodes deploy. Thus the nodes have some secret information on them; using this they set up secure communication infrastructure for use during the operation of the network.

Several solutions based on predeployed keying have been proposed, including approaches based on the use of a global key common to all nodes, approaches in which every node shares a unique key with one or more nodes in the network, and approaches based on each node being deployed with a random set of keys. These approaches can be loosely divided into two main categories: **deterministic schemes** and **probabilistic schemes.** Deterministic schemes have a deterministic relationship between the keys loaded on a node and the identity of the node. More precisely, the existence of a secure link between any two nodes in the network can be predicted exactly. Compromise of nodes in such networks can result in secure communication between noncompromised nodes also becoming vulnerable, but the non-compromised nodes that will be impacted as a result of such compromised nodes can generally be precisely determined. In case of probabilistic schemes, the keys loaded on a node are chosen randomly from a pool. Thus, a secure link between any two nodes in the network exists with a certain probability. Capture of nodes by an adversary in such networks will also result in the compromise of secure communication between non-compromised nodes, but the non-compromised nodes that will be impacted as a result cannot be precisely determined.

3.2.1. DETERMINISTIC

In [8], the author uses a single key for the entire network. They consider sensor networks consisting of tamper-resistant nodes (with the nodes being called pebbles). Hence, the keys on the nodes are not compromised even when the nodes fall into enemy hands. All nodes are initialized with a single symmetric key before deployment. This saves on storage while also minimizing the costs associated with key management. This single key is then used to derive the keys used to protect data traffic. However, this approach of using a single key in the entire network is problematic from a security standpoint since the compromise of a single sensor will break the security of the entire network communication. This will also make selective key revocation difficult.

Pairwise secret sharing avoids this problem of compromise of communication in the entire network following the leakage of a single key. In fact, this scheme is very resilient since compromising any node does not impact the security of communication among any other noncompromised nodes. However, the scheme places great demands on the amount of storage needed on each sensor node. This makes it an impractical solution for large networks. For example, with networks of n nodes, each node will contain (n-1) keys for a total of n(n-1)/2 keys in the entire network, which makes it impractical for networks with, say, 10,000 nodes. Note also that many of these keys would not be used since direct communication between nodes is possible only if the nodes are neighbors. All the nodes in the network cannot be neighbors of each other since the neighborhood sizes are limited due to the communication range and density of nodes. This gives rise to inefficient usage of memory storage while also requiring large memory sizes on the nodes. This solution also

makes it difficult to add more nodes to a deployed system than intended initially since this involves rekeying all the nodes already deployed with the keys corresponding to the new nodes to be deployed. The procedure to load keys into each sensor node also adds to the costs associated with key management.

There are also other deterministic approaches, such as SPINS [9] and LEAP (Localized Encryption and Authentication Protocol)[10] .

3.2.2. PROBABILISTIC

A probabilistic key sharing approach was first proposed in [11]. In that scheme each node is loaded with one or more keys before deployment. These keys are randomly chosen from a pool of keys. After deployment a secure link can be established between a pair of neighboring nodes provided a key happens to be common to both these nodes.

The proposed probabilistic scheme, also called the random key predistribution scheme, consists of three phases namely:

- Key predistribution;
- Shared key-discovery; and
- Path-key establishment.

Other than the deterministic and probabilistic schemes, there are also some works take the location of the nodes or the mobility of the nodes into consideration. For more details, refer to [2].

4. SECURE ROUTING

There are three generic attacks the routing for ad hoc networks:

Wormhole attacks typically require the presence of at least two colluding nodes in an ad hoc network. The malicious nodes need to be geographically separated in order for the attack to be effective. In this attack, a malicious node captures packets from one location and “tunnels” these packets to the other malicious node, which is assumed to be located at some distance. The second malicious node is then expected to replay the “tunneled” packets locally. There are several ways in which this tunnel can be established.

Rushing attacks impacts a reactive routing protocol. Note that, in the case of a reactive routing protocol, a node that needs a path to a destination floods the network with route request packets. Such route request packets are flooded in a controlled fashion in the network. Thus, every node only forwards the first route recovery packet that it receives and drops the rest. The adversary can exploit this feature of reactive routing protocols. It does so by “rushing” the route request packets towards the destination. As a result, the nodes that receive this “rushed” request forward it and discard the other route requests that arrive later. The resulting routes would then include the adversary, which places the adversary in an advantageous position.

