A PAIR-WISE KEY ESTABLISHMENT SCHEME FOR AD HOC NETWORKS

Isra Sitan Al-Qasrawi[1] and Obaida Mohammed Al-Hazaimeh[2]

[1]Department of Information Technology, Al-balqa' Applied University,
AL-Huson University College, Irbid, Jordan.

israonnet@yahoo.com

[2]Department of Information Technology, Al-balqa' Applied University,
AL-Huson University College, Irbid, Jordan.

dr_obaidam@yahoo.com

ABSTRACT

Ad hoc networks are a new wireless networking. However, these kinds of networks do not have an underlying fixed infrastructure as well as tend to be vulnerable to a number of attacks due to dynamic network topology and the lack of centralized network management functionality. In this paper, we have proposed a new scheme that will allow two ad hoc nodes to establish a pair-wise shared key on the fly (dynamic) during any stage of the network operations based on cellular automata (CA) rules without requiring the use of on-line key distribution centre. Our proposed scheme consists of a simple, but strong to provide a secure communication against node compromise due to pair-wise key establishment. In addition, the proposed scheme is computationally efficient because it only relies on simple symmetric key cryptography with high randomness, and meets desired levels of high reliability with low storage requirements.

KEYWORDS

Ad Hoc networks, Pair-wise key establishment, Cellular Automata (CA), Key Pre-Distribution (KPD)

1. INTRODUCTION

Key management is one of the most important issues of any secure communication in any type of network. It provides a way to maintaining the confidentiality of information from unauthorized users. Keys are fixed length streams of random bits, which are known to only the authorized nodes. Sender node encrypts information in the key to produce a stream of bits, which does not reveal anything about the original information. Only particular receiver can come to know the original information. For that reason, keys must be managed securely and efficiently. Key management of any computer network depends upon its characteristics, limitations and applications [1]. A wireless ad hoc network is a decentralized wireless network. It does not rely on a preexisting infrastructure, such as access points in managed wireless networks. Instead, each node participates in routing by forwarding data for other nodes, and so the determination of which nodes forward data is made dynamically based on the network connectivity. These nodes are small, with low power and low storage capacity. Quick deployment and minimal configuration make ad hoc networks very suitable for emergency situations like military conflicts and natural disasters. Such strong characteristics and critical demands make key management a non-trivial problem in ad hoc networks.

Public key cryptography may provide some help in solving this problem, but most of Ad hoc networks cannot afford to deploy public key cryptosystems due to the high overheads of storage, communication, and computation.

Public key cryptography uses up more computer resources (i.e. memory, energy, etc). Therefore, it is not suitable for ad hoc networks. i.e. 512-bit RSA signature generation takes 2 − 6 seconds on a Palm Pilot, and generating a 1024-bit key can take as long as 15 minutes. Consequently, it is better to use symmetric key cryptography [1].

The most important issue when using symmetric keys to providing secure communication is key distribution. Key distribution in Ad Hoc networks is more difficult than in traditional wired networks due to connectivity weakness. Furthermore, using traditional methods such as online key distribution creates a single point of vulnerability.

One approach to avoid a single point of vulnerability is to pre-load all the nodes in the network with their own keying information and with the keying information of all other network nodes, prior to deployment. Then, neighboring nodes establish shared keys after deployment using partial information exchanges. This scheme is called Key Pre-Distribution (KPD) scheme. Because the same keys are spread over a number of nodes in the network and keying information are exchanged between nodes, even a small number of compromised nodes can threaten the security of the network [2]. A key pre-distribution scheme makes the network nonscalable and introduces an inefficient offline initialization phase. These schemes do not allow for ad hoc network establishment. Most secure routing schemes [2- 4] neglect the critical impact of secure key management and assume pre-establishment and pre-sharing of secret key pairs [5].

