SECURE GROUP COMMUNICATION WITH DISTRIBUTED GENERATION OF PRIVATE KEYS FOR AD-HOC NETWORKS

Shrikant Sundaram[1], Peter Bertok[1] and Benjamin Burton[2]
[1]School of Computer Science and Information Technology;
[2]School of Mathematical and Geospatial Science;
RMIT University, Melbourne, Australia

Abstract: Mobile ad-hoc networks are emerging as important computing platforms, and their users expect the security to be comparable to fixed, infrastructural networks. With the increase of group-oriented applications, secure communication has to be provided between a group of nodes that may join and leave the network in an unplanned manner. Previous solutions have achieved limited success, for example did not provide manageable confidentiality of messages. This paper proposes a fast, scalable group encryption key model that is suitable for ad-hoc networks and is resistant against multiple compromised nodes.

Key words: key-insulated, ad hoc networks, key generation, threshold cryptography

1. INTRODUCTION

With the increasing accessibility of wireless networks, securing information exchange is becoming more and more important as these networks are easy to penetrate. The increasing popularity of collaborative applications has raised interest in secure group communication, primarily message authentication and confidentiality. In large groups, setting up secure channels for each communicating pair of nodes can be very expensive, so a common communication platform, including keys for the whole group is often required.

Managing group communication can become quite complex as the keys should be available to all group members but not to outsiders. On the other hand, new members have to be provided with the key, possibly without access to earlier massages, and when a member leaves, it should have no access to later messages. Some solutions utilise a group leader approach to calculate and distribute keys[1] while other approaches arrange the nodes in a key tree that connects group members for efficient communication[7].

Reliability of communication can also be a problem in wireless networks. However, some solutions[12] have already been proposed to overcome this problem, so we do not address it here.

In this paper we propose a key management approach, primarily for ad-hoc networks. In our method, there is only one public key for the whole group and each member has a unique private key that operates with the public key and the member's own unique identity (IP or MAC Address). A certain number of current group members collaborate to provide a new member with the private key. The private key's validity is limited in time and has to be updated on a regular basis to prevent the member from unlimited access to group messages. The validity is certified by a *key certificate* that each group member carries. The *key certificate* is signed with a secret key that never appears in its entirety at any node, but is distributed among the group members. With a minor extension described later, our proposed solution also securely operates in networks where messages between group members have to pass through other nodes in the key distribution phase, a common scenario in ad-hoc networks.

We assume that an authentication scheme, for example one based on certificates, is in use in the network.

In the following section we describe some related work in the area of security in ad hoc networks. In Section 3 we describe our proposed solution and in Section 4 we explain the mathematical background. Section 5 describes the results of our implementation of the proposed solution, followed by discussion and future work in Section 6. Finally, we provide our conclusions in Section 7.

2. PREVIOUS WORK

2.1 Existing Communication Management Solutions

Secure communication in ad hoc networks has been widely discussed recently, and it is very challenging due to the dynamic nature of the network and the roaming devices. There have been a number of solutions proposed, for authentication and key generation/distribution that range from leader based key management schemes[1] to self-organized schemes[2]. Threshold cryptography[8-10, 13] can greatly improve the key generation and distribution process and can tolerate a certain number of node failures. We assume that the reader is familiar with threshold cryptography; in the next section we explain the key insulated scheme that forms the basis of our solution.

2.2 Key-Insulated Cryptography and Signature Scheme

Dodis et al.[3, 4] describe *Key-insulated cryptography and signature schemes* to enable small insecure devices to periodically refresh their private keys at discrete time intervals, without changing the public keys.

The insecure devices update their private keys with the help of another device that we call a base station. The base station is considered to be physically secure. An example scenario can be a mobile phone or PDA that updates its private key with the help of the personal computer of the user.

A *[t, N]* key insulated scheme is one where an adversary who compromises a physically insecure device and obtains private keys for up to *t* time periods is unable to decrypt or sign messages for all other *(N - t)* time periods.

