A Secure On-Demand Routing with Distributed Authentication for Trust-Based Ad Hoc Networks

Meng-Yen Hsieh[1,2] and Yueh-Min Huang[1]

[1] Department of Engineering Science, National Cheng-Kung University, Taiwan
{n9892111, huang}@mail.ncku.edu.tw
http://www.es.ncku.edu.tw/index.htm
[2] Department of Information Science, Hisng-Kuo University of Management, Taiwan
tab.hsieh@mail.hku.edu.tw
http://www.hku.edu.tw/index.htm

Abstract. Due to node mobility, the ad hoc network topology is dynamical so that on-demand routing protocols are more fit than other routing protocols. Most secure on-demand routing protocols are designed that the destination or source is able to detect the attacks on routing paths after accepting routing requests or routing replies. In this paper, we present a secure on-demand source routing protocol without the assumption of a specific cryptographic system provides per-hop broadcast authentication in routing discovery phase and security in communication phase and takes effect on our trust-based ad hoc environment. Our hop by hop broadcast authentication provides forwarding routing packets with their trust levels for abstaining from unreliable or malicious nodes. Through security analysis and discussion, we characterize our mechanism and show that it is effectively and efficiently.

1 Introduction

According to the most secure routing protocols, control packets are able to be authenticated by Source or Destination. For battlefield applications, mobile nodes communicate each other in hostile environments so that we design a secure routing acting on unreliable ad hoc environments. A number of contributions are presented in this paper: First, we construct trust controls in the network so that routing messages are flooding to nodes with certain trust requirements. Second, without any assumption of cryptographic infrastructure, we applied few system asymmetric and symmetric keys for the security and privacy of the ad hoc routing. Third, in the trust-based network, we propose per-hop broadcast authentication during per forwarding.

In our scheme, system asymmetric keys are assigned to different trust levels. However, system private keys are divided into shares hold by ad hoc nodes. A node belongs to a trust level when its certificate signed by the certain private key. With the same level certificate, these nodes are in a common trust-level community. Then any two neighbors of them will share commitments of their one-way hash key chains each other. We also present a secure on-demand routing protocol (SODR) with distributed authentication including two phases, routing discovery phase and communication

phase. The routing discovery can construct temporal session keys in each routing to protect data traffic since a key agreement scheme is applied into our routing discovery. Through session keys, end-to-end communications are encrypted and authenticated multiply. By using one-way hash key chain, routing packets are authenticated hop by hop. And, the SODR adopts one-way hash and message authentication code (MAC) to achieve the integrity of routing packets.

The remainder of this paper is organized as following. In section 2, related techniques in the SODR are described. And we give an overview of recent secure routing protocols. Section 3 describes our trust-based system design in ad hoc networks. Section 4 details the SODR with distributed authentication. In section 5, we give the analysis of SODR defending against attacks under various attack models. And we discuss related problems about one-hop broadcasting authentication and detection of un-trustable nodes. Finally section 6 offers concluding remarks.

2 Background

Ad hoc routing protocols have two categories. One is table-driven protocol. Another is on-demand protocol such as AODV[2] or DSR[1]. Due to network topology, on-demand routing methods are more fit. In AODV, attackers easily damage routing by compromising participants. Hence, the paper considers the DSR instead of the AODV.

Diffie-Hellman Key Exchange Protocol. By using the Diffie-Hellman (D.H.) key agreement protocol proposed in [3], two nodes generate their random private values Xa and Xb to drive their public values with two system parameters, so-called g and q. The q is a prime number and the g is an integer less than the q, with the property: for every number n between 1 and p-1 inclusive, there is a power x of g so that $Y = g^x \bmod q$. Two nodes derive their public values using parameters g and q and their private values x, then exchange their public values. Since $k = (g^{Xb})^{Xa} = (g^{Xa})^{Xb}$, two nodes have a common secret key k.