Pair-wise Key Establishment is the most secure key management scheme for Ad Hoc networks, where pair-wise key is established between every pair of nodes in the network. For example, if the network has x nodes; every node will have x-1 keys stored in its memory. Pair-wise key establishment not only provides authenticity and confidentiality, but also has the ability to revocation the compromised nodes. Despite of all the benefits, the Pair-wise Key Establishment facing some challenges such as it is not suitable for large networks. If the number x becomes too large, the storage required per node increases linearly too. Therefore, appropriate strategies or techniques are needed to carefully manage this problem. In this paper, the size of each key is just a small string of bits, according to desired key length. We will illustrate the storage requirement in evaluation section.

This paper is structured as follows: brief overview of related research, present the details of the proposed pair-wise key establishment scheme, performance and security analysis, and conclusion.

2. RELATED WORK

Most of the proposed symmetric key cryptography schemes for pair-wise key establishment use an on-line key distribution centre. These schemes do not allow for ad hoc network establishment.
In 1993 Gong proposed a new scheme for using threshold secret sharing technique to increase the availability of authentication services [6]. We aim also to secret sharing the information needed to establish pair-wise keys, but unlike Gong's scheme, there is no need for any use of an on-line key distribution centre. One solution to solve this problem is probabilistic keying schemes. Mitchell and Piper [7] proposed a scheme based on probabilistic key sharing that does not depend on such an on-line server. However, the storage requirements imposed on each node in their scheme seems to be unbearable in the context of Ad Hoc networks.

A key management scheme based on probabilistic key sharing for distributed sensor networks proposed by Eschenauer and Gligor [8] with central key servers like base stations. They proposed using the pre-deployed keys for encrypting all communication between nodes. By the pre-deployed keys, a session key between two nodes can also be established. However, the established session key might not be exclusively known to the two nodes involved, because each pre-deployed key is known to several nodes.

Few numbers of works were made researches in finding new techniques based on pair-wise key establishment schemes in ad hoc networks. Many of them focused on wireless sensor networks particularly. i.e. schemes which based on key pre-distribution, shared-key discovery, and path-key establishment [9]. It requires $x > 1$ number of common keys for two nodes to establish a shared key. There is a big opportunity to nodes to become compromised, because the keys used by key pre-distribution are known to more than one pair of nodes. In our scheme, we avoid this kind of attacks by just sharing a CA rule and two parameters used to establish the key. This technique does not give an adversary any attack opportunity that she does not have because the keys are never transmitted or shared. Similar technique called CAB for wireless sensor networks. CAB is cellular automata based key management system that provides sensors with the ability of establishing pair-wise keys during any phase of the network operation using preloaded CAs. The computation in CAB is very simple and fast because it uses simple OR and XOR operations. It also has rekeying capabilities and achieves high resilience against node compromise [10].

3. PROPOSED SCHEME

3.1. Cellular Automata (CA)

It is a discrete model consisting of a regular grid of cells, each in one of a finite number of states, such as 0 and 1. The grid can be in any finite number of dimensions, typically one dimension (which we decide to use in our scheme). For each cell c(i), a set of cells called its neighbours is defined relative to the specified cell (left and right neighbours c(i-1), c(i+1)).

An initial state (t=0) is selected by assigning a state for each cell. A new generation is created (increasing t by 1), according to a rule or a set of rules based on the previous state of a cell and its neighbours, to determine the new state of each cell in the grid. The process of producing successive generations of the grid by updating its cells is called evolution. The evolution of a CA is influenced by three factors: its initial state, number of generations (iterations), and the rules. The combination of these factors impact the randomness of the values that a CA outputs; which will generate a key with high probability in our scheme. If we consider a three cells neighbourhood, i.e. the concerned cell c(i) and its left and right neighbours (c(i-1) , c(i+1)) in a one dimension and two states automaton (0,1) , there are only $2^3=8$ possible combinations that dictate a cell's next state. So, each 1-dimensional rule can be easily represented with 1 byte. According to that, $2^8=256$ possible rules can be used with 1-dimensional CA ranging from 0 to 255, each of which can be indexed with an 8-bit binary number. The evolution of rule 30 is shown in Table 1.

