LIGHT-WEIGHT SELF-ORGANIZED AUTHENTICATION MANAGEMENT FOR MOBILE AD HOC NETWORKS BY USE OF HASH CHAIN

Benny H. Tang, Caren P. Tang, Rosanna Y. Chan, Joseph K. Liu,
Department of Information Engineering
The Chinese University of Hong Kong

Duncan S. Wong
Department of Computer Science
City University of Hong Kong

Abstract: - We propose a hash-chain based authentication management protocol, which is an integrated solution to achieve (i) self-organization and (ii) low computational complexity, for MANETs. We introduce a propagated trust relation model. Under this model, we propose fully self-organized procedures of key chain generation, key chain storage, key chain renewal, key chain distribution and key chain revocation. Our protocol does not require a centralized security infrastructure, even in the initialization phase during the formation of the network.

Keywords: - Mobile ad hoc networks, self-organization, key-chains, trust-chain, propagated trust relationship

1 Introduction

Mobile Ad Hoc Networks (MANETs) face many security challenges. MANETs do not rely on any fixed infrastructure [1,2]. Everyone can join or leave the network at any time and hence, network topology in such network keeps changing dynamically. Nodes are responsible for performing all network services (e.g. routing, security), in a self-organized way [3,4,5]. Due to the lack of centralized administration, securing mobile ad hoc network is challenging.

In contrast to traditional networks, mobile ad hoc networks (MANETs) are characterized by their lack of centralized administration or control over the network principals. Entities may join or leave the network freely, resulting in a dynamically changing network topology. There exists no infrastructure through which one entity may connect some other entities reliably. In this case, the CA, responsible for the security of the entire network, would become a vulnerable point of the network. If the CA is unavailable, nodes cannot get the current public keys of other nodes or to establish secure communication with others. If the CA is compromised and leaks its private key to an adversary, the adversary can then sign any erroneous certificate using this private key to impersonate any node or to revoke any certificate. Traditional solutions that rely on a central authority to manage authentication over the network are not applicable here. For this reason, how authentication can be achieved in MANETs becomes a challenging topic and no perfect solutions have been proposed.

An authentication management protocol which is well-suited for MANETs should include the following properties: secure, self-organized, and lightweight. *Self-Organized* [6,7] is required to tackle the absence of a common security infrastructure. Nodes can perform authentication management on their own in an ad hoc environment. *Light-weight* [8] is important as most MANETs devices are battery-powered with low complexities. Expensive computations are too complex and lead to a decrease in performance.

Our Contributions

We propose a self-organized authentication management protocol with low computational complexity for MANETs. Our protocol performs authentication management in a fully self-organizing manner without requirement on the presence of a common security infrastructure. It includes:

1. A self-organized key chain generation process without any central authority or TTP.
2. A propagated trust relationship establishment process allowing a node to prove itself to a stranger.
3. Only lightweight operations are used.

In Sec. 2, we review some current authentication protocol for MANETs. In Sec. 3, we describe our protocol. Performance and security are analyzed in Sec. 4 and 5. In Sec. 6, we conclude the paper.

2 Related Work

Due to the lack of a fixed infrastructure, traditional authentication models which require the presence of CA are not well-suited for MANETs. In order to find a solution to this problem, many researches have been carried out and various models have been proposed. These models can be classified into three categories:

1. Centralized authority (CA) during the initial stage
2. Distributed authorities (DA)
3. Self-organization

2.1 CA in the initial stage

MANETs devices may not have an access to the CA at all times. Hence, traditional authentication models based on CA cannot be adopted in MANETs. The protocol in [9] extends the ZCK authentication to provide identification at the cost of external infrastructure and moderate computing power. In this protocol, one entity is able to prove its identity to another entity with the help of a trusted third party (TTP). For instance, when entity A is connected to a fixed station (e.g. a PDA connected to a PC), it can perform the traditional PKI authentication to entity B. Such identification process can be used to exchange key-chain elements which are later used in ZCK authentication. Although this protocol helps establish initial trust relationships in ZCK, it requires a TTP and a high computational power of the mobile devices. It may not be applicable in MANETs.

