2012 31st International Symposium on Reliable Distributed Systems

Pairwise Key Generation Scheme for Cellular Mobile Communication

Chetan Jaiswal
Computer Science Electrical Engineering,
University of Missouri- Kansas City,
Kansas City, MO, 64110, USA
{cjvd3@mail.umkc.edu}

Vijay Kumar
Computer Science Electrical Engineering,
University of Missouri-Kansas City,
Kansas City, MO, 64110, USA
{kumarv@umkc.edu}

Abstract: **We present two new key generation schemes for secure communication between a pair of mobile nodes (cell phones). Unlike existing algorithms, our scheme does not (a) store a key chain in the memory from a universal key space, (b) use key broadcast, (c) distribute selected keys to the network nodes, and (d) use database of keys for selecting keys for communication. Rather, the pair of nodes that want to communicate securely generate identical keys independently with the help of a head node. We show the behavior of these schemes through a simple analytical model.**

Keywords - Key distribution; location parameters; security; key generation; cryptography; symmetric; asymmetric.

1. Introduction

The secure communication issues in mobile and wireless networks have become more important because of the widespread use of mobile (or cellular) and wireless networks and possible deployment of mobile database systems [21] in the near future. Unfortunately, mobile communications is more vulnerable to security attacks such as interception, eavesdropping, denial of services, and unauthorized access than fixed network communications [2, 4, 20]. This vulnerability makes a stronger case for securing mobile communications and guaranteeing accurate authentication and privacy of legitimate users in mobile database systems [21]. In this paper we present two innovative low-cost data communication schemes based on cryptographic approach for cellular network.

The cost of crypto-based security schemes has always been a major concern. However, recent development in encryption-decryption, key generation technology, and advancement in mobile hardware has significantly reduced the cost.

All existing cryptographic-based secured communication schemes have two common steps: (a) create sets of keys for communication during system setup time and (b) broadcast a subset of keys to nodes wishing to communicate with each other (see below). We argue that the key broadcast is fairly unreliable [14]. Important data should generally be transferred in unicast mode via radiograms or radio stream. In our scheme, therefore, we have nearly eliminated broadcasting keys and propose an approach to create strong keys dynamically for each pair of communication.

GSM (Global System for Mobile Communications) was not explicitly designed to protect against active attacks on the radio path, because they would require an attacker to masquerade or eavesdrop. In earlier times, these attacks were considered to be too expensive but as mobile service is becoming more widespread, the availability and the cost of the equipment required for such attacks is not a big deal. Thus, encryption has become cost-effective and a powerful way to protect the confidentiality and integrity of the exchanged information. The strength of cipher algorithm (encryption algorithm) depends on the length of cipher key. In GSM, the cipher key used is usually 64 bits; however, in practice the top 10 bits of the cipher key is set to zero to reduce the effective key length to 54 bits. Since these controls have been relaxed it is not possible to use more powerful cipher algorithms with much bigger key length [22].

The rest of the paper is organized as follows. Section 2 shows related work in key generation. In section 3, we present the architecture and working of our scheme. In section 4 we have discussed the problem of multiple authentication of the same node. Performance of our scheme with respect to other schemes in terms of storage and transmission cost is measured and analyzed in section 5 and section 6 concludes the paper.

2. Related Work

Several works have proposed secure mechanisms for key generation and distribution [5, 6, 10, 11, 12]. Chan et al. [8] developed two key pre-distribution techniques: *q-composite* key pre-distribution and random pairwise keys scheme. It also uses a key pool, but requires two nodes to compute a pairwise key from at least *q* pre-distributed keys they share, thus increasing the probability of key overlap required for key-setup. Both schemes, although improve the security over the basic probabilistic key pre-distribution scheme, the pairwise key establishment problem remains unsolved.

Varadharajan and Mu [19] has reviewed a number of works related to security issues in mobile environment and present a solution for secure mobile communication. They have pointed out the problems and limitations of BCY (Beller, Chang, Yacobi) and Carlsen protocols and proposed a novel concept of subliminal keys assigned to every mobile node in the network to conceal the original identity of the node.