Table 1. Evolution of rule 30

Current states	**111**	**110**	**101**	**100**	**011**	**010**	**001**	**000**
Next state	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**

The logic function of rule 30 we have the equation 1:

$$c(i)_{t+1} = c(i-1)_t \oplus \left(c(i)_t \vee c(i+1)_t \right) \quad \text{..... (1)}$$

With the initial state of its centre cell set to (*) and other neighbouring cells set to ( ) space notation, the output of a CA that uniformly applies rule 30 with 15 iterations in our code implementation is shown in figure 1.

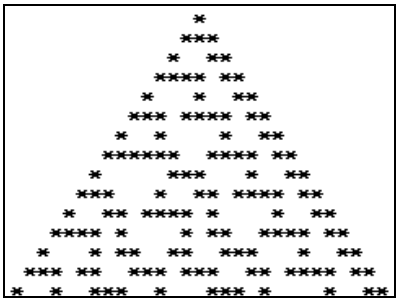

Figure 1. Rule 30 with 15 iterations

In figure 2 representing cells with 0 and 1, with 32-bit CA (Figure 1 representing by 0 and 1).

```
00000000000000001000000000000000
00000000000000011100000000000000
00000000000000100110000000000000
00000000000001111011000000000000
00000000000100010011000000000000
00000000000111101111011000000000
00000000010011000010011000000000
00000000111111001111011000000000
00000001000001110001001100000000
00000011100010011011110110000000
00000100110111101000010011000000
00001111010000101100111101100000
00010001011001100111000100110000
00111011001110111001101111011000
01001001110010001110100001001100
```

Figure 2. Representing 32-bits CA using 0 and 1

The last 32-bit string generated by 15th iteration is: 01001001110010001110100001001100. The values that make up the centre column of cells are believed to be highly random, and are hence used to compose a shared key. This is the base of generating keys in our scheme. In this paper, we have taken four critical factors in consideration during key-establishment phase to provide the highest probability, they are: the key length (i.e. 64, 128, 512, etc), the initial state for the CA, number of iterations until reaching the last state, and finally, the rule.

3.2. Network Assumptions

There are many assumptions in network environments, these assumptions are summarized as the following [6]:

A. There is no any neighbourhood information available to any node before deployment, and there are no keys or rules are pre loaded into nodes. Each node discovers its neighbours and shares parameters via local wireless broadcast after deployment.
B. If node A can hear node B, then B can also hear A. network links will be bidirectional by using Omni-directional antennas.

C. Every node has space for storing hundreds of bytes or a few kilobytes of keying information (in section 5, we show that for huge network of 1000 node and 1024-bit key length, the storage requirement for each node is just 125 kilobytes).
D. There is no an on-line key distribution centre exists in the formed network as it is Ad Hoc network, neither access points nor routers too.

3.3. Scheme Description

Ad Hoc networks are spread over a field. With the broadcast feature of wireless ad hoc networks, the adversaries have the opportunity to perform a variety of attacks and capture nodes. For that reason, keys must not be transmitted over the network. Also, preloaded keys according to key pre-distribution schemes are not suitable too in Ad Hoc networks. Therefore, we propose a new scheme that allows nodes to securely transmit small-sized parameters which prevent node capture attacks.

If an adversary eavesdrops the link and capture the parameters, she will not have any opportunity to guess the key, since parameters will not be sent together (source node will send the rule, and the destination will send the number of iterations and the initial state). However, in the worst case, if an adversary captures all parameters, she will not have the ability to compute the key, as there is no any relation between them to be understandable, (an adversary will not distinguish the different meanings of parameters). The proposed scheme is conducted in two phases: pair-wise key-establishment phase, and key-refreshing phase.

3.3.1. Pair-wise key-establishment Phase

Pair-wise key is established between every pair of nodes without requiring the use of any on-line key distribution center as shown in figure 3 (node A and B).