2.2 Distributed authorities (DA)

Instead of using a single CA, distributing the responsibilities to a set of nodes would yield a higher chance of success. These sets of nodes are known as the distributed authorities (DAs). In [2,10], a model using distributed authorities is proposed. In [2], a set of n nodes are chosen to be the DAs when the network is formed. Each of them has a public key and a secret key. Any new node joining the network uses its identity as public key. It obtains a secret key from *t*-out-of-*n* nodes. Every node within the network uses these public-secret key pair for authentication. In [10], *n* nodes are selected to share the responsibility of a CA to issue certificates. Each of them signs a partial signature on the certificates. Any *t* of the partial signatures can be used to produce a valid signature through a combiner. Although it is easier to obtain verification from the DAs, there are still some drawbacks. Firstly, the value of *t* is a trade-off between availability and robustness. There is no efficient way to determine how the value of *t* should be adjusted when the overall number of nodes increases or decreases significantly. Also, the system is vulnerable to the Sybil attack [3].

2.3 Self-organization

Self-organized authentication models do not rely on any trusted authorities or fixed servers, even in the initialization phase. This approach overcomes the problem of distributing public keys over MANETs. In [3], a self-organized key distribution is introduced. It allows users to generate their public key pairs, issue certificates, and perform authentication without referring to any centralized authorities. However, the computational cost is high and may not be suitable for low-power devices in MANETs.

3 Our Scheme

Let u_X be the t-bit secret key of X and ID_X be the identifier of X. Let $r_{X\to Y}$ be a random *t*-bit string used by X to generate X-Y key chain. Let $S_{X\to Y}$ and $N_{X\to Y}$ be the key chain and lifetime key chain, respectively, of X for communication with Y. Let $s^i_{X\to Y} \in S_{X\to Y}$, $0 \le i \le 2N_{X\to Y}$ be the i^{th} element of the key chain used by X for communication with Y. The root element of the key chain is denoted by $s^0_{X\to Y}$.

Let $d_{X\to Y}$ be the authentication distance from X to Y. Authenticated ID repository of X is denoted by AR_X. The Provable ID, Known ID and Revoked ID repositories of X are denoted by PR_X, KR_X and RR_X, respectively. Let $AL_{X\to Y}$ be the trust chain list from X to Y. Define $A(AR_X)$ to be a function that generates a 2-tuple $(ID_Y, 0)$ for each neighbor ID in AR_X, where Y is the neighbor node and 0 stands for the zero-authentication distance. Define $P(PR_X)$ to be a function that generates a 2-tuple (ID_Y, d_Y) for each $AL_{X\to Y}$ in PR_X, where Y is the remote node and d_Y is the minimum authentication distance. Let h be a hash function and f be a function which maps to *t*-bit strings. Let $(m)_s$ be the MAC of the message m by the key s.

Repositories AR_X is the collection of IDs of nodes to which X has established a trust relationship. These nodes are called the **neighbor nodes** and their IDs are referred to as neighbor IDs. PR_X is a collection of IDs to which a trust chain can be set up from X. These nodes are called **remote nodes** and their IDs are referred to as remote IDs. KR_X is the collection of IDs that X knows. In other words, $KR_X = P(PR_X) \cup A(AR_X)$. RR_X is the collection of IDs with which the trust relationship has been revoked by X yet the revocation has not been expired.

Entities in our protocol Let Alice, Bob, Cathy, Xen and Zita are normal nodes. Suppose Xen is a neighbor of Alice and Zita is a neighbor of Cathy. Let Mandy is an adversary.

Services of our protocol

1. Key Chain Generation Any two nodes generate and exchange the last element of hash chains for future authentication. The received elements are stored in their Authenticated ID repository.

2. Authentication A node can prove itself to another node.

3. Key Chain Renewal A node generates a new hash chain and sends the last element of the chain to its neighbor. The neighbor updates the received element to its Authenticated ID repository.

4. Trust Relationship Revocation A node removes the trust relationship of a neighbour by deleting the ID of the node from its Authenticated ID repository and moves that ID to its Revoked ID repository. The trust relationship between these two nodes is no longer valid.

5. Known ID Repository Exchange Two nodes with trust relationship create their ID repositories and exchange them.

6. Propagated Trust Relationship Establishment A node proves itself to a stranger through a trust chain. Then the entities perform key chain generation to create a trust relationship between them. This service has a sub-routine, Trust Chain Activation: the trust chain between two nodes without trust relationship is activated to verify from the neighbours, in a recursive manner, whether the trust relationship can finally propagate to the target entity.

