UGSP: AUTHENTICATION BASED SECURE PROTOCOL FOR AD-HOC NETWORKS

Neelima Arora
Tata Institute of Fundamental Research
Mumbai, India

R. K. Shyamasundar
Tata Institute of Fundamental Research
Mumbai, India

Keywords: Ad-hoc networks, key distribution, security

Abstract: A wireless ad-hoc network is a collection of mobile nodes with no fixed infrastructure. Security in such networks poses serious challenges due to (i) the network connectivity could be intermittent and hence on-line authentication is not guaranteed, and (ii) susceptible to wide range of attacks due to broadcast communication and large scale number of users. In this paper, we propose a security protocol, called UGSP, for wireless ad-hoc networks using a tamper-proof hardware. We show that the proposed protocol fits well with the resurrecting duckling security paradigm (Stajano and Anderson, 1999). Once the hardware is imprinted for authentication, UGSP is robust to man-in-the-middle attack, passive eavesdropping, active impersonation attacks ensuring source authentication, data confidentiality and data integrity for communication amongst nodes with identically configured hardware. The system is amenable to dynamic addition of new members whose hardware has also been imprinted with authentication information. We provide a comparative evaluation of UGSP with other approaches and show that UGSP is scalable and cost-effective.

1 INTRODUCTION

Ad-hoc networks do not have fixed infrastructure such as base station or mobile switching centers. Mobile nodes, which are within the range of each other, communicate directly while those that are far apart rely on other nodes to forward messages as routers. Node mobility in the network causes frequent changes of the network topology.
The salient features of ad-hoc networks pose both challenges and opportunities in achieving security goals characterized by attributes like availability, confidentiality, integrity, source authentication and non-repudiation (Zhou and Haas, 1999). Nodes, roaming in hostile environment (e.g., a battlefield) with relatively poor physical protection, have non-negligible probability of being compromised. Therefore, we should not only consider malicious attacks from outside a network, but also take into account the attacks launched from within the network by compromised nodes. We envision ad-hoc networks to be formed by nodes without any prior contact, trust or authority relation. This precludes any pre-distributed symmetric keys or a reliable (external) PKI supported across all nodes. This issue has been largely ignored until now; most protocols assume that the key-distribution has already taken place. Further, security mechanisms should be scalable to handle large networks.

1.1 Current State of Ad-Hoc Network Security

One of the largely investigated areas of ad-hoc network security research is devoted to secure routing protocols (Toh, 2001)(Royer and Toh, 1999), that form an essential component of security in ad-hoc networks (Khaili and Arbaugh, 2002). However, most of the routing schemes known neglect the crucial challenge in ad-hoc security: key establishment and key distribution. Protocols such as ARAN, Ariadne(Hu and Perrig, 2002), SPINS(Perrig et al., 2002b), TESLA(Perrig et al., 2002a), SEAD (Hu et al., 2002) and SRP (Papadimitratos and Haas, 2002) all assume the pre-existence and pre-sharing of secret and/or public keys for all the nodes. In other words, key management and key distribution in an ad-hoc networks has been left a wide open problem. Recently some approaches for key distribution in ad-hoc networks have been proposed (Zhou and Haas, 1999; Bobba et al., 2002; Khalili and Arbaugh, 2003). Note that a mechanism is needed wherein we can accommodate the new trust scenarios in ad-hoc networks.

1.2 Our Approach

In this paper we assume that both active and passive attackers are present in the environment. The attacker is allowed to watch regular runs of the protocol between the two communicating nodes and also send arbitrary messages to the parties. We also assume that the attacker has sufficiently large computational power.
In this paper, we describe a protocol called UGSP (User Group Security Protocol) for communication amongst a dynamic user group (DUG), that is resilient against the above mentioned attacks while maintaining all the necessary attributes (confidentiality, integrity and authenticity). Even if the node gets compromised, it should not allow the attacker to gain access to any useful secret of the network. We see examples of DUG in our day to day life such as employees of a company or all the mobile cell-phone users of a particular network.
Rest of the paper is organized as follows: section 2 gives the system architecture required for the protocol, section 3 details the protocol, analysis of the protocol is done in section 4, section 5 gives the implementation details and section 6 contains the conclusion, future work and generalizations.