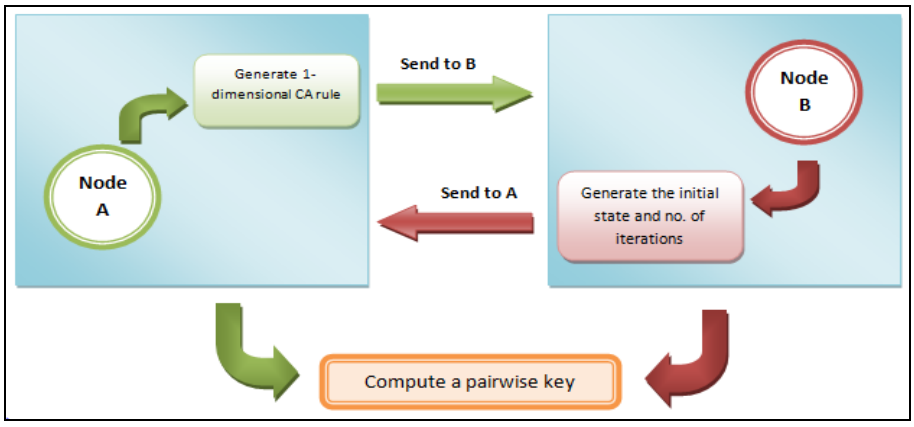

Figure 3. Pair-wise key-establishment between two nodes

In other words, the proposed scheme consists of the following steps:

Step 1: after deployment, when a node senses a neighbour in its range, it will send a message with 8-bit random string, represents a 1-dimensional CA rule. For example, if node A decides to use rule 45 to establish a shared key with node B, this 8-bit string will be sent: 00101101.
Step 2: node B will send a replied message to node A with two parameters used to configure their common CA, represent number of iterations (generations) to apply the rule and the initial state for the CA. Together with these parameters, the rule is applied to generate the pair-wise shared key

between nodes A and B . We can notice that the incorporation of additional parameters in computing a pair-wise key is better than selecting a key based on some partial information for a node. This will provide high randomness and no attack opportunity.

Step 3: if a node (e.g., node B) receives a CA rule (e.g., from node A) and replied with parameters, this means it agreed upon the rule and will not generate other rule to be used with node A.

Step 4: if a node (e.g., node B) received the same rule from two nodes (e.g., A and C), then it is important to ensure that different values for the parameters will be chosen for each node. Notice that for example, the number of initial state probabilities for a 128-bit key is 2^{128}. 340,282,366,920,938,463,463,374,607,431,768,211,456. And the number of probabilities for number of iterations is open. This means there is no any opportunity to generate the same key for more than two nodes.

Step 5: if a node (e.g., node A) sends a CA rule to a neighbour (e.g., node B) and doesn't receive any replied message, it will send the rule again, in case of no reply for the second time then node A will consider B an adversary or a compromised node.

Step 6: in the next step, nodes A and B will compute the pair-wise shared key depending on the rule, the initial state, and the number of iterations.

Step 7: if node A cannot communicate with node B directly, then an intermediate node C which can communicate with A and B directly, will receive the A-B rule encrypted by A-C key, and then transmit it again to node C encrypted by B-C key. By the same method, node C will receive the A-B parameters from B encrypted by B-C key, and then transmit them to A using A-C key. As a result, A and B will generate their shared key without allowing C to know the key, it read the rule and parameters without knowing their purpose. The pair-wise key established between any two nodes can thus be used to encrypt and decrypt messages exchanged between them.

3.3.2. Key Refreshing Phase

A network may need to refresh keys for multiple reasons (i.e. deployed nodes dynamically, and in order to prevent adversaries from guessing secret keys). It is very resilient to change old key to new one according to the high level of randomness available from the proposed scheme. It allows nodes to refresh keys during any stage of the network operation, and independent of any previous key refreshing phase. This phase will not lead to more power consumption and do not need more storage space, because of its efficiency and secrecy during the key establishment.

In details, we have two cases for key refreshing phase; these cases are summarized as the following:

1. If a new node has been deployed in the network, then it will sense for neighbours and make a connection with them to establish pair-wise keys as we illustrated before in the pair-wise key-establishment phase.
2. Key refreshing must be done at appropriate time intervals depending upon frequency of key usage. New values of the parameters will be sent to compute the new key.
3.

Two critical points must be taken in consideration: first, it is very important to ensure that the values of parameters used to establish the new key between A and B nods will not be the same used to establish the previous key, to avoid establishing the same key.