Sybil attacks consist of a node assuming several node identities while using only one physical device. The additional identities can be obtained either by impersonating other nodes or by making use of false identities. These identities can all be used simultaneously or over a period of time. This attack can impact several services in ad hoc networks. For example it can impact multipath routing where a set of supposedly disjoint paths can all be passing through the same malicious node which is using several Sybil identities. This attack can also impact data aggregation where the same node can contribute multiple readings each using a different identity. Fair resource allocation mechanisms will also be affected

since a node can claim more than its fair share by using the various Sybil identities. Mechanisms based on trust, voting, and so on, are also affected by this attack.

Next are some secure protocols for ad hoc networks.

4.1. SECURE OLSR

There are some schemes proposed for extending OLSR to make it secure against attacks. The main idea behind their propositions is to use digital signatures for authenticating OLSR routing messages. Such kind of authentication may be done on a hop-by-hop or on an end-to-end basis.

[12] focus on the hop-by-hop approach, in which each node signs OLSR packets as they are transmitted (such packets may contain multiple OLSR messages originated by a variety of nodes). The next hop verifies the authenticity of the message, strips the signature from the previous node, and adds its own signature. Therefore, the signature only verifies that the node that forwarded the traffic is the one that signed the message but does not verify the authenticity of the original message. Authentication of messages is based on symmetric keys that nodes share and the signature is created using some kind of a hash function such as SHA-1.

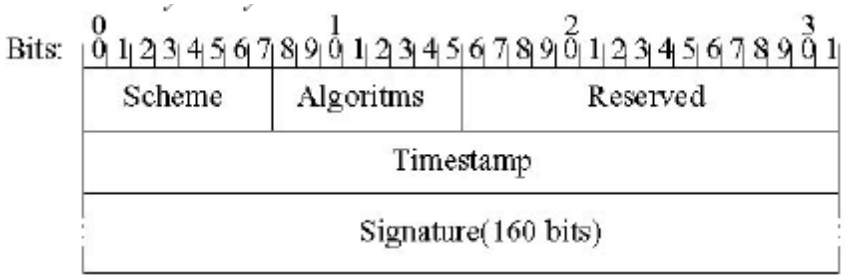

Figure 5 The basic signature message

Figure 5 shows the basic signature extension that is attached to each OLSR packet. The signature is generated by applying the hash function (based on the scheme and algorithm defined in the message) on the OLSR packet header, the OLSR routing messages contained in the OLSR packet, the fields in the signature extension shown in Figure 5 except for the signature field, and the shared secret key. The timestamp field is needed in this scheme so that malicious nodes cannot launch replay attacks by just moving to a different neighborhood and replaying messages they recorded earlier.

In order for this scheme to work, nodes need to know the current time of their neighbors. This does not require that nodes synchronize their clocks. It only requires that the nodes know the approximate time difference between themselves and their neighbors. It is also assumed that clocks move forward at about the same speed. In order for neighbors to discover the relative time of their neighbors, the authors propose the following scheme. When node A needs to discover the relative time of its neighbor, it initiates the timestamp exchange process by sending a challenge message as shown in Figure 6.

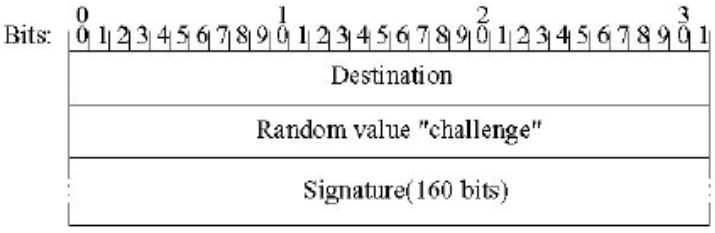

Figure 6 The timestamp exchange challenge message

The destination field contains the IP address of the node (say node B) that node A is trying to get the time value for. The random value field contains a random number to avoid replay attacks and the signature is created by applying the hash function as described earlier. The destination node, that is, Node B, then verifies the authenticity of the challenge message and then responds with a challenge-response message with the format shown in Figure 7.