2 SYSTEM ARCHITECTURE FOR THE UGSP

To make UGSP as generic as possible, we have designed the protocol minimizing the energy cost. Most previous work on secure ad-hoc network relies on asymmetric cryptography for establishing the security parameters every time. However, computing such signatures on resource-constrained nodes is expensive and hence may not be the ideal solution. A protocol with shared secret is the most generic option, as it is not expensive both in terms of bandwidth and computation.
The system has the following three components:
(1) Communication Link: Our system does not assume any special characteristic of the communication link and will work over any form of communication system such as ethernet, bluetooth, 802.11, optical fiber etc.
(2) Nodes: Every node that participates in the ad-hoc network is assumed to have the structure shown in Figure 1. That is, it has

- Processing and storing capabilities as demanded by the protocol,

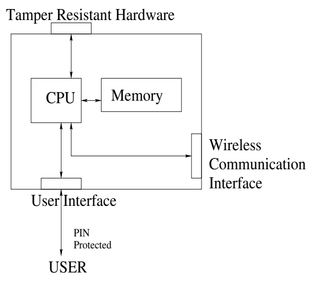

Figure 1: A valid node in the network.

- Tamper Resistant Hardware: the capabilities of this hardware are described in the next paragraph,
- Interface for the tamper resistant hardware.

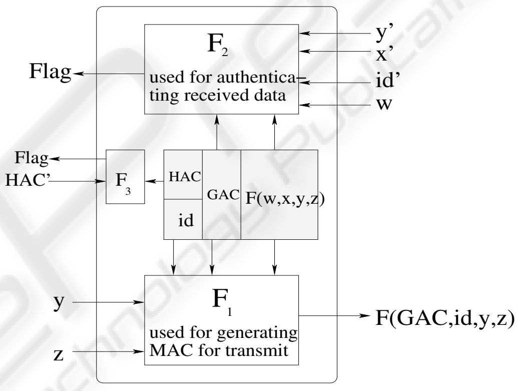
F2
used for authenticating received data
Flag
y'
x'
id'
w
Flag
HAC'
F3
HAC
id
GAC
F(w,x,y,z)
y
z
F1
used for generating MAC for transmit
F(GAC,id,y,z)

Figure 2: Hardware on the node.

The structure of the ideal tamper proof hardware (TPH) for UGSP is shown in Figure 2. The salient features of the TPH are given below:

- a universally unique, unalterable id
- Write only memory for group access code (GAC) and hardware access code (HAC), attempt to overwrite HAC destroys GAC as well.
- non invertible pseudo random hash function $F(w, x, y, z)$ embedded in the hardware (Lamport, 1981)
- provides service of three functions F_1 (used to generate message digest for transmission), F_2 (used for authenticating the message digest received from the sender) and F_3 (used for local authentication of the HAC) defined below:
  $F_1(y, z) = F(GAC, id, y, z)$
  $F_2(w, id', y', x')$
  $= TRUE\ if\ \ w = F(GAC, id', x', y');$
  $= FALSE\ \ otherwise$
  $F_3(HAC', HAC) = TRUE\, if\ HAC = HAC'$
  $= FALSE\ \ otherwise$
- Tamper resistance of the hardware is interpreted as:

  - Codes of functions $(F(w,x,y,z), F_1, F_2, F_3)$ and id cannot be accessed by any one, and
  - For F_1, id and GAC are implicit (or hidden) parameter, for F_2, GAC is implicit parameter and HAC is implicit parameter for F_3.
- Password Enabled Access: the use of the hardware will be restricted by a HAC. Only after a user has entered the HAC correctly, he will be able to use the TPH. This ensures source authenticity at the user level.

Imprintable TPH can be obtained in the market and the user can download the image for the specific application over a secure channel to imprint the TPH.
(3) User Group Administrator: We envisage the presence of a User Group (UG) Administrator with the following roles:

- Assign a GAC for the UG
- Develop and maintain image for imprinting TPH
- Provide image and configure the TPH on request by a user on getting necessary information.

The role of UG administrator is limited to configuring the nodes and does not have any role during the interaction between the end nodes.