The device gets its initial private key and the public key from a key generation algorithm. The key generation algorithm also generates a master private key that is stored at the base station. All encryption/decryption and signing is done on the insecure device with the private keys and the time period information. The base station only helps the device to update its private keys.

When a device needs to update its private key, the base station calculates a partial key and sends it to the device. The device calculates a new private key from the partial key and its previous private key. The updating algorithms allow random access key updates given any two time periods. Even though the base station is considered to be secure, if it is compromised and the master key is exposed, the adversary cannot gain knowledge of any private keys. However, the keys need to be updated from a single, secure device that may not be possible in ad hoc networks.

2.3 Our Contribution

Our aim is to facilitate secure group communication in two ways: reduce the burden of encryption key management by using fewer keys, and to assist mobile and ad-hoc type networks by not relying on a single, fixed and well secured base station.

Our approach has common roots with the threshold scheme[8-10, 13]; however, we provide key generation rather than authentication. We derive our solution from key insulated schemes[3, 4], but with two major additions.

Firstly, we use a single public key for group communication, and individual nodes have their own, individual private keys that are refreshed periodically. Secondly, we do not store the master private key anywhere in the system; instead it is destroyed after initialisation. We dispense with the secure base station used there, and the keys are generated and updated in a distributed manner such that no node, except the intended node, can gain the knowledge of the new private key. Minor improvements, such as using the combination of node identifiers and time period information instead of time only, provide further security in the proposed solution. A small extension makes our solution work for communication between non-adjacent nodes; as is frequently the case in wireless networks.

3. PROPOSED SOLUTION

An ad hoc group consists of several nodes, all which are identified by a certificate and hence all nodes have a public key-private key pair. This key pair is used by the nodes to authenticate themselves to the group. Once they have been authenticated they are issued a public key - private key pair for use within the group. In this key pair the public key is common for the whole group while the private key is known only by that node. The private key is generated by a group of nodes in the group using the principles of threshold cryptography. The private key is based on the identity information of the node and time information. The identity can be the public key of the node or its MAC/IP address and is to be unique for that node. The time information is a time period value that specifies the validity of the private key.

The model supports encryption, authentication and integrity of messages. To send an encrypted message to another member of the group, we encrypt the message with the group public key and the unique identifier of the receiver member. The node with the given identifier decrypts the message with its private key. To sign messages we encrypt with the sender's private key and it can be verified by all with the global public key. These algorithms are the same as in the key insulated schemes [3, 4] and are not discussed further here. Our focus is on distributed key generation, distributed updating and validation of the public and private keys.

The group is a *(t, N)* secure group where an adversary would need to compromise *t* nodes out of the total *N* nodes in the group to break the security of the group.

3.1 Initialisation

In the initialisation phase, a trusted server or dealer generates the public key, the master private key and initialises $t + 1$ nodes with their private keys. The dealer calculates the private keys of $t + 1$ nodes, deletes the master key and disappears from the network. This is a common assumption in threshold cryptography based models[8-10, 13].

3.2 Key Generation and Updating

The nodes update their private keys when the time period information changes. The reason to update the private keys is to renew trust. A node receives its private key when it first comes online, but an adversary can compromise it at some later time. Compromises can go undetected to some nodes and it may be difficult for the group as a whole to generate a revocation certificate. In addition, not all nodes in the group may receive the revocation certificate. Hence, the best possible way, in our view, is to renew trust by refreshing the private keys.

A node gets its private keys from a coalition of $t+1$ nodes that are within one hop distance of that node. We require the authenticating nodes to be within one hop distance, as in ad hoc networks, nodes cannot trust the intermediate nodes to route the authentication information. In section 4.4 we propose extensions whereby the authenticating nodes can be more than one hop away but at the price of additional encryption that affects performance.

The joining node first finds a coalition of $t+1$ nodes in its neighbourhood. It then generates a list of the identifiers of the $t+1$ nodes and sends the list, along with a request for a group private key, to all of the $t+1$ nodes in the list. It can append additional information about itself (like a certificate) to facilitate authentication.