The challenge-response message contains the IP address of node A, the random number, and its timestamp. The response signature field is created by applying the hash function to the IP address of node B, the random challenge, and the shared key. The signature field is produced by applying the hash function to the entire message and the shared key. When node A receives the challenge-response from node B, it first verifies its authenticity (by applying the hash functions as node B and comparing the results with what is contained in the message). Node A then sends its own timestamp to node B by creating a response-response message as shown in Figure 8.

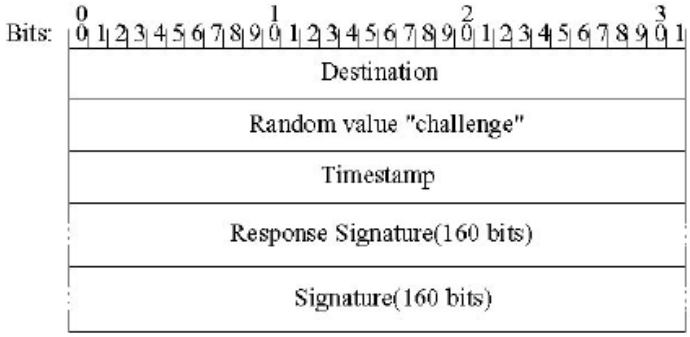

Figure 7 The timestamp challenge-response message

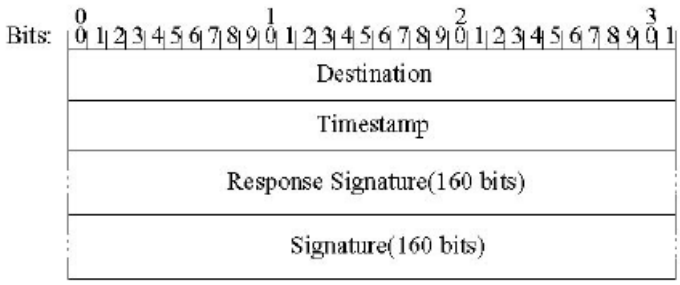

Figure 8 The timestamp response-response message

[13] discusses schemes for authenticating OLSR messages on end-to-end basis so that nodes receiving OLSR message can authenticate the node that generated the original message rather than just the node forwarding the message.

When nodes send out OLSR messages they attach an ADVanced SIGnature (ADVSIG) message. Nodes that advertise links sign the messages so that the origins of the message can be authenticated. Another key concept in this scheme is that, when nodes advertise a link to another node (e.g. through TC messages), they attach a proof that the link actually exists through a signed hello message from that node. This approach of securing OLSR requires a PKI scheme or some other key management scheme that can be used for verifying authenticity of messages. The scheme also requires a timestamp scheme [14] that allows nodes to have consistent timestamps.

SLSP (Secure link-state routing protocol) [15] is a proposed scheme for securing link-state routing where security is achieved via the use of asymmetric primitives. SLSP assumes that each node in the network has a public – private key pair. Each node broadcasts its certified key to all nodes within its zone (i.e. all nodes that are within R hops from it). These broadcasts are periodic or when conditions require it (e.g. when there are substantial changes in the network topology), which enables new nodes entering the zone to discover the key. Certification of public keys can be achieved through a distributed certificate authority. SLSP can be employed as a stand-alone protocol, or fit naturally into a hybrid routing framework, when combined with a reactive protocol. SLSP is robust against individual attackers, it is capable of adjusting its scope between local and network-wide topology discovery, and it is capable of operating in networks of frequently changing topology and membership.

4.2. SECURE DSR

Secure routing protocol (SRP) is a protocol that can be used as an extension of a variety of routing protocols in order to add various security related attributes to these protocols, including DSR.

SRP assumes that there is a bi-directional security association (SA) between nodes that wish to exchange routing messages. The two nodes can share a secret key which they use for protecting the routing messages they exchange. SRP focuses on the nonmutable fields of the routing messages and uses the keys that the two nodes share for signing the routing messages and hence ensuring that those portions of the messages are not tampered with. It is believed that signing only the nonmutable fields of the DSR messages is sufficient for ensuring acceptable operation of the routing protocol.

SRP defines a header that is added to each routing message of the underlying protocol (e.g. DSR). When a node initiates the route discovery process it generates a route request message and attaches the SRP header to it.

