Structuring Proactive Secret Sharing in Mobile Ad-hoc Networks

Hitoshi Asaeda*, Musfiq Rahman†, Yoshihiro Toyama*
*Keio University, Graduate School of Media and Governance, 5322, Endo, Fujisawa-shi, Kanagawa, Japan
†Asian Institute of Technology, P.O. Box 4, Klong Luang, Pathumthani, Thailand

***Abstract*— Threshold cryptography is a novel cryptographic technique sharing secret among members. It divides a secret key into multiple shares by a cryptographic operation. This technique is useful to provide a shared secret key to legitimate nodes in a infrastructure-less mobile ad-hoc network (MANET). As an additional component, Proactive Secret Sharing (PSS) allows a set of nodes holding shares to refresh all shares by generating a new set of shares for the same secret key from the old shares without reconstructing the secret key. It is necessary to reasonably escape from threats of exposing the secret key when threshold cryptography is used.**

In this paper, we design the PSS protocol implementation specification in a MANET environment. In our scheme all share holder nodes synchronize the PSS procedure in a well-managed fashion to keep the protocol consistency. We then introduce our actual implementation and evaluate the behavior and its performance criteria.

I. Introduction

A mobile ad-hoc network (MANET) architecture enables mobile nodes to instantaneous group communication immediately and easily. In this network, each mobile node creates communication path with neighbor mobile nodes in the same radio range and sends data to a destination node directly or through these neighbor nodes. In this manner a data forwarding path is established with arbitrary nodes in a MANET for forwarding or receiving data. According to its property, it is desirable for legitimate nodes to establish secure channels for their communication in order to avoid threats that malicious nodes eavesdrop or tamper the data. However, lack of fixed infrastructure or centralized administration may lead its difficulties; what is more critical and complex is that a secret group key that is used to encrypt data cannot be easily and securely distributed to these legitimate nodes.

A node distinction could be made at the application layer where access to a service or participation to its collaborative support is allowed only to the group members. In order to establish secure communication with them, use of a shared group key for message encryption and decryption fulfills the requirement. Yet another concern relates to the group key sharing mechanism; once a mobile node is authenticated as a legitimate group member, at next the corresponding secret group key must be given securely to the new member.

It is plausible to assume that the key can be distributed by establishing a transient secure channel. Unfortunately, it is a difficult task and not necessarily suitable for group communication in a MANET environment that is comprised of many mobile nodes, especially when we consider the possibility that an end node of a secure channel may be out of range during or after successful setup of a secure channel.

As the potential solution, *threshold cryptography* [2] provides a beneficial approach. In the (n, t) threshold cryptography, a secret group key – a shared key – is divided into n shares and kept by n legitimate nodes, which we call share holders. Later, a new node collects t shares from the response of t nodes (among n nodes) and generates the original secret key as a legitimate node. Since this scheme does not require transmitting a secret key itself to mobile nodes, it increases the secure level. Furthermore it works even with a busy MANET where network topology and mobile nodes are dynamically changed, if we can assume that t or more share holders reside in the network.

In this scheme, the fundamental requirement is that each share must not be disclosed within the share transmission procedure; if t or more shares are stolen by malicious nodes within a long span of time, the secret key is finally generated by them. Here *Proactive Secret Sharing (PSS)* [3][4] reasonably provides the way to escape from threats of exposing the secret key. PSS allows to refresh all shares by generating a new set of shares for the same secret key from the old shares without reconstructing the secret key, and then the old share is useless after the refresh of each share.

PSS is therefore a necessary component of a key management protocol using threshold cryptography. To keep a protocol consistency, all share holders must cooperate with the PSS procedure in a well-managed fashion. In fact, while there are several references that have applied the key sharing technique using threshold cryptography [5][6][7], important properties resided in the PSS procedure have not been detailed therein.

In this paper, we propose a PSS synchronization procedure and its actual implementation. To evaluate its behavior, we measure the performance on top of our wireless testbed. Our main contribution in this paper is to define the structure of PSS implementation that relates to threshold cryptography; yet considering the implementation specification of threshold cryptography, including methods how the secret group key itself is bootstrapped (i.e. initially generated) and how n and t for (n, t) threshold cryptography are defined, is out of scope of this paper.