The authenticating $t+1$ nodes verify the node's identity and check the node's validity by checking their revocation lists. Once they are satisfied with the legitimacy of the node, they will generate partial private keys and send them to the node. Finally, the joining node will construct a full private key from the individual partial private keys.

3.3 Key Certificate Generation

A node v_j uses its private key to sign outgoing messages and decrypt incoming messages sent by other nodes in the network. As many of these communicating nodes may not have been part of the coalition that authenticated and generated the private key for node v_j, nodes need to be able verify that v_j got the keys from legitimate nodes. In addition, our model requires that nodes update their private keys regularly for improved security. Thus all nodes need to know when and how the node got its private key. For this end, each node carries a *key certificate* that contains the node's identity, the coalition that authenticated the node, the time at which the private key was given and the expiration time of the private key.

3.4 Communication Establishment and Termination

Once a node has been authenticated and has been given a private key, the node can communicate with any node in the group with the group public key and its private key pair. This key pair can either directly protect the whole communication, or be used only for negotiating a temporary session key that is symmetric in nature. The manner of communication is left open and can be decided by the group policies and requirements.

4. ALGORITHMS AND MATHEMATICAL BASICS

4.1 Mathematical Basics

Our solution is based on key insulated cryptography and signature schemes [3, 4]; first we give some background information about the algorithms and terminology in the key-insulated schemes.

In key insulated schemes, a key generation algorithm takes as input a security parameter k that is the bit length of a prime number q. The prime number q is chosen such that $p = 2q + 1$ is also prime. This defines a unique subgroup $\mathbf{G} \subset \mathbf{Z}_p^*$ of size q where the Decision Diffie-Hellman (DDH) assumption[11] is assumed to hold. Two random elements $g, h \in \mathrm{G}$ are selected and two random polynomials $f_x(\tau) = \sum_{j=0}^{t} x_j^* \tau^j$ and $f_y(\tau) = \sum_{j=0}^{t} y_j^* \tau^j$ of degree t are defined over $\mathbf{Z}_q$. The public key consists of g, h and $z_0^*, z_1^*, \ldots\ldots, z_t^*$, where $z_0^* = g^{x_0^*} h^{y_0^*}, \ldots\ldots, z_t^* = g^{x_t^*} h^{y_t^*}$. The initial private key is (x_0^*, y_0^*), that is the initial coefficient of the polynomial, and the remaining coefficients $(x_1^*, y_1^*, \ldots\ldots, x_t^*, y_t^*)$ are stored in the base station as the master private key. The remaining private keys of the device are the two polynomial evaluations $f_x(i)$ and $f_y(i)$ with i as the time period. These private keys are updated with the master private key and the old and new time period. Fig. 1 sums up the terminology used in the key insulated schemes.

$PK := (g, h, z_0^*, \ldots.., z_t^*)$ where $z_0^* = g^{x_0^*} h^{y_0^*}, \ldots.., z_t^* = g^{x_t^*} h^{y_t^*}$

$SK^* := (x_1^*, y_1^*, x_2^*, y_2^*, \ldots\ldots, x_t^*, y_t^*)$;
$SK_0^x := x_0^*\ SK_0^x := y_0^*$

$$SK_i^x = x_0^* + x_1^* i + x_2^* i^2 + \ldots. + x_t^* i^t = \sum_{s=0}^{t} x_s^* i^s$$

$$SK_i^y = y_0^* + y_1^* i + y_2^* i^2 + \ldots. + y_t^* i^t == \sum_{s=0}^{t} y_s^* i^s$$

where $1 \le i \le N$ is time period information

Figure 1. Background terminology

Thus, in key insulated schemes, nodes maintain a public and private key pair and change their private keys at discrete time intervals without changing the public key.

In our proposed solution, we make two important changes to the key insulated schemes. Firstly, each node maintains its own unique individual private key (SK_i^x, SK_i^y) based on the parameter i that is now a combination of the time period information and the unique identity of the node. Thus the private keys are generated for all nodes in the group for a single global public key. From now on we refer to i as the unique identifier of a node.