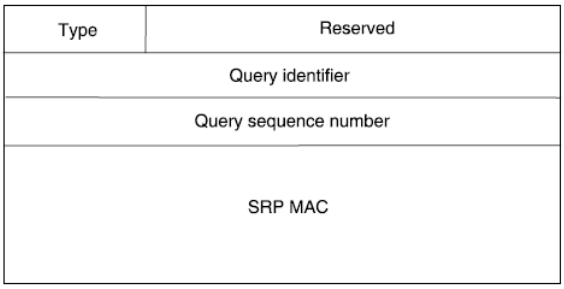

Figure 9 SRP header format

The message authentication code (MAC) is a 96-bit long field generated by a hash function (e.g. SHA-1, MD5). The hash function takes as an input the entire IP header, routing message, the SRP extension, and the shared key between the originator and destination of the routing message. The mutable fields are excluded from this calculation. The mutable fields include any IP-header mutable fields and the list of hops on the route that are added to the route request as it travels towards the destination. The SRP header is attached with the packets being sent in the network for the intermediate nodes and the destination to indentify the receiver.

Ariande [17] ensures authenticity of information provided by intermediate nodes in the path between the source and the destination. It focuses on DSR and provides verification for much of the information provided by DSR. It provides similar security properties to SAODV, allowing nodes to authenticate routing messages and ensuring that

those messages were not altered by nodes forwarding those messages. And Endaria [16] proposes that each intermediate node should sign the route reply instead of signing the route request as in Ariadne. In fact, the word "Endaria" is a reverse of "Ariande".

4.3. SECURE AODV

A secure version of AODV called Secure AODV (SAODV) has been proposed in [18][19]. SAODV provides features such as integrity, authentication, and nonrepudiation of routing data. SAODV incorporates two schemes for securing AODV. The first scheme involves nodes signing the messages that they create (e.g. RREQ, RREP). This allows other nodes to verify the originator of the message. This scheme can be used for protecting the portion of the information in the RREQ and RREP messages that does not change once these messages are created. However, RREP and RREQ messages also contain a field (namely the hop count) that needs to be changed by every node. Such mutable information is ignored by the creator of the message when signing the message. The second scheme of SAODV is used for protecting such mutable information. This scheme leverages the idea of hash chains.

4.4. SUMMARY

Securing the routing protocol for ad hoc networks is a widely discussed topic. Other than those schemes presented above, there are a lot of protocols being introduced.

SEAD (Secure Efficient Distance Vector Routing) [20] attempts to address these problems while providing a secure version of DSDV. SEAD prevents a malicious node from increasing the sequence number or decreasing the metric associated with a destination in the routing updates the malicious node sends out. SEAD does this while making use of one-way hash chains. It provides two main mechanisms, namely a scheme to ensure the security of the metric and sequence number authentication as well as a scheme for neighbor authentication.

To prevent traffic analysis, requires a routing protocol that maintains the anonymity of nodes in the network. Such anonymous routing protocols are orthogonal to the secure routing protocols that we have seen earlier. ANODR [21] and MASK [22], are such anonymous routing protocols designed to prevent traffic analysis in ad hoc networks. These protocols do this by hiding the sender and/or the recipient's identity from an outside observer. This makes it impossible in some cases and harder in others for an adversary to correlate eavesdropped traffic information to network traffic patterns. The approach here is to remove the identities of the nodes present in the packet. Instead, the packets make use of link identifiers. These link identifiers are created between two neighbors and hence an intended receiver can easily recognize a packet destined for it. On the other hand, the link identifier appears as a random number to other nodes. Further, these link identifiers change for every link due to which an adversary will not be able to trace the path of a packet.

In fact, additional security properties can be incorporated in the routing protocol depending on the needs of the application using the network. Note that each of these properties comes with a cost (e.g. additional bandwidth overhead, CPU processing overhead):

- timeliness can be achieved through the use of timestamps;
- ordering can be achieved through the use of sequence numbers;
- authentication can be achieved by using a public key infrastructure and requiring each node to sign the messages they generate and attach their signature to the message;
- authorization can be achieved through the use of credentials;
- integrity of routing messages can also be achieved using digital signatures;

- confidentiality can be achieved using encryption, as discussed earlier;
- nonrepudiation can be achieved by requiring each node to sign the routing message and appending the signature to the message.