3 UGSP: USER GROUP SECURITY PROTOCOL

The TPH token is not integrated with the mobile host, but the user carries the token with himself.
The protocol consists of the following steps:

1. First, every node that is allowed into the network is initialized by the group administrator by writing the HAC (user specific) first and then the GAC (which is same for all nodes) in the write only memory of the tamper resistant hardware. HAC is written before GAC because write to HAC overwrites GAC also. The write-only memory is interpreted as follows: once the secret is written into the memory, no one can read the secret. That is, a node can be a part of a DUG ad-hoc network provided it has the TPH with the same GAC.
2. The user will carry the configured TPH with him as a token. In order to use the hardware, the user will be required to interface it with his mobile device and to enter the HAC correctly, which will be locally authenticated within the hardware. Only after successful authentication, he can use the hardware for the subsequent steps.
3. Communicating nodes generate 1024 bit RSA key pair and exchange their public keys using the MAC generated by the TPH.
4. The sender now generates a 64 bit DES symmetric key, encrypt it using the receiver Public Key, append a MAC generated by the hardware and transmit. The receiver also confirms that he has received the correct symmetric key by sending the symmetric key encrypted with the sender Public Key.

Note that due to the underlying tamper proof hardware, the following properties are satisfied:

- No user has any control over implicit inputs to the functions F_1, F_2 and F_3.
- When a node wants to transmit data, it gives the data to the TPH which in turn evaluates the MAC using function F_1 and transmits. Note that the user cannot change the first two inputs of function F_1, i.e. the GAC and id of the node (they are taken automatically from the memory of the hardware). Because the id cannot be changed, active impersonation is not possible in the network.
- Now consider a node receiving data and the corresponding MAC. The node now needs to authenticate the MAC against the data, id of the sender node and the GAC. For this purpose, the node passes the received data and the sender id to the TPH, which computes the function F_2 taking the GAC stored in its memory. If the result of F_2 is $TRUE$ then the node will accept the data otherwise the node realizes that there is some mismatch in data, id or GAC, and rejects the packet and terminates the communication.

3.1 Formal description

Notation

- Nodes *i* and *j* want to communicate with each other, where i is the sender and j is the receiver.
- *F(w,x,y,z)* is the pseudo random hash function implemented in the TPH, which takes in four input parameters w, x, y, z
- *id(k)* is the identity of the node *k*
- K_j is the public key of the node *j*
- $E_{K_j}(N)$ means *N* encrypted with K_j
- *N* denotes the $nonce$ which is unpredictable.
- $(k \rightarrow l : M)$ is interpreted as follows: node *k* is sending message, *M*, meant for node *l*
- K_{sy} is the symmetric session key established between the pair of communicating nodes
- $(M_1\ ,\ M_2)$ means message M_1 concatenated with M_2
- For ease of reference in the sequel, we shall refer the value of $F(w,x,y,z)$ as MAC.

In UGSP, even if the attacker is able to predict the $nonce$, he will not be able to learn any secret in the network unlike most other protocols (will become clear in the sequel).

Every time a node wants to transmit, it computes MAC using F_1 and transmit, and a receiving node to authenticate the received data will compute F_2 as mentioned above. With this background, we are ready to describe the steps in the protocol formally.

The protocol consists of two Phases. Phase I of the protocol consists of bootstrapping of the valid nodes of the network, while the second phase will be session specific and will happen as and when nodes want to communicate with each other. In more hostile environments, such as battlefield, where the chances of bootstrapped nodes getting compromised is higher, one can make bootstrapping also session specific.

Operational steps in the execution of UGSP:

$Phase\ I$: the network is being formed. In Step 1, the user gets authenticated by the TPH by the HAC. User inputs the HAC when prompted to do so, which will be compared with the HAC stored on the TPH. This can be thought of as if the smart card PIN is stored on the smart card and which can only be read by the smart card reading machine. Hence, user authentication can be done by a standalone ad-hoc smart card machine, not connected to the bank's back-end server, by comparing the PIN stored on the card with the PIN entered by the user. In Step 2, the communicating nodes, which have been authenticated in step 1, will exchange RSA keys.

$Phase\ II$: will be invoked when any node want to communicate with some other node in the network. During step 3, the two communicating nodes establish a symmetric key. We could directly use the private-public key pair established in step 3 for secure communication, but that will be bandwidth and computation expensive. Hence, we need to establish a symmetric key, which is accomplished after this step.
We encrypt the data using the symmetric key and append a MAC computed using the data and the symmetric key and transmit in step 4. As highlighted already, UGSP uses the Public Key operations in a limited way as compared to other protocols.