One Way Trapdoor Function [4]. A one-way function with a "trapdoor" is provided with a key that makes it easy to invert the function. A feasible approach of ad hoc networks in an on-demand routing is applied to an asymmetric cryptosystem or symmetric cryptosystem with a one-way trapdoor function. In an asymmetric cryptosystem, the encryption/verification function f^{-1} uses a public key(pk) such as $y = f^{-1}(pk,x)$, while the decryption/signing function f uses a private key(sk) such as $x = f(sk, y)$. In a symmetric cryptosystem, the one way function f is applied to both encryption and decryption with a common secret key(k) such as $y = f(k,x)$ and $x = f(k,y)$.

Secure Ad Hoc Routing. Most researches on the security design of routing protocol consider end-to-end authentication. Some papers [5,8] provide authentication for validating intermediate nodes. Another paper [6] considers privacy in routing discovery by hiding route information with encryption schemes or distributed route information over participant nodes. Recent papers use asymmetric cryptographies, for example [5,6]. And some papers [7,8,9] adopt symmetric cryptographies. A paper [8] uses TELSA key to achieve authentication.

Authentication routing for ad hoc networks (ARAN) proposed by [5] is a rigorous authentication protocol. Through the assumption of public key cryptosystem, it de-

feats malicious attacks with third parties and peers as modification or fabrication of routing messages or impersonation of valid nodes. By guaranteeing per-hop authentication achieving non-repudiation services with cryptographic certificates, the approach allows a victim selection of routes and denial–of–service attacks.

Features of the secure routing protocol (SRP) [9] are verifiable routing queries and replies, the binding of secure routing and network layer functionality, the partial acceptance of route error messages, a dual identifier in query or reply packets, and the regulation of the query propagation. SRP either rejects or prevents fabricated, compromised, or replayed route replies from the achievement of sending back the source with the only requirement of a priori a shared secret between any two communication nodes in place of any assumption regarding intermediate nodes or cryptosystem.

3 Trust-Based System Design

In this section, a trust-based infrastructure is designed without any specific cryptographic system. Three types of keys are used, as few system asymmetric keys for signature with trust levels, a system secret key for requesting a share, and one-way hash key chains for one-hop broadcasting authentication from a node to its neighbor nodes.

3.1 Trust Level Design

We generate few system public and private keys, so-called PK/SK, matching to the amount of trust levels. A trust level represents a certification signed with a system private key, and a matching system public key can verify the correctness of the certificate. System keys are divided into different trust levels according to the difficulty of obtaining system private keys and adopts a (t,n) threshold scheme. Each SK is divided into n shares and distributed over the network. Collecting t shares can return the key. System asymmetric keys with different trust levels have different t values.

Let q be a large prime and $GF(q)$ be a finite field. The system chooses a polynomial $f_i(x)$ of degree at most t-1, where t < n. Suppose that $f_i(x) = a_{i,t-1}x^{t-1}+\ldots+a_{i,1}x+a_{i,0} \bmod q$, where coefficients $a_{i,j}\varepsilon GF(q)$ are chosen at random for j = 1,…t-1. A system private key, SK_i, is decomposed to the shares $SSK_{i,j} = f_i(j)$ communicated to ad hoc nodes. Since each node enters the network, it holds a share of each system private key and holds a sequence of P elements ($SSK_{1,k}$, …, $SSK_{P,k}$), where P is the number of trust levels. Asymmetric system keys have threshold values as (t_1, …, t_P), where $t_i\varepsilon Z^*$, $t_i < n$, and $t_{i-1} < t_i$. Besides a share of each SK, each node holds all PK for checking the validity of certificates from other nodes, as (PK_1, …, PK_P). A node belongs to the i^{th} trust level since it has the certificate signed with the key, SK_i. For gaining the key, the node collects shares from other nodes up to the threshold value, t_i. For the following example, node u gets a share with the i^{th} trust level from a neighbor node v.

u → *: REQ_{TLi}, $Nonce_u$, where REQ_{TLi} is a request for a SSK_i.

v→ u: $E_{GK}(SSK_{i,u}, Nonce_u)$, where GK is the system secret key.