Second, the time intervals that decided to establish a new key depends on the average time that needed by an adversary to guess the key. For example, for a 40-bit key, there are 2^{40} possible values. By using a personal computer that can try 1 million keys per second, an attacker can try all possible keys in about 14.3 days. So, the key-refreshing phase must be done before spending

14 days of using the previous key. It is clear that this depends on key length (k). Longer key lengths increase the time needed to guess the key. As shown in equation 2.

$$\frac{2^k \text{ possible keys}}{\text{CPU frequency} * 3600 \text{ seconds} * 24 \text{ hours}} \quad \text{..... (2)}$$

4. SCHEME ANALYSIS

In this paper, we have proposed a new scheme that will allow two ad hoc nodes to establish a pair-wise shared key on the fly (dynamic) during any stage of the network operations. The performance and security analysis of the proposed scheme were conducted in the following sections.

4.1. Security

Using symmetric key cryptography, both nodes share the same key for encryption and decryption. To provide privacy, this pair-wise key needs to be kept secret. For that reason, the key must not be transmitted between nodes over the network. In our scheme, the key is not transmitted at all, just the CA rule and the parameters are exchanged between every pair of nodes. Even though the CA used by two nodes may be known to the adversary, the key used by them is not automatically revealed. This technique will keep the keys secret with a greater degree.

The key is established with the highest degree of randomness using the CA rules, and random values of parameters. For illustration, the number of probabilities for choosing a CA rule = 256. The number of iterations to be applied upon an initial state is open. May a node decides to apply 5 iterations and other one decides to apply 30 iterations. The number of probabilities to generate an initial state with key of length (k) = 2^k. For example, to generate a 256-bit key the number of probabilities is 2.3158e+77. These three factors provide the highest degree of randomness in key establishment. This high randomness will not provide any opportunity to more than two nodes to share the same pair-wise key at all. On the other side, the incredible number of the key probabilities will prevent node compromise. No adversary can compute the all probabilities to guess the key. Again, CA provides enough randomness thus it is really impossible for attackers to break the keys.

4.2. Computation Time

The proposed scheme is shown to be computationally efficient because operations are as simple as generating binary strings depending on the values of 3 neighbourhoods c(i-1), c(i), c(i+1). The CA rules calculations are very simple and less time consuming. Compared with CAB scheme, the CA rules used in this proposed scheme are just 1 byte rules consisting of 256 probabilities, while CAB rules may not be able to be represented by 1 byte, as a result they will consume more computation time. Figure 4 shows the establishing 32-bit key with 10 iterations based on CA rule 54. It is does not take more than 30 milliseconds during the generation, (i.e. we have simulated a key with short length such as 32 bits just to simplified the representation on the run screen).

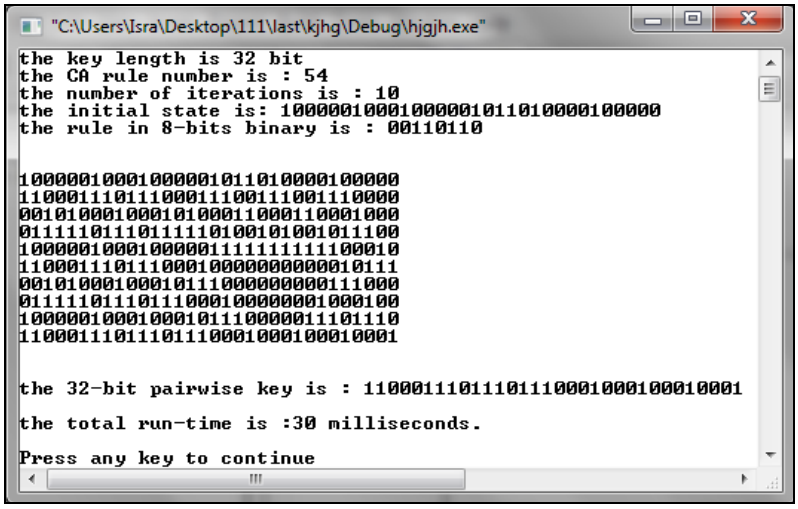

Figure 4. Establishing 32-bit key with 10 iterations based on CA rule 54

The computation time (ms) needed for establishing different length keys with different number of iterations based on CA rule 54 are calculated in table 2.