The remainder of this paper is organized as follows: In Section II we investigate and analyze *threshold cryptography* and *Proactive Secret Sharing (PSS)* techniques. In Section III

0-7803-9410-0/06/$20.00

we detail the structure of the PSS synchronization procedure and its actual implementation. The protocol evaluation and performance measurement are discussed in Section IV, and then Section V concludes this paper with outlining future work.

II. Overview

A. Threshold Cryptography

Threshold cryptography is a novel cryptographic technique sharing secret among nodes. The idea comes from Shamir's discussion about the company's secret key [1]. An (n, t) threshold cryptography scheme (where $n \geq t$) allows n nodes to share the ability to perform a cryptographic operation, so that any t nodes can perform this operation jointly, whereas it is infeasible for at most $t-1$ nodes to do so, even by collusion.

Formally, a $t-1$-degree polynomial is constructed such that the constant coefficient (i.e., S) is the secret and all other coefficients are random elements:

$$y = f(x) = a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + \cdots + a_1 x + S \quad (1)$$

In this case, each of the n shares is a pair of (x_i, y_i) of numbers such that $f(x_i) = y_i$ where $i \in \{1 \ldots n\}$, $x_i \neq 0$. Now given any t shares, the polynomial is uniquely determined and hence the secret S can be computed by using Lagrange interpolation.

In summary, threshold cryptography is a useful technique for sharing and distributing a secret group key with multiple mobile nodes in a MANET, because (1) it does not require any key infrastructure, (2) it avoids the single point of failure for the key distribution, and (3) it works even with a busy network when we assume at least t nodes having corresponding shares reside in the same network.

As a concrete key management system in a MANET environment, one secret group key and its $t-1$-degree polynomial can be defined with an (n, t) threshold cryptography scheme. Distribution of trust in the system is accomplished with allowing n share holders to share the ability to perform a cryptographic operation. When a new mobile node wants to join a secure group communication, the new node asks t or more share holders nodes to the group to give their shares. Each share is then distributed to the new node through a secure link established by their own public/private key pairs [8] whose services could be guaranteed by distributing the Certified Authority (CA) functionality to each share holder as defined in [9] or other mechanism. The new node then constructs the secret group key without knowing the pre-used polynomial or any other information.

Such key management service actually works under several assumptions, which require additional components to be more appropriate for MANETs. For instance, a secret key and its polynomial must be securely initialized by some manner, and each share must be initially distributed to n share holders securely. After these preparation, a new node must be authenticated by some mechanism before each share holder distributes own share to the node. We leave such details and imprinting behavior to a separate paper currently in preparation.

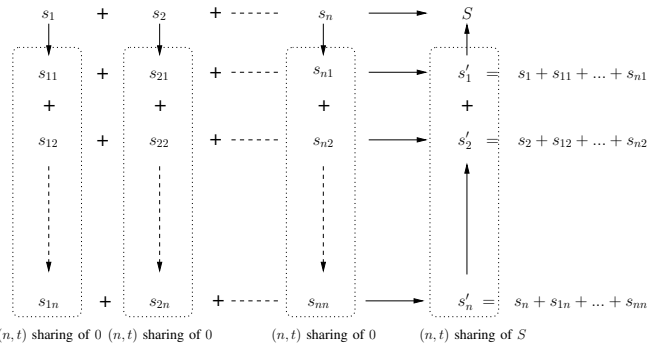

Fig. 1. Mechanism to refresh sub-shares in an (n, t) sharing of S.

B. Proactive Secret Sharing

Threshold cryptography provides the way to convey a shared key to a node without the aid of key infrastructure and is suitable for a secret key sharing in a MANET. However, given t or more shares in an (n, t) threshold cryptography scheme, the secret S can be found. Without the share refresh and with infinite time span it is not hard for malicious nodes to compromise at least t share holder nodes and finally obtain the secret key.

To make each share refresh without disclosure of any share or a secret key itself, Proactive Secret Sharing (PSS) [3][4] can be employed with threshold cryptography as an additional component. It allows to refresh all shares by generating a new set of shares for the same secret key from the old shares without reconstructing the secret key.