PHASE I : Bootstrapping
Step 1 : local authentication of HAC
1.a : Node i : user i enters his HAC
1.b : Node j : user j enters his HAC

Step 2 : exchange public keys
2.a : $\mathrm{i \rightarrow j : F(GAC, id(i), N_1, K_i), N_1, K_i, id(i)}$
2.b : $\mathrm{j \rightarrow i : F(GAC, id(j), E_{K_i}(N_1), K_j), N_2, K_j, id(j)}$
2.c : $\mathrm{i \rightarrow j : F(GAC, id(i), E_{K_j}(N_2), *)}$

PHASE II : Session specific
Step 3 : establish a shared symmetric key using the established public key
3.a : $\mathrm{i \rightarrow j : F(GAC, id(i), N_3, E_{K_j}(K_{sy})), N_3, E_{K_j}(K_{sy})}$
3.b : $\mathrm{j \rightarrow i : F(GAC, id(j), E_{K_{sy}}(N_3), E_{K_i}(K_{sy}))}$
3.c : $\mathrm{i \rightarrow j : F(GAC, id(i), E_{K_{sy}}(id(i)), *)}$

Step 4 : data transfer
4.a : $\mathrm{i \rightarrow j : E_{K_{sy}}(Data), F(id(i), data, K_{sy})}$

Figure 3: Protocol for communication between Node i and Node j

4 ANALYSIS OF THE PROTOCOL

In this section, we will analyze UGSP with respect to security assurance and scalability.

4.1 Security Analysis

In this section we will discuss possible attacks in ad-hoc network and analize the security assurance provided by UGSP as well as the approaches of TESLA and PKI based security system. We shall show robustness of our protocol with respect to outside attack and the risks of compromised nodes.

4.1.1 Attacks from outside the network

When we say a node is outside the network, we mean a node that does not have the TPH token configured with the GAC for that group, say GAC_1. In this case, either the attacker will fail to authenticate himself to the hardware token in the $Step$ 1 itself (cf. Figure 3) or if the user knows the valid HAC for the TPH (which means that he is a valid user for some other similar network with group access code GAC_2) then he will generate the MAC corresponding to GAC_2. Such a MAC would fail to match with the MAC generated at the receiving end because of the different GAC. While if such a node wants to become a receiver, then it will fail to authenticate itself to the sender as it will not be able to form a valid $Packet$ $2.b$ (cf. Figure 3) because of a different GAC. Thus, such a node would fail to be a sender as well as a valid receiver in the network.

4.1.2 Attack from a compromised node

Here, the attacker is assumed to have the valid compromised node, i.e. a node with the TPH token configured with the group access code for that group. Here, again two level of attacks are possible:

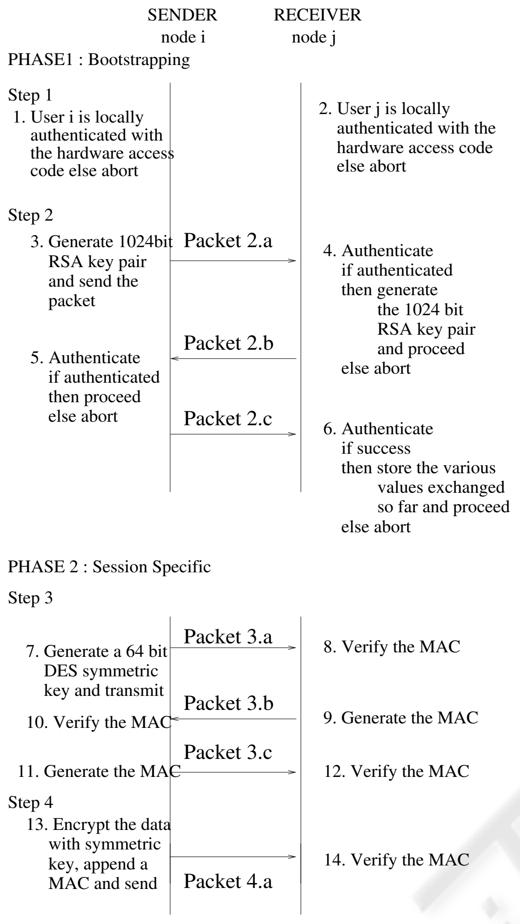
SENDER
node i
RECEIVER
node j
PHASE1 : Bootstrapping
Step 1
1. User i is locally authenticated with the hardware access code else abort
2. User j is locally authenticated with the hardware access code else abort
Step 2
3. Generate 1024bit RSA key pair and send the packet
Packet 2.a
4. Authenticate if authenticated then generate the 1024 bit RSA key pair and proceed else abort
Packet 2.b
5. Authenticate if authenticated then proceed else abort
Packet 2.c
6. Authenticate if success then store the various values exchanged so far and proceed else abort
PHASE 2 : Session Specific
Step 3
Packet 3.a
7. Generate a 64 bit DES symmetric key and transmit
8. Verify the MAC
Packet 3.b
9. Generate the MAC
10. Verify the MAC
Packet 3.c
11. Generate the MAC
12. Verify the MAC
Step 4
13. Encrypt the data with symmetric key, append a MAC and send
Packet 4.a
14. Verify the MAC