Table 2. Computation time for different key lengths (ms)

No. of iterations	Key Length (bits)						
	32	64	128	256	512	768	1024
5	10	10	15	46	78	160	200
10	30	30	31	109	171	330	450
20	37	50	78	265	358	630	870
30	50	70	156	327	592	946	1208
40	60	110	202	468	780	1267	1530
50	70	120	290	577	967	1557	1900

The following column charts showing the relationship between the key lengths and the computation time (ms) in figure 5.

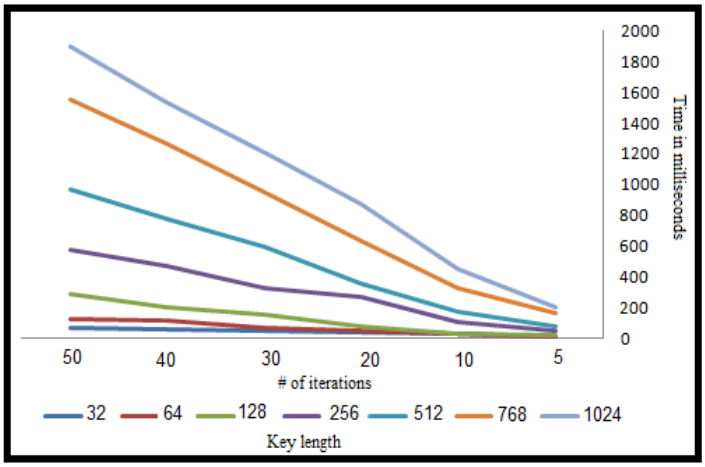

Figure 5. Column charts for key lengths and the computation time

In addition, the differences between the proposed scheme and CAB scheme in establishing a pair-wise 1024-bit key by CA rule 45 based on the equation 3, and different number of iterations in term of computation time are shown in figure 6.

$$c(i)_{t+1} = c(i-1)_t \oplus \left(c(i)_t \vee \left(\sim c(i+1)_t \right) \right) \quad \text{..... (3)}$$

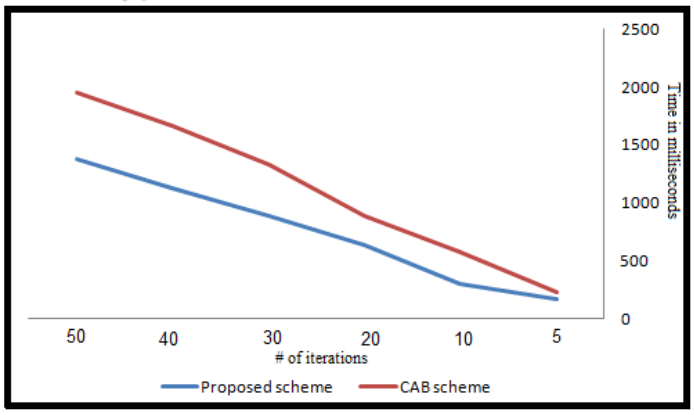

Figure 6. The differences between the proposed scheme and CAB scheme

4.3. Communications Overhead

The communication overhead is very low, one byte CA rule and the two parameters (number of iterations and the initial state) are exchanged between each pair of nodes to establish the pair-wise key. This shared information is a few bytes only. Our scheme achieves lower communication overhead than in-situ schemes such as iPAK and SBK [11, 12], whose communication needs transmitting the share of a symmetric matrix or a symmetric bivariate polynomial.

