QoS-based Authentication Scheme for Ad Hoc Wireless Networks

P Muppala, Johnson Thomas
Department of Computer Science
Oklahoma State University, USA
jpt@cs.okstate.edu

Ajith Abraham
School of Computer Science and Engineering
Chung-Ang University, Korea
ajith.abraham@ieee.org

Abstract

In this paper we investigate the selection of an optimal threshold level that takes into account security and quality of service requirements for ad hoc networks. We incorporate intelligence into the selection of an optimal threshold for a distributed threshold cryptography scheme for a distributed trust model in ad hoc networks. We investigate both local and global threshold schemes.

1. Introduction

Ad hoc wireless networks provide ad hoc connectivity among a group of mobile devices without the support of any infrastructure. A fixed centralized trusted authority for authentication purposes is not possible in such a network. Threshold cryptography for authentication has been proposed for developing a distributed certification authority, which does not require a fixed centralized server [1]. The absence of a centralized server in a mobile network affects the quality of service (QoS). In this paper we specifically look at one QoS parameter, namely, authentication delay or the time required for authentication. Authentication performance is based on two factors: threshold level and authentication delay. While a centralized architecture can guarantee the authentication delay, this is not possible in a distributed authentication scheme where nodes are mobile. We investigate how security impacts QoS in a distributed system by looking at local and global schemes for achieving security while maximizing QoS. We propose an intelligent approach to determine the optimum threshold level (OTL) under different conditions.

2. Ad Hoc Network Security

Secure ad hoc networks require authentication which involves validating the identity of the user to the node and the identity of the node to the network [2]. A single centralized authentication server is unsuitable for ad hoc networks and, moreover, an intruder may subject it to single point of attack [3].

Threshold cryptography is a secret sharing technique where a secret is shared over a group of servers [4]. A threshold level is selected, such that the number of secret sharing servers should exceed the threshold level in order to reveal the secret shared by them [5]. This technique works under the assumption that a majority of the servers can be trusted. The selection of the threshold level depends on various factors. This technique selects a threshold level of "k" (k<n), so that k-1 entities cannot collaboratively reveal the secret. It requires at least "k" number of entities to generate the complete secret [6]. Let the secret to be shared is "D". Consider a polynomial of k-1 degree

$Q(x) = a[0] + a[1] *x + a[2]*x^2 + \ldots\ldots + a[k-1]*x^{k-1}$,

such that a[0] = Q(0) = D – the secret to be shared.

D1, D2, D3,….Dn, "n" number of shares are generated such that

D1 = Q(1), D2 = Q(2), D3 = Q(3), …….. Dn = Q(n).

In order to construct D (complete secret) from its shares (Di's) that are shared by "n" number of entities, at least "k" number of Di's are required. Using the coefficients, the secret (a[0]) can be determined [6]. Shamir has proved that polynomial interpolation technique cannot be broken computationally by using k-1 shares [6]. By fixing threshold level (k), the number of Di pieces can be increased. This makes it possible to dynamically add new shareholders for the secret.

A virtual certification authority (VCA) issues a digital certificate for each node certifying that node and its corresponding public key. All the nodes present in the network take part in a threshold secret sharing scheme to construct the VCA. If the threshold level is fixed as "k", then "k" number of nodes collaboratively provides the functionality of an authentication server. This model works on the basic assumption that at any point of time, an intruder cannot break into "k" or more nodes present in the network [1]. A node is trusted if it is certified by at least "k" of its one-hop neighbors. The certificate held by

a node has an expiration time after which it has to be renewed The certificate acquiring process has a time constraint "T_{cert}", the time elapsed between the request and the issuance of the certificate. All the "k" nodes must certify the node within this time. Hence the trust model is distributed both in space (k) and time (T_{cert}). This certificate is valid for "T_{valid}" seconds. When a node requires a new certificate, it sends requests to all of its one-hop neighbors. If at least "k" of these one-hop neighbors sign the requests using their VCA private key shares (SK_i) within T_{cert} seconds, the requesting node can construct a new certificate that remains valid for another T_{valid} seconds.