5. SUMMARY

This note has focused on the basic of network security, key management and secure routing protocols. In fact, there are still a lot of topics to be discussed, which include intrusion detection systems, policy management, secure localization, etc. Even more, there are several advanced problems such as secure group communication, secure time synchronization, secure MAC protocols, interplay of security and performance, trust establishment, management, privacy and so on. Thus, in summary, this field is full with open search problems and far from well addressed and studied.

REFERENCES

[1] Andrew S. Tanenbaum, Computer Networks, fourth edition. 2003 Pearson Education, Inc.

[2] Farooq Anjum, Petros Mouchtaris. Security for Wireless ad hoc Networks. 2007 John Wiley & Sons, Inc.

[3] Hongmei Deng, Dharma P. Agrawal. Security in Wireless ad hoc Networks. Handbook of Algorithms for Wireless Networking and Mobile Computing 2006 p937-959

[4] Ray Hunt, Jenne Wong. Security Architectures in Wirelss LANs. Handbook of Algorithms for Wireless Networking and Mobile Computing 2006, p905-936

[5] L. Zhou and Z. Hass, Securing Ad hoc Networks, IEEE Network, 13(6), 24-30, 1999

[6] H. Luo, J. Kong, P. Zerfos, S. Lu and L. Zhang, URSA: Ubiquitous and Robust Access Control for Mobile Ad Hoc Networks, IEEE/ATM Transactions on Networking, December 2004, 1049-1063

[7] S. Capkun, L. Buttyan, J.P. Hubaux, Self-Organized Public-Key Management for Mobile ad hoc Networks, IEEE Transactions on Mobile Computing, 2(1), 52-64(2003).

[8] S. Basagni, K. Herrin, D. Bruschi, E. Rosti, Secure Pebblenet, in Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc2001, Long Beach, CA, 4-5 October 2001, pp. 156-163.

[9] A. Perring, R. Szewczyk, V. Wen, D. Culler, J. D. Tygar, SPINS: Security Protocols for Sensor Networks and Distributed System Security Symposium, NDSS 2001, Feb 2001.

[10] S. Zhu, S. Setia, S. Jajodia. LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks, in Proceedings of the 10th ACM Conference on Computer and Communications Security(CCS'03), Washington, DC, Oct 2003.

[11] L. Eschenauer, V. Gligor, A Key-management Scheme for Distributed Sensor Networks. In proceedings of the 9th ACM Conference on Computer and Communications Security, Nov 2002, pp. 41-47.

[12] A. Halfslund, A. Tonnesen, R.B. Rotvik, J. Andersson, O. Kure, Secure Extension to the OLSR Protocol, OLSR Interop and Workshop 2004.

[13] D. Raffo, T. Clausen, C. Adjih, P. Muhlethaler, An Advanced Signature System for OLSR, SASN'04, Oct, 2004.

[14] C. Adjih, T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, D. Raffo, Securing the LSR Protocol, Proceedings of Med-Hoc-Net, June 2003

[15] P. Papadimitratos, Z. J. Haas, Secure Link State Routing for Mobile ad hoc Networks. in Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops), 2003, p.379

[16] L. Buttyan, I. Vajda, Towards Provable Security for ad hoc Routing Protocols, in Proceedings of the 2nd ACM workshop on Security of ad hoc and Sensor Networks, 2004, p94-105

[17] Y.C. Hu, A. Perrig, Davic B. Johnson. Ariadne: a Secure On-demand Routing Protocol for ad hoc Networks, in Proceedings of the ACM Conference on Mobile Computing and Networking (Mobicom), 2002, pp. 12-23.

[18] Manel Guerrero Zapata, Draft-guerrero-manet-saodv-06.txt, http://ietfreport.isoc.org/idref/draft-guerreor-manet-saodv/

[19] M. G. Zapata, N. Asokan, Securing ad hoc Routing Protocols, WiSe, Sep 2002.

[20] Y.C. Hu, D.B. Johnson, A. Perring, SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless ad hoc Networks, Ad Hoc Networks, 1(1), 175-192(2003)

[21] J. Kong, X. Hong, ANODR: Anonymous On Demand Routing with Untraceable Routes for Mobile ad hoc networks, ACM MobiHoc, June 2003.

[22] Y. Zhang, W. Liu, Y. Fang, MASK: an Anonymous Routing Protocol for Mobile ad hoc Networks, IEEE Transaction on Wireless Communications Vol 5, Issue 9, Sep 2006, p 2376-2385