Secondly, we do not maintain the master private key anywhere in the system. The private keys for nodes are generated and updated by a coalition of $(t+1)$ neighbouring nodes, where t is the degree of the two random polynomials described earlier.

4.2 Private Key Generation and Updation

Without loss of generality, let us assume that the new node v_j finds a coalition $\beta = \{v_1, v_2, \ldots\ldots, v_{t+1}\}$. It sends this list to all nodes in β with a request for a private key. Each node v_i in the coalition β would authenticate the requesting node and generate partial shares $Psh_i^x = SK_i^x . \prod_{r=1, r\neq i}^{t+1} \frac{v_j - v_r}{v_i - v_r}$ and $Psh_i^y = SK_i^y . \prod_{r=1, r\neq i}^{t+1} \frac{v_j - v_r}{v_i - v_r}$

Using Lagrange's interpolation, node v_j can calculate its private key by adding the x and y components of the above partial shares respectively.

$$SK_j^x = Psh_1^x + Psh_2^x + \ldots + Psh_{t+1}^x = \sum_{r=1}^{t+1} Psh_r^x = \sum_{s=0}^{t} x_s^* v_j^s$$

$$SK_j^y = Psh_1^y + Psh_2^y + \ldots + Psh_{t+1}^y = \sum_{r=1}^{t+1} Psh_r^y = \sum_{s=0}^{t} y_s^* v_j^s$$

4.3 Distributed Generation of Key Certificates

The *key certificate* contains a message $\boldsymbol{M}$ that runs something like, "This is to certify that node v_j has been issued a private key by coalition β at time T and it would expire at time $T + T_{lifetime}$." The value $T_{lifetime}$ is the duration for which the certificate is valid. The certificate is signed with a signing key that is known by none of the nodes in the group, which is the initial private key (SK_0^x, SK_0^y). This key can also be used to generate group messages to other groups in the network. No single node knows this secret signing key as all nodes have non-zero identity values. However, $t+1$ nodes can generate a signature using Lagrange's interpolation, without revealing the key at any time.

The coalition of nodes β generates the *key certificate* of node v_j along with the private key. Since all nodes in the coalition know the time periods T and $T_{lifetime}$ together with the coalition list β, the certificate message $\boldsymbol{M}$ is well known. Thus, each node in the coalition β can construct the certificate message $\boldsymbol{M}$ independently. The certificate-signing algorithm is similar to the signing algorithm in key-insulated signature schemes except that the certificate signature is generated in a distributed manner.

To sign the *key certificate* of node v_j, each node v_i generates two random numbers $r_{i,1}$ and $r_{i,2} \leftarrow \mathbf{Z}_q$. It then calculates $w_i = g^{r_{i,1}} h^{r_{i,2}}$ and sends w_i to all nodes in the coalition ß so that all nodes in the coalition can calculate $w = w_1 \times w_2 \times \ldots\ldots \times w_{t+1}$. This is equivalent to $w = g^{r_{1,1}+r_{2,1}+\ldots+r_{t+1,1}} h^{r_{1,2}+r_{2,2}+\ldots+r_{t+1,2}} = g^{r_1} h^{r_2}$ where $r_1 = r_{1,1} + r_{2,1} + \ldots. + r_{t+1,1}$ and $r_2 = r_{1,2} + r_{2,2} + \ldots. + r_{t+1,2}$ and $r_1, r_2 \in \mathbf{Z}_q$.

Once all nodes have calculated w, each node generates a hash $\tau = H(0, M, w)$ of the certificate message $\boldsymbol{M}$ and w. Each node v_i further calculates $a_i = r_{i,1} - \tau P_i^x$ and $b_i = r_{i,2} - \tau P_i^y$ where $P_i^x = SK_i^x . \prod_{r=1, r\neq i}^{t+1} (\frac{v_r}{v_r - v_i})$ and $P_i^y = SK_i^y . \prod_{r=1, r\neq i}^{t+1} (\frac{v_r}{v_r - v_i})$.