Each node monitors its neighbor's behavior through a certificate revocation list (CRL), [1]. A node may be marked as convicted if a threshold number of other nodes accuse it. When a node receives a certificate request, it refers to its CRL If the requesting node is in the CRL convicted list, the certificate request is rejected. Otherwise a partial certificate is issued to the requesting node. If the requesting node fails to collect at least k partial certificates within T_{cert} seconds, it fails in its attempt to construct a new certificate.

A very high threshold level ensures greater security, but the QoS requirement may not be satisfied. If the threshold level is lowered, it becomes easy for a node to construct its digital certificate within the QoS requirements (or specified authentication delay time), but the security aspect is compromised. The threshold level selection process is influenced by various network dynamics such as network density, node speed, node transmission range, threshold requirements etc. The objective of our security model is:

$$\max\left[\sum_{i=1}^{M} f(S,Q)\right] \qquad (1)$$

subject to a constraint such as:

- level of security must meet minimum requirement S_{min}.
- for each authentication, the QoS must meet a minimum requirement Q_{min} and the security must meet a minimum requirement S_{min}.

S may be defined as the threshold ratio (that is *k* out of *n* ratio). *Q* is defined as a delay time. Other parameters such may also be added. (1) states that the objective is to obtain the maximum security and QoS for each of *M* authentication requests.

However this optimization problem cannot be solved with standard optimization techniques as the function *f*(*S*,*Q*) is not known. We therefore use simulations to optimize the above function to derive the OTL. We investigate two ways to fix the threshold level.

- Global Selection (GTLSS) – Threshold level is fixed, i.e. it is the same for all nodes at all times.
- Local Selection (LTLSS) – Threshold level is selected based on the local environment of a node at that moment. This method is more responsive to the dynamic nature of a mobile network. We investigate the performance of the authentication scheme resulting from each implementation, that is, GTLSS and LTLSS.

3. Ad hoc Model

The primary goal of this work is to determine OTL. Each node in the network is designed as an abstract two-layer model for simulation purposes. The lower mobility layer uses a random-waypoint mobility pattern [7] as the mobility model in each node. The upper layer implements the authentication scheme.

GTLSS protocol:

1. Certificate-requesting node broadcasts `REQUEST` packets to all of its one-hop neighbors.
2. Neighboring nodes, which receive the `REQUEST` packets, send `REPLY` packets that contain partial certificates for the requesting node.

The requesting node upon receipt of the replies, will try to construct its full certificate using partial certificates. If the number of partial certificates received is equal to or greater than the threshold level, the requesting node can successfully construct the full certificate. The number of replies required to construct the full certificate is the same for all the nodes present in the network. The time taken between the issue of the certificate requests and the construction of the certificate after receiving the certificate replies from its one-hop neighbors is T_{cert}. T_{valid} determines the amount of time the certificate remains valid once it is constructed. Before T_{valid} expires, the node again sends certificate requests for its next certificate construction. Early issuance of certificate requests gives the node time to get its next certificate before the old one expires.

In the LTLSS scheme the number of partial certificates required to construct a full certificate is determined dynamically based on the number of one-hop neighbors and the threshold level. Each node has a set of keys each corresponding to a k-1 degree polynomial for different values of k.

LTLSS Protocol

1. The certificate-requesting node sends `HELLO` messages to all its one-hop neighbors by broadcasting.
2. Upon receiving `HELLO` messages, neighbors reply with a `HELLO_REPLY` message. This helps determine the number of active one-hop neighbors (local environment) that are ready to issue partial certificates.

Based on this, the number of partial certificates required to construct the full certificate is computed.

3. Certificate requesting node constructs a reply node list that has the identity of all the nodes that sent HELLO_REPLY message. A certificate REQUEST message is constructed that includes the reply node list and this is broadcasted to all its neighbors.

4. Neighbors upon receiving REQUEST messages check the number of nodes present in the reply node list and select a corresponding key and this key is used to sign a partial certificate. This partial certificate is transmitted back to the requesting node in a REPLY message. An extruder is prevented from sending certificate request messages with false k value.

5. The requesting-node tries to construct a full certificate using these partial certificates. If the number of partial certificates is equal to or greater than the required number (computed in step 2), the node is successful in its attempt to obtain the certificate.