After enough collecting, it computes the private key ($SK_i = \Sigma^{t}_{j=1} SSK_{i,j} \bmod q$) to sign its certificate, so-called $CCert_{TL}$. An extended filed of a certificate records a trust level (TL) value. A $CCert_{TL}$ is used restrictedly in a period time such as (the expiration time (RET) – the issue time (ISST)) < Max_Used_Time (*MUT*). Certificates signed with different *SK* have different *MUT* value. Max_Used_Time of P kinds of certificates are (MUT_1, MUT_2, …, MUT_P), where $MUT_{i-1} < MUT_i$.

3.2 One-Hop Broadcast Authentication

For one-hop broadcasting authentication, our one-way hash key chain is different from TESLA or uTELSA. Each node generates a key hash chain of an appropriate length according to past change rate of neighbor nodes. By repeatedly computing with a hash function *hash*, the key chain values are: <TK_0, …, TK_N> since TK_{i-1} = *hash*(TK_i), where the TK_0 is the commitment key. A node needs to announce its current trust level with its certificate ($CCert_{TL}$) by periodically broadcasting its HELLO message to neighbor nodes. The format of HELLO is <HELLO, $CCert_{TL}$>. If a node enters the network just now, a pure HELLO represents that the node is in trust level 0 and without any signed certificate. By listening for HELLO messages from neighbor nodes, a node gains their certificates and verifies their trust levels by corresponding *PK*. If they have the same TL certificate with it, the node encrypts with the *PK* to forwards a commitment to them. A encrypted message with PK is <E_{PK}(TK_0, Nonce), $CCert_{TL}$>.

Keys from a one-way key chain are used for one-hop broadcasting authentication per RREQ forwarding. Whenever a node floods a RREQ, it appends a MAC with the next key, TK_{next}, of its key chain since the key is disclosed in a reverse order of its key chain generation. The TK_{next} is disclosed immediately and appended to the RREQ since it is only effective in one-hop distance. The format of a RREQ from a node to neighbor nodes is <RREQ', MAC(TK_{next}, RREQ'), TK_{next}>. The MAC of a RREQ is authenticated by the instantaneous disclosed key since (TK_0=*hash*(..*hash*(TK_{next})..)).

4 Secure On-Demand Routing With Distributed Authentication

A routing discovery phase has two steps between source (Sour) and destination (Dest): the path discovery step and the path reverse step. In the discovery step, a routing request (RREQ') packet is addressed: <RREQ, Sour, Qid, Tdoor, NList, PVList, HChain, MList>. The node list (NList) represents intermediate nodes whose trust levels are same with Sour and Dest. The public values list (PVList) is a set of D.H. public values of intermediate nodes. The MList is a set of MAC values of intermediate nodes. The Hash chain value (HChain) is multiply hashing by per-hop intermediate node, and the field is equal to: $hash_i$[…,[$hash_1$[a init-hash value]…]], where the intermediate nodes are from 1 to i. In the reverse step, a routing reply (RREP') is addressed: <RREP, Sour, Td_Proof, NList, PVList, MList, HAKList>. For adopting the trapdoor scheme, this paper assumes that each node shares an encryption key with each of its recipients to communicate with. Only a recipient is able to accept and

decrypt a packet through its trapdoor, then gain a proof and a hiding public value of D.H.

A trapdoor (Tdoor) is constructed by Sour. According the description of trapdoor, a trapdoor is implemented with asymmetric key or symmetric key. Initially Sour just knows the certifiable public key PK_D of Dest. The Tdoor format in RREQ is: Tdoor = [Dest, Tstamp, $K_{S,D,}$ PWD]$_{PK_D}$, PWD(Dest). The trapdoor is only opened by Dest and the random $K_{S,D}$ selected by Sour will be used for next route request as a shared symmetric key. The all later Tdoor format is: [Dest, Tstamp, PWD]$_{K_{S,D}}$, PWD(Dest). In the network, only Dest can see the destination tag Dest and conclude it is the intended destination. The random PWD is a secret during the discovery step. However it is be exposed during the reverse step. The proof of trapdoor format in RREP is: Td_proof = PWD, PWD(Dest'). By comparing PWD(Dest) = PWD(Dest'), any forwarding node can verify the proof of a trapdoor opening.