In the PSS implementation, each share holder randomly generates own sub-shares (e.g., $(s_{i1}, s_{i2}, \ldots, s_{in})$ on node i), and each sub-share is mutually exchanged to refresh own share. More precisely, the PSS procedure shown in Fig.1 can be performed in the following steps:

1) Let $(s_1, s_2, \ldots, s_n)$ be an (n, t) sharing of the secret key S of the service, with node i having s_i.
2) Node i ($i \in \{1 \ldots n\}$) randomly generates s_i's sub-shares $(s_{i1}, s_{i2}, \ldots, s_{in})$ for an (n, t) sharing of 0.
3) Every sub-share s_{ij} ($j \in \{1 \ldots n\}$) is distributed to node j through secure link.
4) When node j gets the sub-shares $(s_{1j}, s_{2j}, \ldots, s_{nj})$, it computes a new share from these sub-shares and its old share with an equation:

$$s'_j = s_j + \sum_{i=1}^{n} s_{ij} \quad (2)$$

5) Now each share $(s'_1, s'_2, \ldots, s'_n)$ is an (n, t) sharing of the secret key S, because $\sum_{j=1}^{n} s_{ij} = 0$, $\forall i \in \{1 \ldots n\}$.

After each PSS, all the shares will be changed, so that old shares become useless. In such case, since it is impossible to obtain new share from old share, a malicious node must collect at least t shares during the time between two executions of the PSS, which obviously makes his job more difficult.

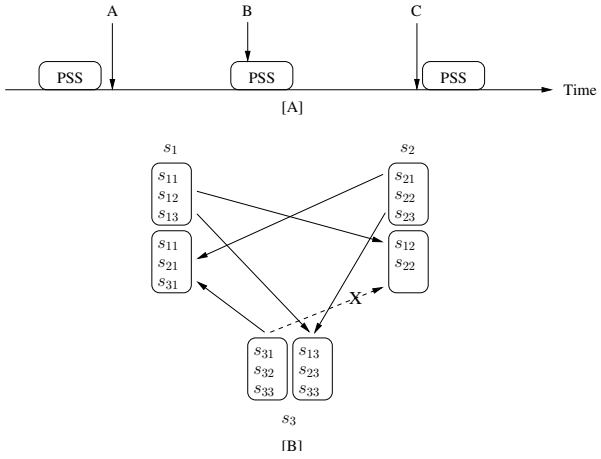

Fig. 2. PSS and timing issue.

III. PSS Implementation Structure

A. Objective

While existing PSS proposals prove its literature and mechanism, lack of its concrete implementation does not promote its functionality to the real communication environment. This may cause the situation that PSS is not widely used in a MANET in spite of its advantages.

PSS is a periodic or on-demand protocol; any active share holder in the network can trigger the PSS protocol at any time. Here, objective exists in this condition. If an arbitrary share holder starts a PSS procedure without any timing coordination with other share holders, or if each share holder tries to refresh its own share using some concurrent scheme, where all share holders just perform communication and computation in parallel, protocol inconsistency may happen, and a new node may receive inconsistent shares and then cannot generate the secret key.

Let us see Fig.2 [A] showing periodic PSS procedures. In an (n, t) sharing of S, if a new node requests shares and can get t shares between the last and the next PSS procedures (such as A in this figure), the node will generate S without any trouble. However, if a new node requests to get shares during PSS (e.g. B), share holders should make the node wait for their PSS and give their new shares afterward or suspend their PSS and give their current shares. Or, if a new node tries to get shares just before PSS is started (e.g. C), each share holder should put the PSS on hold until the node successfully get current t shares. These matters imply that a share holder needs to know *when* a new share can be used as own share and *when* the new share can be given to a new node.

As well as such timing issue, there is another concern related to untraceable sub-share transmission. The considerable story is, while a share holder can understand whether it has gotten sub-shares generated by other share holders successfully, it does not know other share holders also have gotten necessary sub-shares successfully. Or, if a share holder does not get sub-shares from corresponding share holders due to loss of connectivity with them, it should not keep waiting for the reply from the share holders for a long span of time. This situation will be especially happened in a MANET environment in which mobile nodes may leave from the network or turn off itself without any notification.

Let us see Fig.2 [B]. Three share holders (s_1, s_2, and s_3) start their PSS procedure and exchange each sub-share. For instance, because of the link failure, s_3 cannot give its sub-share to s_2. The worst phenomenon here is s_1 does not know that s_2 and s_3 had the share transmission problem and hence may start giving its new share to a new node, which is an inconsistent share to other old shares. In this case, share holders should keep own old shares until all share holders complete their PSS (for a short period) or change the (n, t) sharing to an $(n-1, t-1)$ sharing (with eliminating the left share holder) along their pre-determined policy.