LTLSS adds overhead at both the requesting node and replying nodes. T_{cert} here includes the time taken for the hello requests and replies along with the certificate requests and replies. The performance factor g for the authentication protocol gives % number of successes in certificate constructions. This is a measure of the satisfied QoS, in other words, it shows the number of certificates successfully obtained within the required QoS or delay time T_{cert}. T_{cert} is the Q_{min} parameter in eq.(1) where $Q_{min} = 1/ T_{cert.}$ Performance g = (Total Number of certificates successfully constructed/Total number of certificate requests) *100. If the node is able to construct the certificate before the T_{cert} timer expires, it is termed as a successful certificate construction, in other words, it satisfies the second constraint in eq. (1). If the node fails in its attempt to construct the certificate within T_{cert} time, it repeats the certificate acquisition process all over again. If the node still could not acquire a new certificate by the time T_{valid} expires, it is termed as certificate acquisition failure.

4. Results

4.1 GLTSS Performance

T_{cert} is set to 8s and T_{valid} to 1min for the authentication protocol. The performance is noted at different threshold levels with an interval of 5% between them. Fig. 1 shows that the authentication protocol's performance is satisfactory at low threshold levels but as the threshold level increases, the performance degrades, that is, at high security threshold, the QoS decreases, as indicated by the number of failures for constructing full certificates. The main reason behind the dismal performance of GTLSS is due to the uneven distribution of nodes in the simulation region. Since the number of partial certificates required to construct a new certificate by a node are the same for all the nodes present in the network, nodes located in thinly populated regions fail in their attempt to acquire the required number of partial certificates.

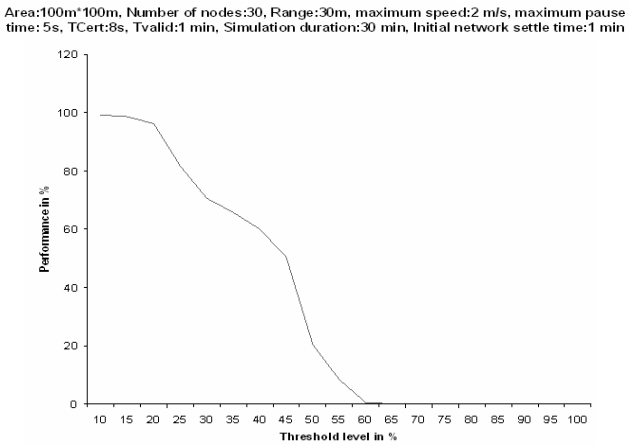

Figure 1: GTLSS Performance

Performance can be improved by increasing the communication range of the nodes. But this is not a practical solution, since the nodes have limited power resources. One possible solution is to increase time T_{cert} so that the nodes are given additional time to construct their certificates (that is, QoS requirements are relaxed).

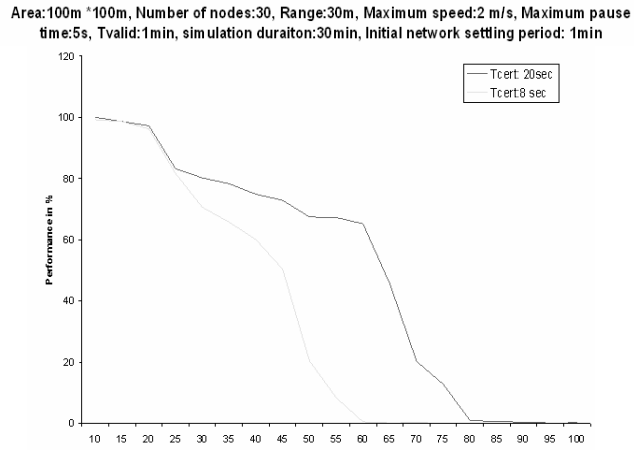

Figure 2: Performance of GTLSS with increased T_{cert}

Fig. 2 shows the performance of GTLSS with T_{cert} time period increased to 20 seconds from 8 seconds. There is a significant improvement in the performance of the authentication protocol since nodes are given additional

time to collect partial certificates from neighbors, that is, Q_{min} value is increased; but there is a penalty as nodes have to wait longer to construct the certificate. The biggest drawback is that the number of partial certificates required to construct the full certificate is fixed for all the nodes in the network. This results in failure to construct certificates as the QoS requirements cannot be met. The network traffic increases steeply as a result of the higher number of certificate construction failures. This could result in network congestion.