PATH DISCOVERY STEP (S→A→B→D):
S: Tdoor = [Dest,Tstamp, $K_{S,D}$,PWD]$_{PK_D}$,PWD(Dest) or
[Dest,Tstam,$K_{S,D}$,PWD]$_{K_{S,D}}$,PWD(Dest).
S: h0 = MAC_{PK_D}(RREQ, S, Qid, Tdoor, PWD(Dest), g^s) or
$MAC_{K_{S,D}}$(RREQ, S, Qid, Tdoor, PWD(Dest), g^s).
S: RREQ' = <RREQ, S, Qid, Tdoor, PWD(Dest), g^s, h0>
S→*: < RREQ', MAC(TK_S, RREQ'), TK_S>
A: Verify RREQ' by computing TK_0 = hash[hash…[TK_S]..]; Store PWD(Dest); Compute $g^{s,a}$
A: h_A=hash (h0, A, g^a); M_A=$MAC_{g^{s,a}}$(RREQ, S, Qid, Tdoor, A, g^s, g^a, h_A);
A: RREQ'=<RREQ, S, Qid, Tdoor, A, g^s, g^a, h1, M_A>
A→*: <RREQ', MAC(TK_A, RREQ'), TK_A>
B: Verify RREQ' by computing TK_0 = hash[hash…[TK_A]..]; Store PWD(Dest); Compute $g^{s,b}$
B: h_B=Hash (h1, B, g^b); M_B=$MAC_{g^{s,b}}$(RREQ, S, Qid, Tdoor, A, B, g^s, g^a, g^b, h_B)
B: RREQ'=<RREQ,S,D,Qid,Seq,Tdoor,A,B,g^s,g^a,g^b,h_B,M_A,M_B>
B→D: <RREQ', MAC(TK_B, RREQ'), TK_B>

Fig. 1. An example of a path discovery step

PATH REVERSE STEP: (D→B→A→ S)
D: Td_Proof = PWD, PWD(Dest')
D: M_D = $MAC_g^{s,b}$(RREP, S, Td_Proof, A, B, M_A, M_B)
D: RREP' = <RREP, S, Td_Proof, A, B, g^d, g^a, g^b, M_A, M_B, M_D, h($g^{d,a}$), h($g^{d,b}$), h($g^{s,d}$)>.
D→B: <RREP', MAC(TK_D, RREP'), TK_D>
B: Verify M_B, PWD(Dest'); Compute $g^{d,b}$ and check h($g^{d,b}$); Replace h($g^{d,b}$) with h($g^{s,b}$)
B: RREP' = <RREP, S, Td_Proof, A, B, g^d, g^a, g^b, M_A, M_B, M_D, h($g^{d,a}$), h($g^{s,b}$), h($g^{s,d}$)>.
B→A: <RREP', MAC(TK_D, RREP'), TK_B>
A: Verify M_A, PWD(Dest'), Compute $g^{d,a}$ and check h($g^{d,a}$); Replace h($g^{d,a}$) with h($g^{s,a}$)
A: RREP' = <RREP, S, Td_Proof, A, B, g^d, g^a, g^b, M_A, M_B, M_D, h($g^{s,a}$), h($g^{s,b}$), h($g^{s,d}$)>.
A→S: <RREP', MAC(TK_A, RREP'), TK_A>

Fig. 2. An example of a path reverse step

The example, as Fig. 1, describes our routing discovery phase, where S discoveries route to D (S→A→B→D). Node A and B are intermediate nodes, participating the route. First, S sets an initial RREQ appended with its id, Query ID (Qid), Tdoor, a

public value of S (g^S), and an initial-hash value (h0). At the first routing, h0 is constructed with public key of D. However, in the later routing, h0 is encrypted with a pairwise key shared between S and D. Since S creates a RREQ' for its one-hop broadcasting authentication in its trust level. A TK_S is disclosed in an order reverse to the S key chain. If A, receiving the message can verify its authenticity based on the commitment or a recently disclosed TK key of S. Before forwarding a route request, A adds itself and related information to the NList, PVList, and MList fields of RREQ. It replaces the hash chain field by hashing with Hash($h_{previous}$, A, g^A). A needs to generates a temporal key gs,a for this session with a D.H. public value g^A. The key $g^{s,a}$ mapping to the public value g^a is stored in its memory. After storing PWD(Dest) and appending a MAC to the MAC list, the A finally rebroadcasts the modified RREQ' to one-hop neighbor nodes. Like operations of the A, the B processes and discard/rebroadcast it. Finally, the RREQ' reaches the Dest D.