3. Our Key Generation Scheme

We present two common key generation schemes for secure communication in cellular networks. In all existing schemes key distribution occur in some form. We discussed the vulnerability of the key distribution approach in generating a common key for communication. In our approach, therefore, we replace this phase by the process of mutual key generation. Under our scheme the nodes wishing to communicate generate the same common key independently using common information. These nodes obtain this information with the help of a head node (H). We observe that all existing schemes assume the presence of a head node in some form. We state below the importance of a head node in the key generation process.

- A pair of mobile units wishing to communicate with each other cannot authenticate each other without the help of *head*. Without authentication there is no trust on information exchanged.
- There are many researches done for developing trust between mobile nodes. [23] explains the trust development using game theoretic approach. However, only trust will not add to security among mobile needs. It needs secured channels with trust which can only happen with the help of a third party i.e. *head node*.
- In order to securely exchange information, two mobile needs a pair of private and public key. It is practically infeasible to keep a database of key pair in every mobile node for every other mobile node.
- Without authentication from trusted *head*, any malign node can masquerade other genuine node's identity.

We, therefore, argue that it is not practically feasible to develop a security protocol for communication between mobile nodes in current cellular wireless network without the help of *head node*.

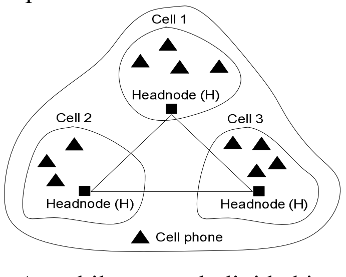

Figure 1. A mobile network divided into three cells

We have organized nodes into clusters. A node in our case could be a cell phone or a PDA, etc. Each cluster has a head node (H). In intra-cell communication there is only one and in the inter-cell communication there are two Hs. It does not matter which H is selected as a third party, in this paper we have used H of the requestor node (node requesting to communicate) as the third party. Any pair of nodes from different or the same cluster can communicate. For example, if a node from cluster 1 wants to communicate with a node from cluster 3, then the H of cluster 1 is selected as the third party and helps to generate the common key. A communication setup has two parts (a) information gathering and (b) key generation. In information gathering phase each node securely acquires necessary information from H. This is essentially a set of unique parameter values and geographical location of each node. Note that these parameters have never been used in any key generation algorithm or any security scheme the way we have done.

Our algorithms use one way hash functions. They obey the rule of *referential transparency*. It is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest. Any change to a message will (with a very high probability) result in a different message digest due to avalanche effect [15].

3.1 Assumptions

Similar to earlier works, we assume that there exists a secured channel between any mobile node and *H* that implements public key cryptosystem (RSA etc.).

We use $m_1, m_2, ..., m_n$ to represent mobile nodes. We further assume the presence of a *Secure Hash Algorithm (SHA-256)*, *MD-5* and *Advanced Encryption Standard (AES, formerly known as Rijndael)* that is replicated at every node of the network. *H* stores necessary information to generate common keys.

3.2 Key Generation Parameters

In order to generate the common key independently by the pair of nodes wishing to communicate (C-pair), the following set of parameters is used.

a. *IMEI (International Mobile Equipment Identity)*: It is a unique number that identifies a mobile device. This is used as a parameter because this is unique for each device and only known to mobile owner and subscribed network.
b. *IMSI (International Mobile Subscriber Identity)*: It is a unique identification number [16] associated with a subscriber. The IMSI of every mobile node is known to *H.* This is used as a parameter because this is unique for each subscriber and only known to subscribed owner and network.
c. *Time of Request*: The nodes of the C-pair that wish to communicate is referred to as a *requester* and the other node is referred to as the *receiver*. This parameter indicates the time when H receives a

communication request from the requester. Our scheme uses time implemented in *UNIX Epoch* format. This is used as a parameter because it adds dynamic nature to key generation.

d. *Geographical Location (L/L)*: The location of the receiver is captured in the form of L/L (latitude/longitude) and is measured by GPS in UTM (Universal Transverse Mercator coordinate system) format and converted to a series of alphanumeric codes. This is used as a parameter because it will add a random feature to key generation because user can be at any location.