3.1 Trust Chain Activation

Scenario: Suppose $AL_{A\to C}$ exists in PR_A and Alice wants to activate a trust chain to Cathy. Note that all helper nodes have ID_C in their known ID repositories. Let X_i be the node in the i^th position of the trust chain from Alice to Cathy, where $i \ge 0$. Alice is denoted as X_0.

Steps:

1. Alice(X_0) extracts the neighbour ID from $AL_{A\to C}$ which corresponds to a trust chain with the minimum authentication distance. Let X_1 be the chosen neighbour node. She then authenticates herself to X_1 and sends $(ID_A, ID_C, d_{X\to A})$ to X_1, where $d_{X_0\to A} = d_{A\to A} = -1$. Set i = 1
2. X_i sets $d_{X_i\to A} = d_{X_i\to A}+1$ and checks if $ID_A = ID_{X_{i-1}}$. If so, go to Step 3. If not, X_i checks if $AL_{X_i\to A}$ exists in PR_{X_i}. If so, X_i updates $AL_{X_i\to A}$ with $(ID_{X_{i-1}}, \mathrm{d}_{X_i\to A})$. If not, X_i adds $AL_{X_i\to A} = \{ID_A: (ID_{X_{i-1}}, \mathrm{d}_{X_i\to A})\}$ to PR_{X_i}..
3. X_i checks if $ID_C = ID_{X_i}$. If so (i.e. Cathy is notified of the trust chain), the process is finished. If not, X_i checks if ID_C exists in AR_{X_i}. If so, let X_{i+1} be Cathy. If not, X_i extracts the neighbor ID from $AL_{X_i\to C}$ which corresponds to a trust chain with the minimum authentication distance. Let X_{i+1} be the chosen neighbor node.
4. X_i authenticates itself to X_{i+1} and sends $(ID_A, ID_C, d_{X_i\to A})$ to X_{i+1}. If any failure occurs, X_i checks if other neighbor IDs exists in $AL_{X_i\to C}$. If so, X_i extracts another neighbor ID from $AL_{X_i\to C}$ stored with the next minimum authentication distance. Let X_{i+1} be this node and try again from Step 4. If not, for $j = i$ to 1, X_j authenticates to X_{j-1} and reports to X_{j-1} about the trust chain activation failure and the process is terminated.
5. Set $i = i + 1$ and repeat from Step 2 again.

3.2 Propagated Trust Relationship Establishment

Scenario: Cathy wants to establish a propagated trust relationship with Alice.

Steps:

1. Cathy sends her own ID, i.e. ID_C, to Alice.
2. Alice checks if $AL_{A\to C}$ exists in PR_A. If exists, Alice activates a trust chain to Cathy. If the activation is successful, Alice asks Xen to get ID_Z and the last opened key of $S_{C\to Z}$. If Alice receives $(ID_Z,\ s^{2q}_{C\to Z})$ within the timeout period, go to Step 4. Otherwise, Alice asks Cathy to activate a trust chain to her. If the trust chain activation fails, Alice asks Cathy to activate a trust chain to her. If not exist, Alice asks Cathy to activate a trust chain to her.
3. Cathy checks if $AL_{C\to A}$ exists in PR_C. If exists, Cathy activates a trust chain to Alice. If the activation is successful, Cathy asks Zita to deliver its ID, i.e. ID_Z, and the last opened key of $S_{C\to Z}$ to Alice. If Alice receives $(ID_Z,\ s^{2q}_{C\to Z})$ within the timeout period, go to Step 4. Otherwise, she halts. If the trust chain activation fails, the process is terminated.
4. Alice asks Cathy for $s^{2q-1}_{C\to Z}$ of $S_{C\to Z}$. Cathy sends $s^{2q-1}_{C\to Z}$ to Alice. Alice checks if $h(s^{2q-1}_{C\to Z}) = s^{2q}_{C\to Z}$. If so, Alice asks X to deliver its ID, i.e. ID_X, and the last opened key of $S_{A\to X}$ to Cathy. Otherwise, she halts.
5. If Cathy receives $(ID_X,\ s^{2p}_{A\to X})$ within timeout period, Cathy asks Alice for $s^{2p-1}_{A\to X}$ of $S_{A\to X}$. Otherwise, the process is terminated. Alice sends $s^{2p-1}_{A\to X}$ to Cathy. Cathy checks if $h(s^{2p-1}_{A\to X}) = s^{2p}_{A\to X}$. If not, the process is terminated. Cathy performs an additional process with Zita: Cathy sends $s^{2q-2}_{C\to Z}$ to Zita and suspends the process until she has sent $s^{2q-3}_{C\to Z}$ to Alice. Cathy then resumes the process and sends $s^{2q-4}_{C\to Z}$ to Zita.
6. Alice checks if $h^2(s^{2q-3}_{C\to Z}) = s^{2q-1}_{C\to Z}$. If so, Alice performs the key chain generation process with Cathy. Otherwise, the process is terminated.