Figure 4: Algorithm at the nodes (Packets are labeled in Figure 3).

- *Attacker does not know the HAC:* The authentication will fail during Step 1 (cf. Figure 4). While if the attacker tries to overwrite the HAC, the GAC also gets overwritten (as mentioned above, TPH token is has been designed with such a feature) and then the case becomes similar to the one mentioned in Section 4.1.1.
- *Attacker knows the HAC:* In this case the attacker can send valid packets to the network, but with his own id. Active impersonation is not possible in the network. The attacker cannot simulate the hardware behavior and try to pass any other id to the model, because the group access code is not known outside the hardware. In such a scenario, if other nodes in the network come to know that a node is compromised, then they can block that particular node from any communication because when that node wants to participate in the communication, it can do so with its own identity, which is blocked.

In case of PKI or TESLA, when the node gets compromised the attacker would get the keys and will be able to actively impersonate the user. Even though the keys may be password protected but note that they would be accessible through brute-force methods (such as bit by bit reading of the hard disk), to which TPH is resistant.

4.2 Scalability

After the configuration of HAC and GAC at the hardware, the key establishment and data transfer etc can be initiated as per the ad-hoc network requirements. The nodes can form an ad-hoc network with other nodes in DUG securely without having to maintain any database, without going to trusted third party, without requiring any other authenticated channel for each data transfer. Thus, UGSP is suitable for ad-hoc wireless network security. Note that, the cost of TPH required for the purpose is available and is quite cheap. In other words, UGSP is quite scalable.
Let us consider the needs of TESLA and PKI-based security from the perspective of scalability. In TESLA, every time two nodes want to communicate, they will be required to exchange some information (like: T_{int}, key disclosure delay, key commitment to the key chain) over an authenticated channel. This is not possible in ad-hoc environment as it requires constant presence of a trusted third party.

Now, let us consider PKI-based security: Assume that each node eligible to join the network has been given a public-private key pair by PKI. Thus, when any node wants to communicate, it will send the certificate along with the data. For the other node to verify the certificate either it has to go to the trusted third party (which is not possible in ad-hoc network) or maintain its own database of public keys of the other nodes (which is not scalable). Due to the property that on-line authentication is needed, a typical PKI-based security is not feasible for ad-hoc network security.

5 IMPLEMENTATION DETAILS

We have implemented UGSP using $iButton$[1]. The $iButton$ is a computer chip enclosed in a 16mm stainless steel can available off-the shelf for less than ten dollars in retail. Note that the $iButton$ provides more functionality than what is needed for UGSP. We have used it as it is available off-the shelf. The steel button can be mounted virtually anywhere because it is rugged enough to withstand harsh environments, indoors or outdoors. Each $iButton$ has a unique and unalterable address that is laser etched onto its chip inside the can. In response to tampering, the $iButton$ would rather erase the key than reveal its secrets.