4.2 LTLSS Performance

Fig. 3 shows that the performance remains at almost 100% for threshold levels up to 85%, thereafter the performance curve dips drastically. This performance is attributed mainly due to the dynamic nature of the authentication protocol. Based on the local density of a node, the number of partial certificates required is determined. In most of the cases the node is successful in its attempt to construct the certificate. At high threshold levels, performance degrades drastically.

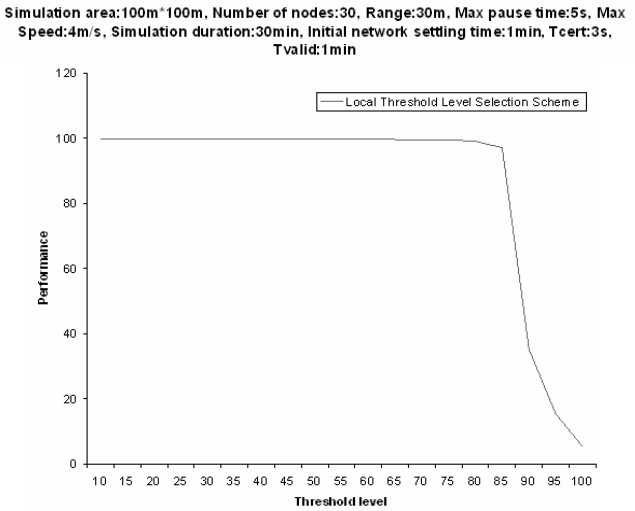

Figure 3: Performance of LTLSS

Simulation analysis of different network situations for LTLSS show that the QoS requirements are met most of the time as the curve remains at high 90's for most of the threshold levels, but then drops at some particular point. QoS and threshold levels are complementary to each other and it is desired to have a maximum threshold level where the QoS requirement is satisfied. This maximum threshold value which satisfies QoS requirements is called the critical point. From the LTLSS analysis the critical threshold value can be set a value slightly less than the point where the curve starts to drop (that is when the QoS requirement is not satisfied).

4.2.1 Analysis for Nodes at High Speeds The QoS (or T_{cert}) is dependent upon a number of factors as listed earlier. It is beyond the scope of this paper to discuss all these factors, but we investigate the impact of node speed on the performance of authentication. Speed affect QoS which impacts authentication performance. Fig 4 shows the performance of the authentication protocol for the LTLSS scheme if the nodes are moving at high speeds. The performance of the authentication protocol decreases as the maximum speed of the nodes is increased. The reason behind this is that the protocol has four steps whereas GTLSS has only two steps. Due to the high speed of the nodes, there is a greater chance that a node moves out of range from its neighbors before completing all four steps.

A node gets two types of request packets in its incoming queue. They are HELLO requests and certificate REQUESTs. Equal precedence is given to both types of packets. But, this delays a node from sending certificate REPLYs and thereby there is a greater chance that the replies might not reach the requesting node. To satisfy the T_{cert} (or QoS) requirement, precedence can be given to process certificate requests over hello requests. This enables a node to send certificate replies faster than in normal FIFO queue without precedence. Thereby the requesting node can collect its partial certificates faster before it moves out of range from the sender.

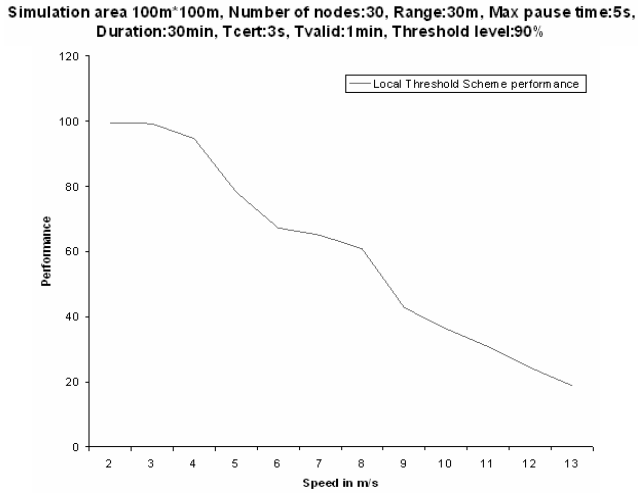

Figure 4: Performance with increase in speed

Each node checks its incoming request queue for a certificate request; if there is any, the first such request in the queue is processed first. If there are no certificate requests in the queue, then the first hello request is