4 Performance Analysis

4.1 Storage Requirements:

Assume t=80 which provides a comparable security to 1024-bit RSA. Private key takes 10 bytes, the 5-tuple takes 28 bytes, one entry in Provable ID repository (assume maximum five elements/AL) takes 35 bytes, and one entry in Revoked ID repository takes 7 bytes.

4.2 Computation Requirements:

Consider most significant component (hash operation).

Service\Operations	# hashes
Key Chain Generation	2N
Authentication	2(q+k’)
Key Chain Renewal	2(q+k’+N’)
Known ID Repository Exchange	2(q+k’)
Trust Relationship Revocation	0
Propagated Trust Relationship	2(q+N)+3
Trust Chain Activation	2(q+k’-1)

5 Conclusion

We introduce a secure and self-organized yet lightweight authentication management scheme for mobile ad hoc networks. Two critical issues have been handled, namely the decentralization of administration and the efficiency requirement. Our protocol works in a fully self-organized manner under the propagated trust relation model. No centralized security infrastructure is needed in none of our services. Only hash chain but no expensive operations like RSA is used. Because of the one-wayness of hash chain, our protocol maintains a high security level with low computation power consumption. It provides a secure, self-organize and light-weight authentication management. All these virtues enable our authentication management scheme to be highly suitable for the mobile ad hoc networks.

References

[1] S. Zhu, S. Xu, S. Setia and S. Jajodia, “LHAP: A Lightweight Hop-by-Hop Authentication Protocol For Ad-Hoc Networks”, *In Proc. of ICDCS 2003 International Workshop on Mobile and Wireless Network (MWN 2003)*, May 2003

[2] A. Kahlili, J. Katz, W. A. Arbaugh, “Toward Secure Key Distribution in Truly Ad-Hoc Network”, *IEEE Workshop on Security and Assurance in Ad-Hoc Networks 2003.*

[3] S. Capkun, L. Buttyan and J. Hubaux, “Self-Organized Public-Key Management for Mobile Ad Hoc Networks”, *Proc. IEEE Transactions on Mobile Computing, Vol. 2, No. 1, January-March 2003.*

[4] H. Yang, X. Meng and S. Lu, “Self-organized network-layer security in mobile ad hoc networks”, *Proc. of the ACM workshop on Wireless security2002 (WiSE ’02).*

[5] J. Hubaux, L. Buttyan and S. Capkun, "The Quest for Security in Mobile Ad Hoc Networks", *MobiHOC ’01.*

[6] Klaus Herrmann, “A Middleware for Self-Organization in Ad hoc Networks”, *23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03).*

[7] L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J. Hubaux, and J. LeBoudec, “Self-organization in mobile ad-hoc networks: the approach of terminodes”, *IEEE Comm. Magazine*, 2001.

[8] Mathias Bohge and Wade Trappe, “An Authentication Framework for Hierarchical Ad Hoc Sensor Networks”, *ACM Workshop on Wireless Security (WiSe ‘02)*

[9] A. Weimerskirch and D. Westhoff, “Identity Certified Authentication for Ad-hoc Networks”, *2003 ACM Workshop on Security of Ad Hoc and Sensor Networks.*

[10] Zhou and Z. J. Haas, “Securing Ad Hoc Networks”, *IEEE Network Magazine*, vol. 13, no.6, Nov/Dec 1999