[1]www.ibutton.com

Each has an onboard 512-bit SHA-1 engine that can compute 160-bit MACs in less than 0.0005 seconds as compared to 0.5 seconds for a typical microcontroller. $iButton$ can be interfaced with a host system via serial/parallel port or USB. Our protocol has been tested with $iButton$. Mobility of the nodes was also emulated and multihop routing scenarios were evaluated against performance and energy cost. However, $iButton$ does not realize the function F_2 but indicates that the TPH described in the protocol can be developed at a much cheaper cost. We have completed the design of the required TPH and are in the process of testing it on an FPGA board.

6 DISCUSSIONS

To sum up, it can be seen that UGSP is resilient to all attacks on an ad-hoc network forming a DUG mentioned earlier. UGSP is based on mutual authentication rather than only the client authenticating to the server, or only the sender authenticating to the receiver. Our protocol provides dual security since we are using a TPH token and access code for using the TPH. Thus, even if the configured hardware token is stolen or compromised, an attacker cannot use the token without knowing the valid hardware access code. In a sense, it achieves security using the paradigm of *"Something you know, and something you get"* providing dual security to the network. This concept is similar to the one used by banks for cash dispensation at ATM's (a combination of card and PIN is required to access the account).

Based upon our experience of using the prototype, we have found that implementation of UGSP can be done in cost-effective way. UGSP is scalable and robust to addition of new members in the User Group. In this paper we have demonstrated and discussed UGSP for data transfer in a User Group in mobile ad-hoc network. However, there are certain generalizations possible as stated follows:

Communication Protocol Independent: Using UGSP, we are able to establish a secure communication channel between nodes at the end of Phase 1. Once this happens, we can use any of the existing protocols, such as TESLA, for data communication.

Multiple Applications: Although, we have chosen data transfer as a sample application for the demonstration of the protocol, UGSP can be used for several purposes like authenticated routing, node-to-node key agreement and ubiquitous computing.

Network Infrastructure Independent: UGSP has been developed for mobile ad-hoc networks, but it is equally efficient in wired networks as well. It replaces PKI in the sense that there is no need to go to the trusted third party everytime you want to validate any certificate.

Membership to Multiple DUG: The TPH token can be made to have more than one location for storing the GAC. A node can thus be a valid user in more than one different ad-hoc networks simultaneously. $iButton$, for example, has eight locations for storing GAC. We are evaluating the system performance when a node is a part of eight simultaneous ad-hoc networks.

REFERENCES

Bobba, R. B., Eschenauer, L., Gligor, V., and Arbaugh, W. A. (2002). Bootstrapping security associations for routing in mobile ad-hoc networks. In *Technical Report, Institute for Systems and Research, UMD, TR 2002-44*.

Hu, Y., Johnson, D., and Perrig, A. (2002). Sead: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *Workshop on Mobile Computing Systems and Applications, IEEE*.

Hu, Y.-C. and Perrig, A. (2002). Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Mobicom*.

Khaili, A. and Arbaugh, W. A. (2002). Security of wireless ad hoc networks. In *http://www.cs.umd.edu/ aram/wireless/survey.pdf*.

Khalili, A. and Arbaugh, W. (2003). Toward secure key distribution in truly ad-hoc networks. In *IEEE Workshop on Security and Assurance in Ad-Hoc Networks*.

Lamport, L. (1981). Password authentication with insecure communication. In *Communications of the ACM, pg. 770-771, Number 81, Volume 24*.

Papadimitratos, P. and Haas, Z. (2002). Secure routing for mobile adhoc networks. In *Communication Networks and Distributed Systems Modeling and Simulation Conference*.

Perrig, A., Canetti, R., Tygar, J., and Song, D. (2002a). The tesla broadcast authentication protocol. In *RSA Cryptobytes*.

Perrig, A., Szewczyk, R., Tygar, J., Wen, V., and Culler, D. E. (2002b). Spins: Security protocols for sensor networks. In *Wireless Network Journal (WINE)*.

Royer, E. M. and Toh, C. K. (1999). A review of current routing protocols for ad hoc mobile wireless networks. In *IEEE Personal communications*.

Stajano, F. and Anderson, R. (1999). The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the 3rd AT & T Software Symposium*.

Toh, C. K. (2001). Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks. In *IEEE Communications Magazine*.

Zhou, L. and Haas, Z. (1999). Securing ad hoc networks. In *IEEE Network Magazine, 13(6)*.