processed as done normally without any precedence. By delaying the processing of hello requests, the number of required partial certificates might decrease which could relax the security aspect of the authentication protocol. Fig 5 gives the comparison between the performance of the authentication protocol before and after setting precedence to certificate requests. But the threshold level is increased from 80% to 90% when precedence is introduced in order to bolster the security of the nodes. At lower speeds, the performance without any precedence is higher than that with precedence. But as the speed of the nodes increases, the authentication scheme with precedence does not drop much in its performance as compared with the other curve.

The required number of partial certificates is determined based on the locality of a node. Moreover, it is easier to select the critical threshold value for a given network. However, due to more number of steps involved in the protocol, performance of the protocol drops down for nodes that move at higher speeds. But this can be overcome by setting precedence level to certificate request packets.

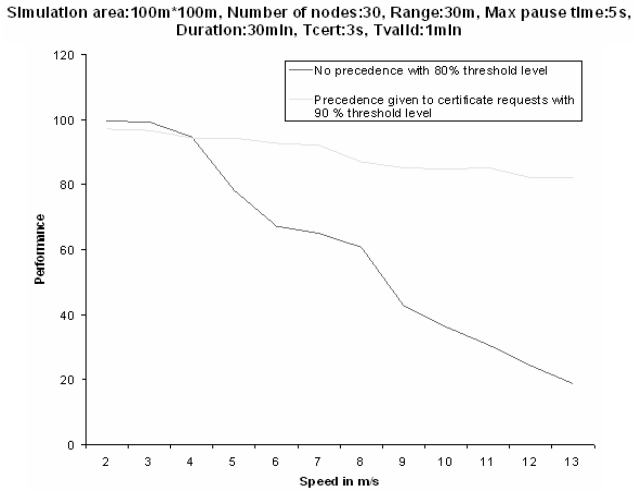

Figure 5: Performance comparison after adding precedence

5. Neural Networks for OTL

Using intelligent techniques, the simulation results obtained in the previous section can be used to predict the OTL for a new set of network and authentication protocol parameters. The artificial neural network (ANN) methodology enables us to design useful nonlinear systems accepting large numbers of inputs, with the design based solely on instances of input-output relationships.

One of the popular training algorithms for neural networks is the Scaled Conjugate Gradient Algorithm (SCGA). Moller [8] introduced it as a way of avoiding the complicated line search procedure of conventional conjugate gradient algorithm (CGA). According to SCGA, the Hessian matrix is approximated by

$$E''(w_k)p_k = \frac{E'(w_k + \sigma_k p_k) - E'(w_k)}{\sigma_k} + \lambda_k p_k \quad (2)$$

where E' and E'' are the first and second derivative information of global error function $E(w_k)$. The other terms p_k, σ_k and λ_k represent the weights, search direction, parameter controlling the change in weight for second derivative approximation and parameter for regulating the indefiniteness of the Hessian. In order to get a good quadratic approximation of E, a mechanism to raise and lower λ_k is needed when the Hessian is positive definite. Detailed step-by-step description can be found in [8]

When the performance function has the form of a sum of squares, then the Hessian matrix (H) can be approximated to $H = J^T J$; and the gradient can be computed as $g = J^T e$, where J is the Jacobian matrix, which contains first derivatives of the network errors with respect to the weights, and e is a vector of network errors. The Jacobian matrix can be computed through a standard backpropagation technique that is less complex than computing the Hessian matrix. The Levenberg-Marquardt (LM) algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e \quad (3)$$

When the scalar μ is zero, this is just Newton's method, using the approximate Hessian matrix. When μ is large, this becomes gradient descent with a small step size. As Newton's method is more accurate, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. By doing this, the performance function will always be reduced in each iteration of the algorithm.