Similar to the private key generation algorithm, node v_j gets the full signature of the certificate by adding up $a = a_1 + a_2 + ... + a_{t+1}$ and $b = b_1 + b_2 + ... + b_{t+1}$. With Lagrange's interpolation we get $a = r_1 - \tau SK_0^x$ and $b = r_2 - \tau SK_0^y$ which is equivalent to a message signed with the initial private keys in the key insulated schemes. The final generated certificate is of the form $(M, [0, (w, a, b)])$. The certificate can be verified by anyone who has the public key $PK := (g, h, z_0^*, ..., z_t^*)$.

Steps			
1	Let $\beta = \{v_1, v_2, v_3, ...v_{t+1}\}$	6	$w = w_1 \times w_2 \times ... \times w_{t+1}$
2	for each node v_i in $\mathfrak{B}$	7	$\tau = Hash(0, M, w)$
	$r_{i,1}, r_{i,2} \leftarrow \mathbb{Z}_q$	8	$a_i = r_{i,1} - \tau P_i^x, b_i = r_{i,2} - \tau P_i^y$
3	$w_i = g^{r_{i,1}} h^{r_{i,2}}$		where $P_i^x = \sum_{s=0}^{t} x_s^* v_i^s . \prod_{r=1,r\neq i}^{t+1} (\frac{v_r}{v_r - v_i})$
4	send w_i to all nodes in $\mathfrak{B}$		and $P_i^y = \sum_{s=0}^{t} y_s^* v_i^s . \prod_{r=1,r\neq i}^{t+1} (\frac{v_r}{v_r - v_i})$
5	collect all $w_1, ..., w_{t+1}$	9	node v_j calculates $a = a_1 + a_2 + ... + a_{t+1}$ and $b = b_1 + b_2 + ... + b_{t+1}$

Figure 2. Key certificate generation algorithm

4.4 Protecting the Partial Key Shares

Whenever a node v_i sends a share Psh_i^x, Psh_i^y, there is a serious risk that an adversary may eavesdrop and extract the private key of the node. If an adversary gets $Psh_i^x = SK_i^x . \prod_{r=1, r\neq i}^{t+1} \frac{v_j - v_r}{v_i - v_r}$ it can calculate the private key of the node v_s, since the second parameter in the equation $\prod_{r=1, r\neq i}^{t+1} \frac{v_j - v_r}{v_i - v_r}$ can be easily calculated if the coalition β is known. This is a serious security concern as even the node that requests the key can try to gain knowledge of the private keys of its authenticators.

There are two ways of protecting the partial shares of the authenticating nodes. One is by mixing the partial shares with another value so that the transferred value does not reveal any information about the actual partial shares. This is known as shuffling. The second way is to encrypt the partial shares with the public key of the receiving node so that only that node can decrypt it.

4.4.1 Shuffling

All nodes in the group securely exchange secret factors with each other. Specifically, two nodes v_a and v_b in the coalition β can securely exchange a factor $d_{a,b}$. The node with the higher id, say *a*, would then add the factor to the partial share Psh_a while the other node *b* would subtract the factor from its share Psh_b. These nodes would similarly exchange more such factors with other nodes in the network. When the joining node adds up all the partial shares it receives from the coalition, the factors cancel out and the final result is still the new private key of the node. Luo and Lu[9] first described this process.

4.4.2 Public Key Encryption of Shares

The partial shares can also be encrypted with the public key of the joining node. This public key is not the group public key but a personal public key of the node that could be authenticated in the form of a certificate. When the nodes generate the partial shares, they would encrypt the shares with the public key of the joining node so that only the intended node can decrypt the share with its private key.

This method puts additional burden on the joining node, as it has to decrypt the shares, but it is more robust than the shuffling method as the authenticating nodes can be more than one hop distance away from the joining node.

4.5 Certificate Revocation

We require that nodes continuously update their private keys so that faulty nodes can gracefully leave the network. Another advantage is that compromised nodes cannot operate long in the network. However, we still need a mechanism to evict a compromised node immediately from the network as keys are refreshed after some time period and we may not want a compromised node to be in the network till its time for it to update its keys.