One may deduce that simply all share holders may be able to broadcast own sub-shares to other share holders and count the number of received sub-shares to know that all share holders successfully synchronize the PSS process, instead of implementing PSS iteratively. However, sub-share transmission must be secure; if a mobile node in the same MANET receives sub-shares via broadcast, it can generate a new share from the corresponding old share and maintain the new share that should not be owned by the node. This will contradict our approach, and therefore it is necessary to securely transmit sub-shares only to the corresponding share holder.

B. Specification

According to the consideration aforementioned, we need to have some mechanism that informs the status that all share holders have finished each share refresh and the new ones are now ready to use. To that purpose, special notation called *token* is introduced for triggering the PSS procedure in our protocol. Using *token* is straightforward manner to control the timing of the procedure; only one share holder in the network can have *token* and the PSS protocol is started by the *token holder*. Since it is assumed all share holders are the members of the group and hence have the corresponding secret group key S, *token* can be given by $H(S)$, where H is 160 bits of the SHA-1 [10] hash.

Upon triggering a PSS procedure, *token holder* exchanges own sub-shares with other $(n-1)$ share holders. After *token holder* computes its new share from its old share and received sub-shares, it sends a `PSS_REQUEST` message to all share holders in order to notify the PSS procedure is started and passes the *token* to the next share holder. (For these procedures, we assume each share holder has maintained a list of IP addresses of all share holders with the same order.) Likewise *token* passes through the $(n-1)$-th share holder and finally reaches to the n-th share holder (i.e. the last share holder). After this n-th share holder updates its share, it sends the *token* to the original *token holder*. Since the original *token holder* can recognize all share holders have finished refreshing shares at that time, it sends a `PSS_DONE` message to all share holders to indicate the completion of PSS. These share holders can

then discard old shares and start using new shares. The share holder that sends the PSS_DONE message keeps the mission to work as *token holder* again and will later start the next PSS procedure.

When *token holder* does not exist in a network due to an initial phase or its departure from the network, or when a share holder that does not hold the *token* needs to refresh the share (due to some security reason etc.), one of arbitrary share holders will spontaneously send a PSS_REQUEST message to all share holders to become new *token holder* and then trigger a new PSS procedure. If some share holder disappears from the network during or before a PSS procedure, that node must be skipped from the procedure after pre-determined time out. In this situation, active share holders may keep using own old shares or may change the (n, t) sharing to an $(n-1, t-1)$ sharing by eliminating non-existing share holder and regenerate their new shares, whereas it depends on a pre-determined policy in the group and this paper does not discuss such details.

For the timing matter shown in Fig.2, if some share holders receive a share obtainment request from a new mobile node during their PSS procedure – in the time period that the share holders have received a PSS_REQUEST message but have not received a PSS_DONE message yet – or just before their PSS procedure, the share holders offer to suspend the PSS procedure by sending a PSS_SUSPEND message to all share holders. When *token holder* receives this message, it stops forwarding *token* and the all share holders wait for the new node to receive shares. After the reasonable time period, the *token holder* sends a PSS_REQUEST message to share holders to notify they resume the previous PSS procedure.

In fact, our current protocol does not assume that a share holder traces when a new node completes its share obtainment, yet some intelligent mechanism to do it may be needed in the future.

C. Message Transmission

As well as share transmission to a new node, sub-share and *token* transmission must be in secure. For this matter, since we assumed that all share holders have certified other share holders' public keys [8] prior to share or sub-share transmission and *token* distribution, these data can be encrypted with the corresponding public key of the share holder and transferred to each share holder.

Regarding a PSS_DONE message, a PSS_REQUEST message and a PSS_SUSPEND message, they should be also securely distributed to all share holders. Unlike share or sub-share transmission, they can be encrypted by the secret group key S because these messages are distributed to multiple share holders, and therefore using multicast would be the valuable solution. Yet there is one concern of this model; several group members whose group keys are different possibly coexist in a same MANET and may run independent PSS procedures with different nodes in the network. In this case, differentiating the destination multicast address of these messages for each PSS procedure would filter out unneeded messages on share holder