3.3 Secure Key Generation Schemes (H-scheme and M scheme)

We present two key generation schemes for communication. We refer to the first scheme as *H-scheme* and the second as *M-scheme*. Both schemes use the same set of parameters discussed earlier but they generate them differently.

3.3.1 H-scheme

In this scheme, the requester generates a parameter referred to as *F'* using a *salted* [20] hash function (input) and the output of is *F'*. To make our scheme more secured, we made *F'* a dynamic parameter by using a *Nonce-Salt* approach. The other input to the hash function is its own IMEI number. We represent our one-way hash function as $f(x, y)$ (e.g. MD5, SHA-256 etc.) where x represents the message (input) and y nonce-salt. The steps of this scheme are listed below. The IMSI of every mobile node (subscriber) is known to *H*.

Requester's tasks: Suppose m_i (requester) wants to communicate with m_j (receiver). The m_i generates a request and *F'* and sends them to *H*:

a. $f_{\text{MD5}}(IMEI, NS) \Rightarrow F'$ *(NS = Nonce-Salt)*
b. Generate a Communication Request (CR) with $m_j \Rightarrow CR(m_j)$.
c. Compose a message for $H \Rightarrow (F', CR(m_j))$.
d. The message $(F', CR(m_j))$ is encrypted by two keys: first with the private key of m_i and then with the public key of *H*: $Encrypt(Encrypt(CR(m_j), F')_{Private\ key(mi)})_{Public\ key(H)}$. This dual encryption prevents masquerades and replay attacks, and authenticates m_i and makes $(F', CR(m_j))$ secured.
e. The encrypted message is sent to *H*.

Head node's tasks: *H* receives $CR(m_j)$. It decrypts it first with its private key and then with the public key of m_i and authenticates the node. Since no other node is aware of private key of m_i and if *H* is able to decrypt the message with public key of m_i that implies that the message was surely encrypted by using the private key of m_i. From the received message, *H* learns that m_i wants to communicate with m_j. If m_j is free to communicate with m_i (*H* finds this from m_j) then it generates the following parameters, otherwise it sends a denial message to m_i.

a. Current Time Stamp (*TS*) in (padded if required)
b. Unique communication session (UID) for m_i and m_j: $UID = f_{\text{SHA-256}}((IMSIm_i + IMSI\ m_j), TS)$. *(Note: + indicates concatenate operation)*
c. Location of m_j (*LL*).

Apart from the above three parameters, *H* also has *F'* from m_i. It sends the following messages to m_i and m_j. It encrypts each message with public key of m_i (PUm_i) and public key of m_j (PUm_j) respectively. Message to m_i is: $Encrypt\ (LL, TS, UID)_{PUmi.}$ and message to m_j is: $Encrypt\ (LL, TS, UID, F')_{PUmj}$.

At the end of these steps, m_j has *LL*, *TS*, *UID*, and *F'*. m_i uses SHA-256 [1] to generate the common key (Key_{Secret}) to communicate with m_j as follows:

a. *message* = *LL* + *TS* + *UID* + *F'*
b. Key_{Secret} = SHA-256 (*message*). It generates a *message digest* (hash value) of the size 256 bits.

m_j receives the following data from *H: LL, TS, UID*, and *F'*. Similar to m_i, m_j generates the Key_{Secret} as follows:
message = LL + TS + UID + F'. Key_{Secret} = *SHA-256 (message)*.

It generates a *message digest* (hash value) of the size 256 bits. Since the input parameters to m_i and m_j are the same and both generate same *message,* SHA-256 generates the same Key_{Secret}.

Example 1 – H Scheme

We use numerical values of these parameters to illustrate that the node pair generates identical Key_{Secret} for communication.

Requester – Suppose m_i wants to communicate with m_j:

a. IMEI(m_i) = 490154203237518. CR for $m_j = CR(m_j)$
b. Using IMEI number and Nonce-Salt, m_i will generate *F'* using MD-5 protocol: f_{MD5} *(IMEI, NS)* ⇒ *F'* ⇒ *f6e6c49813c59bae05259779fa8ed153 (HEX)*. m_i generates $Encrypt(Encrypt(CR(m_j), F')_{Private\ key(mi)})_{Public\ key(H)}$ and sends it to *H*.