After accepting the RREQ, D opens the trapdoor to get a PWD and a symmetric key ($K_{S,D}$) and other information. The PWD is a proof of opening the trapdoor and the $K_{S,D}$ is for next communication session. D checks intermediate nodes by verifying the hash chain value of the RREQ. Then, D makes a set of temporal session keys, ($g^{d,a}$, $g^{d,b}$, $g^{s,d}$), which will be shared with intermediate nodes and S. When a reverse step starts, D makes a RREP consisting of some parts of the RREQ, a Td_proof, a MAC code M_D of the entire RREP. In addition, D insets its public value $g^{d,}$ and hashed session keys into the HAKList field of RREP, ($h(g^{d,a})$, $h(g^{d,b})$, $h(g^{s,d})$). In Fig. 2, we give the detail about a reverse path from D to S. Along a reverse routing path, a RREP' is also applied into broadcast key for one-hop authentication. When B receives the RREP', then check if the packet is from D with the current trust level. B checks PWD(Dest) = PWD(Dest') to know the trapdoor is opened. B gains a session key $g^{d,b}$ shared with D after checking the validity of $h(g^{d,b})$ and replaces $h(g^{d,b})$ with $h(g^{s,b})$. Then B rebroadcast the modified RREP'. All intermediate nodes forwarding the RREP' in the same trust level will gains session keys shared with S or D. Finally the RREP' reaches S. S generates session keys ($g^{s,a}$, $g^{s,b}$, $g^{s,d}$) shared with all intermediate node and D, then authenticates hashed session keys of the RREP'. And S verifies these codes of MList (M_A, M_B, M_D) to verify the correctness of each-hop forwarding.

$S \rightarrow A \rightarrow B \rightarrow D$: A, g^a, $E_g^{s,a}$ (B, g^b, $E_g^{s,b}$(D, g^d, $E_g^{s,d}$ (Msg)))
$D \rightarrow B \rightarrow A \rightarrow S$: B, g^b, $E_g^{d,b}$(A, g^a, $E_g^{d,a}$ (S, g^S, $Eg^{s,d}$ (Msg)))

Fig. 3. Communication phase between S and D

After finishing routing discovery phase, S or D shares session keys with all intermediate nodes so that multiply encryption and authentication are used for data transport between them. Fig. 3 is an example of communication of S and D. From S to D, S transmits messages with multi-layer encryption. A or B receiving the messages will strip one layer encryption with its session key corresponding to the D.H. public value.

5 Analysis and Discussion

5.1 Security Analysis

The SODR with distributed authentication can prevent external and internal adversary attacks. We give few scenarios to describe how our protocol is secure and authentic against active and passive attacks. Scenario1: a malicious node can not modify the route request RREQ since per-hash method guarantees the integrity of NList information and MAC codes of the MList of RREQ provide the integrity of per hop routing request during routing discovery phase. Scenario2: Since a trapdoor is designed by source, only destination can open it. When a proof of trapdoor is back to source, source knows RREQ has reached the target. The trapdoor is not been reused by adversary since timestamp is inside the trapdoor. Scenario3: When a malicious node without a certain trust level of source, it does not insert any packets to hurt the routing. The malicious node without certain Cert can not distribute its commitment to neighbor nodes. This scheme enhances the security, since a malicious node is difficult to participate in routing protocol. Scenario4: After receiving a RREQ, a node is able to verify the RREQ authenticity through confirming the disclosed key. It means the RREQ is passed by neighbor nodes owning certain trust level. Scenario5: Temporal session keys are to protect transferring messages such that the transport with multiply encryption proves to be resilient against path hijacking [6].