4.4. Storage Requirements

The storage requirements of our scheme are determined by two factors: the number of nodes in the network, and the key length. As we illustrated, a pair-wise key is established between every pair of nodes in the network. If the network has x nodes; every node will have x-1 keys stored in

its memory. The larger the number of nodes and the larger the key, the larger the storage space is needed, but it is not a constraint in our scheme, where the total storage space needed for each node to store all keys is very little. i.e. for a network consisted of 300 nodes, and 256-bit key per pair of nodes, every node needs to store 299 keys in its memory, with just $299*256/2^{13}= 9.34375$ nearly 9 kilobytes. And for a huge network of 1000 nodes and 1024-bit key length, every node needs to store 999 keys in its memory, with just $999*1024/2^{13}= 124.875$ nearly 125 kilobytes.
Compared with CAB scheme, the storage space needed for each node to store the key information is smaller in our scheme, based on the fact that node do not store pre loaded CA rules before deployment, nor initial states or parameters after deployment. CAB does not store keys, but instead it stores CA rules, which actually smaller than the key, but it needs every time to share the parameters and compute the key. At the same time, symmetric key cryptography needs smaller storage space compared with asymmetric one which use two keys one for encryption and another one for decryption.

4.5. Key Refreshing Ability

In the proposed scheme, establishing a new pair-wise key will not lead to increase power consumption, or wasting storage space. The key refreshing phase must be done frequently at appropriate time intervals depending upon frequency of key usage, to prevent guessing the key by time. Ad Hoc networks do not use an on-line key distribution centre, so no centralized server is needed for establishing new keys. Nodes will refresh or initiate a key by exchanging CA rule and the parameters as mentioned above. In case of refreshing key between two nodes, the CA rule and the parameters must be encrypted by the previous key. And in case of establishing a key for newly deployed nodes, the nodes will be agreed upon the CA rule and the parameters to be used.

4.6. Attacks Prevention

Proposed scheme is robust against many kinds of attacks (i.e. Eavesdropping attacks and Brute force attacks). Eavesdropping attacks, all the messages sent between two nodes are being encrypted by pair-wise key which are purely random, and if key length are of 160 bits then it is impossible to break down the system by guessing attack. Longer key lengths decrease the possibility of successful attacks by increasing the number of combinations that are possible. However, the attacker cannot eavesdrop to explore the key, because no any keys are transmitted over the network.

Brute force attacks, the adversary tries each possible key until the right key is found to decrypt the message. Most attacks are successful before all possible keys are tried. The proposed scheme minimizes the risk of this kind of attacks by choosing shorter key lifetimes (key refreshing phase), and longer key lengths. A shorter key lifetime reduces the potential damage if one of the keys is compromised. As we mentioned before, the time intervals that decided to establish a new key depends on the average time that needed by an adversary to guess the key.
with longer key length, for example, for an 128-bit key, if you use a computer that allow you to try 100 billion keys a second and you used 10 million of these computers, it takes about 1013 years to try every possible 128-bit key value.

Therefore, the longer the key, the more protection you provide to nodes from attacks. Nowadays, symmetric keys that are 128-bits or longer are considered unbreakable by brute force attacks. Moreover, our scheme establishes keys with 1024-bits and more with low computation time and low storage requirements.

In other words, in this paper the key plays the major role in providing a higher level of security where a key length of 1024-bit is chosen. But the key selection is done randomly so as to make it more difficult to intercept.

4.7. Correlation Analysis

The correlation between any kinds of data is known as intrinsic features. The existence of this feature can help attackers to trace the key. Therefore, correlation analysis is usually used to test the security and the correlation between the key. To analyze the correlations between the keys, Equation 4 is used to calculate the correlation coefficients.

$$cov(x,y) = \frac{1}{N}\sum_{i=1}^{N}\big(x_i - E(x)\big)\big(y_i - E(y)\big) \quad , r_{xy} = \frac{cov(x,y)}{\sqrt{D(x)}\sqrt{D(y)}},$$

where

$$E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i \ , D(x) = \frac{1}{N}\sum_{i=1}^{N}(xi - E(x))^2 \quad \text{..... (4)}$$

The strength of the relationship between set of keys (i.e. *Key 1, Key 2, Key 3,…, Key n*) is determined by a correlation coefficient, which ranges from -1 to +1. The closer the coefficient is to +1/-1, the stronger is the relationship. This means that the keys are related and are the same. In other words, if the correlation coefficient is equal to zero, then the keys are totally different. If the correlation coefficient is perfect correlation then the keys are same [13, 14].