H decrypts the message, authenticates m_i, and generates the following parameters:

a. Current time-stamp (*TS)* = 1319349790034 ms
b. Unique communication session (*UID) for* m_i *and* m_j $= f_{\text{SHA-256}}([IMSI_{mi} + IMSI_{mj}], TS)_{SHA\text{-}256}$. *Suppose* $IMSI_{mi}$ = 310150123456789, $IMSI_{mj}$ 460001234567890 and
TS = 1319349790034. The generated *UID* is
0203b152f033a2226b16d419804b50f8866757f6084 288d9b324f1f19415f713

c. Let the location of m_j be: 15N, X – 363569, Y – 4321694, LL = 15NX363569Y4321694

After generating the above parameters, H sends the message *Encrypt (LL, TS, UID)*$_{PUmi}$ to m_i and message *Encrypt (LL, TS, UID, F']*$_{PUmj}$ to m_j. At the end of these steps, m_i and m_j has four parameters: *F', UID, LL, and TS. Finally,* m_i and m_j generate the common Key_{Secret} as follows that can be used to encrypt and decrypt message.

a. Create Message to feed SHA-256, *message = LL + TS + UID + F'*, which is: *15NX363569Y43216941319349790034 89f11ac3d5db4ecf68eec78755989de + 10adc3949ba59abbe56e057f20f883e*

b. Finally, Key_{Secret} = SHA-256 (*message*) Key_{Secret} = 01914f110fdb35735f9004737ca9c25604ab2cf1dfa7a39a335a9e7b5cf7270c

3.3.2 M-Scheme

Unlike *H-Scheme*, the requester in this scheme generates *F'* using a *salted* [20] hash function as follows.

Requester's tasks: m_i (requester) generates a *CR* and *F'* and sends them to *H*:

a. f_{MD5}*(IMEI, NS)* ⇒ *F'*. m_j ⇒ *CR(m_j)*. Compose a message for *H* ⇒ (*F', CR(m_j)*).
b. (*F', CR(m_j)*) is encrypted twice: *Encrypt(Encrypt(CR(m_j), F')*$_{Private\ key(mi)}$*)*$_{Public\ key(H)}$ and sent to *H*.

H's tasks*:* It decrypts the message and authenticates the node. If m_j is free to communicate with m_i then it generates *TS* in milliseconds and *UID=*$f_{SHA-256}$*((IMSIm$_i$ + IMSI m$_j$), CTS). (Note: "+" indicates a concatenate operation). H* forwards *Encrypt [UID, F', CR(m_j)]*$_{PUmj}$ message to m_j.

Receiver's tasks*:* m_j receives the message from *H* and decrypts it using its own private key. After deciphering the message m_j generates the following parameters:

a. Its *LL* and *TS*. *TS'* = *TS*⊗*UID*. This will eventually lead to authentication of m_j by *H* in the later stage.
b. Compose *H* ⇒ (*LL*, *TS*, *TS'*). m_j sends *Encrypt (LL, TS, TS')*$_{PUH}$ message to *H*. m_j has *LL, TS, UID*, and *F'*. m_j generates *message = LL + TS + UID + F'*. Finally, Key_{Secret} *= SHA-256 (message)*. It generates a *message digest* (hash value) of the size 256 bits.

H's tasks (at the time of responding to requester): It receives the message from m_j and decrypts it using its own private key. It receives 3 parameters from m_j ⇒ (*LL*, *TS*, *TS'*) and calculates *UID'' = TS*⊗*TS'*. *H* generated *UID* earlier to uniquely identify the communication. *UID"* will be compared with *UID* and if both are equal then m_j will be authenticated. If not then communication will be denied to m_i. Once the authentication is done and m_j is authenticated, *H* composes a message for m_i: (*LL*, *TS*, *UID*). It encrypts the message using public key of m_i and sends the message: *Encrypt (LL, TS, UID)*$_{PUmi.}$

At the end of these steps, m_i has *LL*, *TS*, *UID*, and *F'*. The Key_{Secret} is generated as follows: *message* **=** *LL + TS + UID + F'***.** Key_{Secret} = SHA-256 (*message*). Since the input parameters to m_i and m_j are the same and both generate same Key_{Secret}.