To revoke the certificate of compromised or faulty nodes we propose to use a model similar to those suggested by[6, 8-10]. When a node is found to be faulty, $t+1$ nodes need to sign a special counter certificate that would nullify the presence of the node in the network. The counter certificate is signed by the same key that was used to sign certificates. The counter certificate is sent to all possible nodes in the network by flooding.

A node needs to be accused by at least $t+1$ nodes before being marked as 'faulty' and evicted from the group. Any number of nodes less than the coalition size $t+1$ cannot evict a node from the group.

5. RESULTS

We implemented the algorithms in Java and used the Jini platform for communication. We used Java's built-in classes for hashing messages and support for large integers (java.math.BigInteger). The Jini platform allowed the program to be independent of the network configuration and allowed a certain degree of interoperability. We can, however, replace Jini with any other communication platform.

5.1 Setup

We tested the authentication process in a laboratory environment with wired and wireless networks, and with different types of machines being the nodes of the ad hoc network. The experiments tested the computational costs of a worst-case scenario and compared it with network latency. There was no other network traffic that could affect the results.

The testing environment had three PCs with Pentium III, 1 GHz processors, 256Mb RAM and running Debian Linux. These PCs were connected to other PC's through a wired Ethernet switch. Two more PC's with the same configuration but running Gentoo Linux were connected via wireless LAN. We name two of the wired Pentium PCs as *pentium1* and *pentium2* and the Wireless PCs as *wireless1* and *wireless2*.

Besides these, there were two Sun Blade 150 workstations, each with a 650MHz processor and 512Mb RAM (*blade1* and *blade2*) and two Sun Netra X1 machines, with 500MHz processors and 1024Mb RAM (*sparc1* and *sparc2*). All Sun machines were running the Sun Solaris 8 operating system.

All these machines were connected via Ethernet (100Mbps) while the wireless machines were connected to the wired network through an access point. The wireless connections were operating at 56Mbps. All machines had Java 1.4.2 and Jini 2 installed locally.

We tested the algorithms for a threshold limit t of 8. A Jini lookup service provider was running on another Sun Netra machine that facilitated the locating of the nodes. Once the remote nodes had been located, a proxy object was downloaded and used to call methods through RMI.

The joining node sent the requests and received the partial shares from the coalition nodes in a serial order. The serial order of requests and replies gave us the worst-case scenario of the working of the algorithms. We could improve the speed by using multithreading and multicasting, but this was not implemented.

5.2 Key and Signature Generation Results

The results of the experiments are given in Fig. 3 and Fig. 4. As expected, the processing time increases as key length increases, and the faster processors perform better than the others. An interesting point to note is that even for a reasonable high key length of 1280 bits, the *process time* is under 10sec for all machines.

The total processing time is the time taken for a node to generate and send the partial key requested by a joining node.

6. DISCUSSION AND FUTURE WORK

In the previous sections, we presented a design for distributed generation of private keys and signatures and showed the implementation results. We now discuss some of the design issues we considered for the proposed (t, N) secure model.

Initialisation of t nodes: In section 3.1 we discussed that at the time of the creation of the ad hoc group, an initialisation program or dealer would initialise some number of nodes in the group. This is a common assumption in secret sharing[5, 8-10, 13]. One possible future direction is on how to initialise t or more nodes in a distributed fashion. Distributed initialisation can also lead to better pro-active security for the model.