In correlation analysis, we randomly choose different keys. Analysis result, the correlation coefficients for the set of keys is 0.0424 close to zero. This indicates that the keys are generated using the proposed schema are not correlated.

5. CONCLUSION

This paper introduced a new pair-wise key-establishment scheme in Ad Hoc networks to compute and generate pair-wise key on the fly between every two nodes during any stage of network operation without requiring the use of any on-line key distribution centre. Although there have been many researchers on this area, but most of the existing schemes have several weaknesses either caused by low security level or increase the delay time due the design of the scheme itself. The proposed scheme have been tested against different known attacks and proved to be secure against them. Therefore, it can be consider as a good alternative because of the high level of security with low computation time and low storage requirements.

REFERENCES

[1] M. Brown, D. Cheung, D. Hankerson, J. Hernandez, M. Kirkup, and A. Menezes.*" PGP in Constrained Wireless Devices". In 9th USENIX Security Symposium*, August 2000.

[2] Hu, Y.C., Johnson, D.B., Perrig, A.: Ariadne: "A Secure OnDemand Routing Protocol for Ad Hoc Networks". *Eighth ACM International Conf. on Mobile Computing and Networking (Mobicom)* 2002.

[3] Papadimitratos, P., Haas, Z.J.:" Secure Routing for Mobile Ad Hoc Networks". *SCS Communication Network and Distributed System Modeling and Simulation Conf.* 2002.

[4] Acs, G., Buttyan, L., Vajda, I.: "Provably Secure On-demand Source Routing in Mobile Ad Hoc Networks". *IEEE Trans. on Mobile Computing 5(11)*, 1533– 1546, 2006.

[5] Zhou, L., Haas, Z.J.: "Securing Ad Hoc Networks". *IEEE Network: Special Issue on Network Security* ,13(6) 24–30 ,1999.

[6] L. Gong. "Increasing Availability and Security of an Authentication Service". *IEEE Journal on Selected Areas in Communications*, 11(5):657–662, 1993.

[7] C. Mitchell and F. Piper. "Key Storage in Secure Networks". *Discrete Applied Mathematics*. 21pp 215-228. 1988.

[8] L. Eschenauer and V. Gligor. "A Key-Management Scheme for Distributed Sensor Networks". *ACM CCS*, 2002

[9] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, p. 197, 2003.

[10] Y. Teymorian, Liran Ma, and Xiuzhen Cheng. CAB: A Cellular Automata-Based Key Management Scheme for Wireless Sensor Networks. IEEE 2007.

[11] L. Ma, F. Liu, X. Cheng, and F. An, "An in-situ pair-wise key bootstrapping scheme for wireless sensor networks," IEEE Transactions on Parallel and Distributed Systems, 2006.

[12] F. Liu and X. Cheng, "SBK: A self-configuring framework for bootstrapping keys in sensor networks", *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2006.

[13] B.-H. Kang, D.-H. Lee, and C.-P. Hong, "High-Performance Psedorandom Number Generator Using Two-Dimensional Cellular Automata," *in the Proceedings of the* 4*th IEEE International Symposium on Electronic Design, Test & Applications*, vol. 46, pp. 597-602, Hong Kong, 2008.

[14] M. Alani, "Testing Randomness in Ciphertext of Block-Ciphers Using DieHard Tests", *International Journal of Computer Science and Network Security*, vol. 10, pp. 53-57, 2010.

Authors

Author[1] Isra Sitan Mohammed Al-Qasrawi received the B.S. degree in Computer Science from Al-Balqa' Applied University, Jordan in 2004, the MSc in Computer Science from Yarmouk University, Jordan in 2009, Working as instructor in Al-Balqa' Applied University / Al-Huson University College- Department of Information Technology.

Author[2] Obaida Mohammad Awad Al-Hazaimeh received the B.S. degree in Computer Science from Applied Science University (ASU), Jordan in 2004, the MSc in Computer Science/ Distributed system from University Science Malaysia (USM), 2005, and PhD in Computer Science/ Network security (Cryptography) for Real-Time Application (VoIP), 2010. Working as instructor in Al-Balqa' Applied University / Al-Huson University College- Department of Information Technology.