Example 2. M Scheme

We use numerical values of these parameters to illustrate that the node pair generates identical Key_{Secret} for communication.

Requester – Mobile Node m_i: Suppose m_i is the requestor. It does the following. IMEI(m_i) = 490154203237518, m_j = *CR(m_j)*. Using IMEI number and Nonce-Salt (NS), m_i generates using MD-5, f_{MD5}*(IMEI, NS)* ⇒ *F'*⇒ *f6e6c49813c59bae05259779fa8ed153 (HEX)*. m_i sends the message *Encrypt(Encrypt(CR(m_j), F')*$_{Private\ key(mi)}$*)*$_{Public\ key(H)}$ to *H*.

H at the time of receiving request from requester (m_i): After decrypting the message and authentication of m_i, *H* generates the following parameter: TS = 1319349790034, *UID* =$f_{SHA-256}$*([IMSI$_{mi}$ + IMSI$_{mj}$], CTS)*$_{SHA-256}$. *Suppose, IMSI$_{mi}$*=310150123456789, *IMSI$_{mj}$*=460001234567890 and *TS*=1319349790034. It generates *UID* as
0203b152f033a2226b16d419804b50f8866757f6084288d9b324f1f19415f713
After generating the above parameters, *H*, sends message to the m_j: *Encrypt [UID, F', CR(m_j)]*$_{PUmj}$

Receiver – Mobile Node m_j: After decrypting the message, m_j generates its own location. Let the location be 15N, X – 363569, Y – 4321694 that gives LL = 15NX363569Y4321694. (TS) = 1322182276780, *TS'* = *TS*⊗*UID*. m_j sends *Encrypt (LL, TS, TS')*$_{PUH}$ message to H. m_j creates *message = LL + TS + UID + F' = 15NX363569Y4321694132218227678 00203b152f033a2226b16d419804b50f8866757f6084288d9b324f1f19415f73 f6e6c49813c59bae05259779fa8ed153.*

a. Generate, Key_{Secret} = SHA-256 (*message*) Key_{Secret}=0e65aa4c4d69af998f262298a6d29c1dba6d870af83b0289391ffcd88c0f5d70

H's tasks (at the time of responding to requester)*:* After receiving message from m_j and decrypting it, *H* has *UID* =0203b152f033a2226b16d419804b50f8866757f6084288d9b324f1f19415f713, *TS* = 1322182276780, and *LL* = 15NX363569Y4321694. *H* authenticates m_j as *UID" = TS ⊗ TS'*. If *UID" = UID*, then m_j is authenticated, if not then communication is denied to m_i. Suppose m_j is authenticated, *H* will send a message to m_i*: Encrypt (LL, TS, UID)*$_{PUmi}$

Requester –m_i: m_i has *F' = f6e6c49813c59bae05259779fa8ed153, UID* = 0203b152f033a2226b16d419804b50f8866757f6084288d9b324f1f19415f713, *LL* = 15NX363569Y4321694, and *TS*=1322182276780. m_i generates *message =LL+TS+UID+F' = 15NX363569Y43216941322182276780 0203b152f033a2226b16d419804b50f8866757f6084288d9b324f1f19415f73 f6e6c49813c59bae05259779fa8ed153* and finally Key_{Secret} = SHA-256 (*message*) = 0e65aa4c4d69af998f262298a6d29c1dba6d870af83b0289391ffcd88c0f5d70

3.3.3 Flexible Negotiation Handshaking

After the completion of key generation, both nodes have Key_{Secret}. Now m_i initiates the *Flexible Negotiation Handshaking (FNH – 2way)* with m_j and m_i will sends all available hash algorithms and key sizes it supports. It will also send available encryption algorithms with it. The message will be encrypted by using *AES* (Advanced Encryption Standard, originally called *Rijndael*) using the Key_{Secret}. It is important to point here that since we want nodes to use AES for the very first communication and AES requires key size to be 128 bits/192 bits or 256 bits [5]. Thus, for generating Key_{Secret} SHA-256 is used which gives the hash value of 256 bits that can be easily used with AES. However, *FNH* will allow the nodes to negotiate on hash algorithm, key size and even encryption algorithm. In addition to this, by using special *On-The-Fly* control frame dynamic key update can be done by exchanging new parameters. *FNH* will also serve the purpose of authenticating m_i to m_j, since m_i will also send *UID* it received from H to m_j. Consider: HA = {h1, h2, … hn: Set of all hash algorithms m_i supports}