5.2 Challenge Scheme and a problem of one-hop broadcasting authentication

We assume four possible examples that a node can be challenged by neighbor nodes since it has misbehavior. In the first example, a node does not broadcast periodically HELLO messages with its certificate. In the second example, a node has been in the network for a long time, but it does not have a certificate in a reasonable trust level. In the third example, a node announces out-of-date certificates. Or a node steals and uses certificates of other nodes. In the fourth example, a node challenges the authenticity of other nodes with a certain probability. For solving four problems, two kinds of challenges are provided as the following. The first kind challenge is for a malicious node:

$X \rightarrow M$: C1, N_X, MAC(GK, C | N_X); $M \rightarrow X$: N_M, MAC(GK, C | N_X | N_M)

The second kind challenge is for a specific node holding a wrong certificate:

$X \rightarrow M$: C2, $E_{PKi}(N_X, CCert_X)$; $M \rightarrow X$: N_X, MAC(GK, N_X)

The one-way hash key chain for one-hop broadcast authentication does not demand loose time synchronization and delay key disclosure. A nod transmits the commitment to these nodes in a same trust level, encrypted with the matching system key. Due to the triangle inequality theorem, when a node floods a packet containing a message and a one-way hash key, its neighbor nodes will accepts the packet before a re-forwarded copy from a malicious node, as Fig. 4. The malicious node cannot reuse disclosed keys since these keys are effective in one-hop distance.

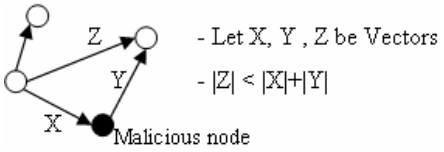

Fig. 4. The problem of one-hop broadcasting authentication

6 Conclusions

In this paper, a secure on-demand routing (SODR) is provided in trust-based ad hoc network. The SODR is with distributed authentication such as routing packets act on broadcasting authentication per flooding. Our design has these proprieties: (a) each node stores few system asymmetric keys and a system secret key. Temporal session keys are used for the confidentiality of end-to-end communication. (b) SODR provides the trapdoor method for end-to-end authentication since the trapdoor can be implemented with asymmetric or symmetric keys. (c) Through per-hop broadcasting authentication, SODR avoids that non-trusted nodes attend the routing communication. The control packets are forwarded only by these nodes with the same trust level. (d) SODR adopts per-hop hash and MAC schemes to achieve the routing integrity.

References

1. D. B. Johnson, etc.: The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-09.txt, Apr. 2003.
2. C. E. Perkins and E. M. Royer: Ad hoc on-demand distance vector routing. In Proc. WMCSA, New Orleans, LA, Feb. 1999, pp. 90–100.
3. W. Diffie and M. Hellmann: New Directions in Cryptography. IEEE Transactions on Information Theory IT, vol. 22, no. 6, pp. 644 – 654, 1976.
4. A. C.-C. Yao.: Theory and Applications of Trapdoor Functions (Extended Abstract). In Symposium on Foundations of Computer Science (FOCS), pp. 80–91, 1982.
5. Sanzgiri, K., LaFlamme, D., Dahill, B., Levine, B.N., Shields, C.; Belding-Royer, E.M.: Authenticated routing for ad hoc networks. Selected Areas in Communications, IEEE Journal on Vol 23, March 2005 pp:598 – 610.
6. Boukerche, A., El-Khatib, K., Li Xu, Korba, L.: SDAR: a secure distributed anonymous routing protocol for wireless and mobile ad hoc networks. Local Computer Networks, 2004. 29th Annual IEEE International Conference on Nov. 2004 pp:618 - 624
7. Ting-Yao Jiang; Qing-Hua Li: A secure routing protocol for mobile ad-hoc networks. Machine Learning and Cybernetics, In Proc. of 2004 International Conference on Vol 5, Aug. 2004 pp:2825 – 2829.
8. YC Hu, A. Perrig and DB Johnson: Ariadne: A Secure On-Demand Routing Protocol for Ad hoc Networks. in Proc. of MobiCom 2002.
9. P. Papadimitratos and Z. J. Haas: Secure Routing for Mobile Ad hoc Networks. in Proc. of SCS Communication Networks and Distributed Systems Modeling and Simulation Conference, Jan. 2002.