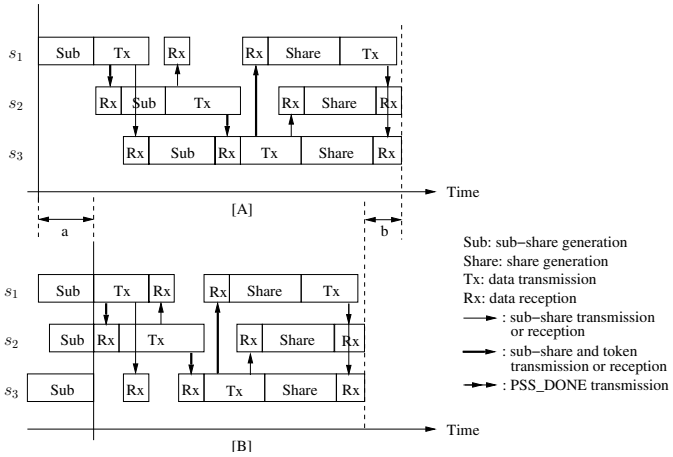

Fig. 3. Timing chart to exchange sub-shares and token among three share holders.

side. Our protocol therefore arranges the multicast address for each message as with the following rule:

$\{$*multicast address prefix*$\}$ | $\{$*leftmost* $H(S)\}$

where a *multicast address prefix* comes from a tentative 16 bits (for IPv4) or 64 bits (for IPv6) address prefix prepared for our PSS protocol and *leftmost* $H(S)$ indicates part of *token*, which is the leftmost 16 bits (for IPv4) or 64 bits (for IPv6) of SHA-1 hash value of the group key (S). Here, "|" is an operation to concatenate left and right strings. Thanks to this manner, we can relatively reduce the possibility of multicast address duplication of PSS procedures; share holders listening on the same multicast group address are usually the same group member and expect that given PSS_DONE, PSS_REQUEST and PSS_SUSPEND messages are sent from the corresponding share holders. Note that, even if the multicast address of these messages is duplicated with different PSS procedures, each share holder can finally choose appropriate messages by checking the *token* inside the messages.

IV. Performance Evaluation

Our PSS protocol is implemented in Java, using Java 2 SDK 1.4.2_08. We have adopted Triple DES (24 bytes) as the cryptographic function with embedded Java class, `javax.crypto.KeyGenerator`.

We have made performance measurement tests to evaluate our PSS implementation. All tests have been done on our testbed in which multiple Linux laptops are connected through a dedicated wireless network in which all mobile nodes communicate directly (i.e. without a MANET routing protocol). In this experiment we have obtained the average time for completing the PSS procedure on all share holders in a MANET. The test results indicate the performance of generating each sub-share and share, but do not include data communication costs, which may be increased by packet loss or other issues in a real wireless MANET environment.

Our protocol uses *token* in terms of synchronizing the PSS procedure with all share holders. This protocol controls the timing of share refresh by keeping all share holders informed

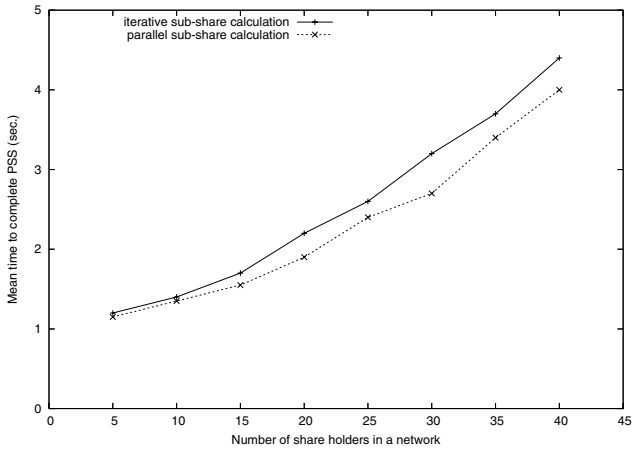

Fig. 4. Average time to complete a PSS procedure.

about each other and avoids inconsistency of share distribution to a new mobile node. While each sub-share is transferred by *token holder* in this manner, it can be calculated by each share holder prior to the sub-share distribution. Fig.3 shows the situation that three share holders (s_1, s_2 and s_3) make a PSS procedure with using *token*. [A] shows the straightforward manner: *token holder* starts a PSS procedure and each share computes own share upon reception of *token*. Instead, in [B] sub-shares are preliminarily prepared on each share holder before PSS is triggered, and PSS is compiled only with sub-share transmission. Comparing with Fig.3 [A] and [B], the time difference given by $(a + b)$ may indicate some effectiveness due to parallel sub-share calculation. We therefore prepared additional implementation that provides parallel sub-share calculation and measured the performance of the implementation in order to know how it reduces the total time to complete PSS.