5.1 Experiments

In order to train the neural network, a training data size of 1430 samples is obtained from the simulation runs described in sections 3 and 4. Each sample has a combination of input parameters – density, T_{cert}, transmission range, maximum speed of the nodes in the random waypoint mobility model and its corresponding OTL. OTL is picked by choosing a threshold level where the sum of authentication protocol's performance and

threshold level is the maximum (with a low threshold for security as the constraint).

$$OTL = \text{Max}\,(\text{Threshold level}_i + \text{Performance}_i)_{\,i=1\ldots100}$$

For our experiments we gave the same priority values (*SPI* = 1 = *QPI*) (see eq (1.1)) In other words, equal importance is given to both QoS performance and security (threshold level). Depending on the applications, these priorities may be adjusted. To keep things simple, equal priority was given to security and QoS performance to obtain the training data.

Feedforward neural network is chosen among the different types of neural networks available. Two layers of neurons are taken for each neural network, with just one neuron in the output layer since there is only one output parameter –threshold level. We used the SCGA and LM algorithms to train the networks. Tangent-sigmoidal activation functions were used. Mean Squared Error (MSE) is used as performance function. RMSE and Correlation Coefficient were used as performance metrics that are used to compare the results obtained from testing the neural network and their corresponding results obtained from our simulation. In order to train the neural network, training data is constructed using different sample sizes picked randomly from the master training data obtained from simulation runs.

5.2 System Architecture

A trained neural network can be embedded into each node, so that nodes can compute an OTL for different network conditions and use it in the authentication protocol. This automated process helps a node in fixing the threshold level faster and accordingly adjusts the authentication protocol. There are three functional blocks present in each node that cooperate with each other to perform the authentication protocol. The security protocol block issues `HELLO` requests when T_{valid} is about to expire. The security protocol block then collects `HELLO_REPLY` messages. The replies are sent to the Neural network block which uses the threshold scheme to determine the number of partial certificates required based on the the number of `HELLO_REPLY` messages obtained. Certificate constructor uses this information from the neural networks block to issue certificate `REQUEST`s and it constructs the complete certificate from certificate `REPLY` messages.

Three such training data sets are formed by picking 50%, 75% and 90% of samples from the master data set randomly. Number of neurons that constitute the first layer of the neural network is taken between 25 and 45 in intervals of 5. Different neural networks are constructed, each with a different combination of training function and number of neurons present in the first layer. These neural networks are trained with the three training data samples. After training them, they are tested by picking 10% data samples from the remaining simulation results randomly. RMSE and CC values were obtained during testing and best test results were obtained with 40 hidden neurons trained using SCGA.

6. Conclusions

The problem of determining an optimal threshold level for ad hoc networks such that security and QoS requirements are satisfied is investigated in this paper. We have only looked at one QoS parameter, namely authentication delay. This research shows that LTLSS performs better than the global scheme. We have proposed an intelligent approach to determine OTL given a network configuration. Future work will include analysing the authentication scheme in the presence of intruders. Association rules can also be investigated as an alternative to neural networks to train the network for OTL. Complete and rigorous optimization of can only be achieved after obtaining an understanding of the relationship between QoS and security.

References

[1] H Luo, P Zerfos, J Kong, S Lu, L Zhang, "Self-securing ad hoc wireless networks", *Proc. 7th Int. Symp. Computers and Communications*, pp. 567-574, ISCC, July, 2002

[2] William Stallings, *Cryptography and network security: principles and practice*, Prentice Hall, 2003

[3] Lidong Zhou, Haas, Z.J., "Securing ad hoc networks", *Network, IEEE*, Volume: 13, Issue: 6, Nov.-Dec, 1999

[4] Helger Lipmaa, "Threshold Cryptography", http://www.tcs.hut.fi/Studies/T-79.159/slides/L8.pdf, 2003

[5] E. Okamoto, G. Davida, and M. Mambo, "Some Recent Research aspect of Threshold Cryptography", *Proc. Information Security*, pp. 158-173, September, 1997

[6] Nick Szabo, "Shamir's secret sharing", http://szabo.best.vwh.net/secret.html, 1997

[7] T Camp, J Boleng, V Davies, "A survey of mobility models for ad hoc network research", Dept. of Math. and Computer Science, Colorado School of Mines, Sept. 2002.

[8] A F Moller, "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning", *Neural Networks*, Volume (6), pp. 525-533, 1993.