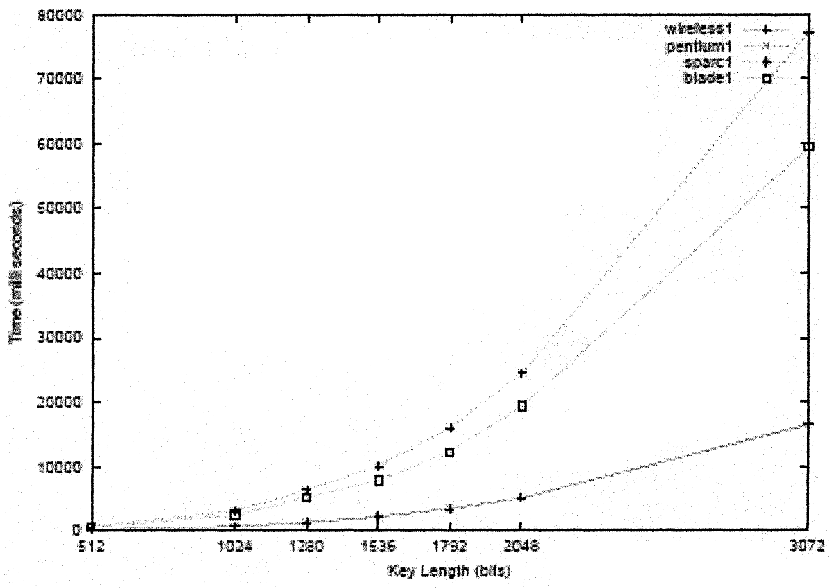

Figure 3. Authentication times for different machines (Part 1)

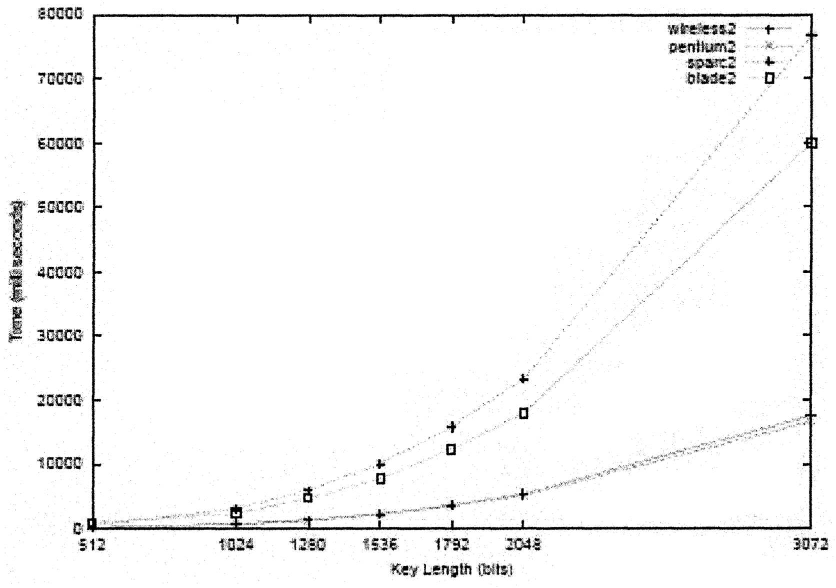

Figure 4. Authentication times for different machines (Part 2)

Parameter t: We require that a node have at least *t* valid nodes in its neighbourhood to get a private key. This is a very critical assumption as an adversary needs to compromise *t* nodes to break the security of the group. In addition, the public key of the group also becomes larger as *t* increases. Thus, the parameter *t* determines the availability, security and computational complexity of the model. A larger *t* value would lead to less availability as a node would need more authenticator nodes in its neighbourhood, but at the same time the security of the system will increase as an adversary will need to compromise more nodes. At the same time a node would need more computations to encrypt messages and verify signatures as the public key becomes larger. A small *t* value will have the inverse effect that is more availability, less computational complexity and a less secure network. We note that for practical purposes, the parameter *t* would be reasonably small (less than ten) such that the size of the public key would not present a large computational burden. The *t* value is fixed at initialisation time and should provide good security and availability of the nodes in the group. As discussed before in this section, if we can initialise the nodes in a distributed fashion, we can change the parameter *t* to suit the security and availability requirements of the group.

Renewal of Private Keys: We require that nodes in the group update their private keys at regular intervals. The interval is an important factor, as a short interval means that nodes need to frequently update their keys. This would lead to more network traffic but would lead to better security, as faulty and compromised nodes would be unable to update their keys, provided that some of the neighbouring nodes know about it. On the other hand, a longer interval would lead to less network traffic but at the cost of greater security risks. The interval period is predefined by the group founder (initialisation dealer) and cannot be changed by the nodes. More work is needed to dynamically adjust the interval period according to the network traffic and security risks.