Now we show the required items every share holder must hold prior to the PSS procedure:

- Corresponding share (s_i), which is not owned by other share holders
- IP address list of all share holders, which are sorted in ascending order
- Other share holders' public keys to encrypt sub-share transmission

Additionally, one of share holders behaves as *token holder*, and it has *token*, which is given by hash value of S, while it is not a problem even if *token holder* does not exist in the network (see Section III-B).

Our measurement results are shown in Fig.4. According to this performance graph, the time difference between "iterative sub-share calculation" and "parallel sub-share calculation" is small. The former result shows the time of sub-share calculation that are given by step-by-step manner such as Fig.3 [A], and the latter one shows the time that each share holder preliminarily generates own sub-shares instead of waiting for the sub-share transmission in a parallel fashion such as Fig.3 [B]. From this observation, the cost of sub-share calculation is much smaller than the delay of sub-share transmission over TCP in a wireless network.

The time that it takes for the nodes to synchronize with our protocol scheme, increases linearly in the number of share holders. This fact actually does not imply an optimistic situation; a desirable property of a synchronization algorithm would be that all share holders become aware of all others in a constant amount of time, as opposed to in $O(n)$ as here.

As another discussion, although PSS makes the share distribution securely, more than enough iteration of PSS procedures will simply overload share holders and consume their precious power. Therefore refining a well-managed timing to trigger a PSS procedure is important criteria, whereas this consideration would be in our future work.

V. Conclusion and Future Work

In this paper, we have proposed a PSS synchronization structure used with a secret key sharing mechanism on top of threshold cryptography in a MAMET environment. We have explained our actual PSS implementation, which uses *token* to synchronize the PSS procedure among all share holders.

To know whether performance advantage can be given by sub-share generation in a parallel fashion, we have also examined the implementation which makes preliminary sub-share generation instead of waiting for the sub-share transmission. In terms of performance measurement of our PSS implementations, we prepared a dedicated wireless network and ran the PSS protocol over the network.

According to the test results, while the PSS procedure can be well organized with feasible sense, it may have some performance impact when the number of share holders is accordingly increased. To the best of our knowledge, our PSS protocol is a novel proposition with no other protocols to compare to, and hence our next step would be observing the performance impact over a real MANET routing protocol like AODV [11].

It is necessary to complete a secret key sharing mechanism with additional procedures and components for PSS. Integrating various related protocols like threshold cryptography with our PSS protocol would provide a feasible communication model. Considering the appropriate specification of threshold cryptography, including methods how the secret group key itself is bootstrapped and how n and t for (n, t) threshold cryptography are defined, is our future work.

References

[1] A. Shamir, "How to Share a Secret", Communications of the ACM, vol.22, pp.612-613, November 1979.

[2] Y. Desmedt, "Some Recent Research Aspects of Threshold Cryptography", Proc. Information Security, (Lecture Notes in Computer Science 1396), pp.158-173, Springer-Verlag, 1997.

[3] S. Jarecki, "Proactive secret sharing and public key cryptosystems", Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, September 1995.

[4] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive public-key and signature schemes", Proc. the Fourth Annual Conference on Computer Communications Security, pp.100-110, ACM, 1997.

[5] L. Zhou and Z. J. Haas, “Securing Ad Hoc Networks”, IEEE Network Magazine, vol.13, no.6, Nov/Dec 1999.
[6] J. Hubaux, L. Buttyan and S. Capkun, “The Quest for Security in Mobile Ad Hoc Networks”, Proc. ACM MobiHOC, October 2001.
[7] M. Narasimha, G. Tsudik and J. Hyun, “On the Utility of Distributed Cryptography in P2P and MANETs”, Proc. IEEE ICNP, November 2003.
[8] RSA Security, <http://www.rsasecurity.com/rsalabs/node.asp?id=2214>.
[9] J. Kong, P. Zerfos, H. Luo, S. Lu and L. Zhang, “Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks”, Proc. IEEE ICNP, November 2001.
[10] D. Eastlake, 3rd and P. Jones, “US Secure Hash Algorithm 1 (SHA1)”, RFC3174, September 2001.
[11] C. Perkins, E. Belding-Royer and S. Das, “Ad hoc On-Demand Distance Vector (AODV) Routing”, RFC3561, July 2003.