Network Failures: We assume limited reliability in the group messaging service. We do not require all nodes be contactable at the same time, some may be out of range, as is often the case in mobile networks. We expect, however, that most of the messages, e.g. those about node exclusion will reach their destinations eventually. Network partitions do not cause a problem as long as each partition has at least the number of nodes required by the scheme. The only requirement is that the threshold number of nodes needs to be available to admit a new member or refresh private keys. If the number of available nodes is less, the operation has to be postponed.

7. CONCLUSIONS

In this paper we proposed an architecture that generates individual private keys for a common public key dynamically, in a distributed manner. We combined the benefits of the group key based approaches and the public key based approaches, while we do not need to maintain lists of public keys or a shared secret key.

Our proposed solution is based on the key insulated cryptography and signature schemes, and on threshold cryptography. We modified the key insulated schemes to suit ad hoc networks, and generate and update keys in a distributed fashion. We do not require any base stations or servers to provide keys, except at the initialisation phase.

We also generate a certificate showing that the node has received the key from t legitimate nodes and that it would expire after a predetermined time. The certificate is signed with a secret signing key that is known by none of the nodes but t nodes can produce a signature without revealing the key.

The proposed model is (t,N) secure as an adversary who compromises less than t nodes in the group cannot gain any more information. In addition, t nodes can generate a private key and a certificate for a node signed with a secret signing key. The network traffic is restricted to the local neighbourhood of the nodes and hence the performance is good. The results of our experiments are promising, as even for large key lengths the distributed key and signature generation algorithms work well.

ACKNOWLEDGEMENTS

This research was performed on equipment partially provided by Sun Microsystems grant no. 7832-030217-AUS.

REFERENCE

1. T. Aura and S. Mäki. Towards a survivable security architecture for ad-hoc networks. In *Security Protocols*, volume 2467, pages 63–73. Springer-Verlag Heidelberg, Lecture Notes in Computer Science, 2002.
2. S. Čapkun, L. Buttyán and J.P. Hubaux. Self-organised public-key management for mobile ad hoc networks, *IEEE Transactions on Mobile Computing*, 2(1):52–64, Jan-Mar 2003.
3. Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-insulated public key cryptosystems. In *Advances in Cryptology- EUROCRYPT 2002*, volume 2332. Springer-Verlag Heidelberg, Lecture Notes in Computer Science, April 2002.
4. Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong key-insulated signature schemes. In *Workshop on Public Key Cryptography (PKC)*, volume 2567. Springer-Verlag Heidelberg, Lecture Notes in Computer Science, January 2003.

5. Y. Frankel, P. Gemmell, P. D. MacKenzie, and M.Yung. Proactive RSA. In *Advances in Cryptology - CRYTPTO 1997*, volume 1294, pages 440–454. Springer-Verlag, Heidelberg, Lecture Notes in Computer Science, 1997
6. S. Kaliaperumal. Securing authentication and privacy in ad hoc partitioned networks. In *Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, pages 354–357, Orlando, FL, January 2003.
7. Y. Kim, A. Perrig, and G. Tsudik. Group key agreement efficient in communication. *Computers, IEEE Transactions on*, 53(7):905–921, July 2004.
8. J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for mobile ad hoc networks. In *IEEE 9th International Conference on Network Protocols (ICNP '01)*, 2001.
9. H. Luo and S. Lu. Ubiquitous and robust authentication services for ad hoc wireless networks. *UCLA Computer Science Technical Report*, 200030, October 2000.
10. H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang. Self securing ad hoc networks. In *Proceedings Seventh International Symposium on Computers and Communications (ISCC 2002)*, pages 567–574, July 2002.
11. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
12. K. Obraczka, K. Viswanath, and G. Tsudik. Flooding for reliable multicast in multi-hop ad hoc networks. *Wireless Networks*, 7(6):527–634, November 2001.
13. Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *Network, IEEE*, 13(6):24–30, Nov-Dec 1999.