E = {e1, e2, … en: Set of symmetric Encryption algorithms m_i supports}

K = {k1, k2, … kn: Set of possible key size m_i supports}

Message sent by m_i is *Encrypt (HA, E, K, UID)*$_{KeySecret}$

a. m_j receives the message from m_i that contain *HA, E, K and UID*. m_j will verify the *UID* it received from *H* and authenticates m_i**.** If authentication fails m_i discards all future communication with m_i until a new request message arrives from *H*. Once authentication is done, m_j will select hash algorithm h', encryption algorithm e' and key size k' and will reply back to m_i. Message sent by m_j is *Encrypt (HA', E', K')*$_{KeySecret}$. If the selection is different from SHA 256 and AES, then m_i will generate a new Key_{Secret} and will assume that the same key will be generated on other node.
b. m_i will receive message from m_j, and based on h' and k' it will generate new Key_{Secret}. Both mobile nodes now have Key_{SECRET} and e' (negotiated encryption algorithm). Thus, m_i will start sending data to m_j and vice versa.
c. At any time in the communication, any mobile node can ask the other to update key by using a new set of parameters. For example, m_i can ask m_j to send new value of location or timestamp, similarly m_j can ask m_i to update key based on new parameters. A control frame can be designed for such *On-The-Fly* key update. We have considered the case of key generation in a hand-off scenario; however, in this paper we do not discuss our solution.
d. When both nodes are done with the communication, they will send CommEnd ("Communication with m_k ended") to *H*. When *H* receives this message, it updates its log and records m_i and m_j as available for communication.

4. Correctness

It is mandatory to use deterministic one way hash function that also follows referential transparency (e.g. SHA-256) to generate key. We establish correctness of our schemes as follows:

a. Since our scheme uses SHA-256 for the same input, the output is same.
b. Our scheme makes sure that the parameters exchanging process preserves the accuracy and integrity, and also the order in which the parameters are fed to the key generator hash function.
c. For m_i and m_j, every-time a new communication is set-up, it will result in a new key. This is because the four parameters needed to generate the key, will change with time, and if the parameters change it will result in new hash value (key).
d. The requesting mobile node is authenticated before the request can be served, this prevents the masquerading attack.
e. If the L/Ls of two different mobile nodes differ by less than one meter (sensitivity may vary from 1 to 10 meters) then the GPS may not be able to identify these as two different locations. However, our scheme will not be affected by this issue because they use four parameters to compute the key.

5. Performance Analysis

We have studied the behavior of our schemes using M/M/1 queuing model with arrival of communication requests modeled under Poisson distribution. We make the following assumptions:

- H consists of a single server and a single queue
- λ = Average Arrival Rate. μ = Av. service rate
- Average Waiting Time, $w = \frac{\lambda}{\mu} * \frac{1}{\mu - \lambda}$
- Service Time = $\frac{1}{\mu}$
- Average Delay at *H* = Service Time + *w*

The behavior of our schemes is illustrated in Figures 2 and 3. The graphs in Figures 2 and 3 show the cost in terms of key set up time with increasing number of communication requests originating in the same cell.

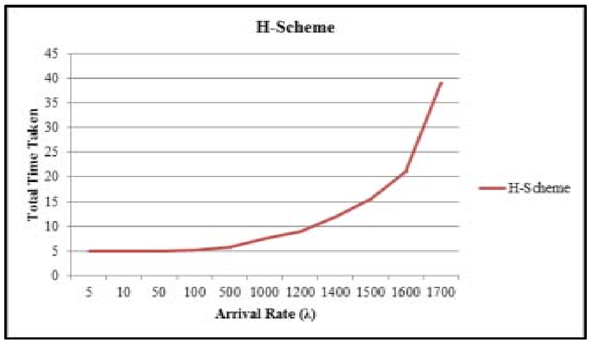

Fig. 2. H-Scheme

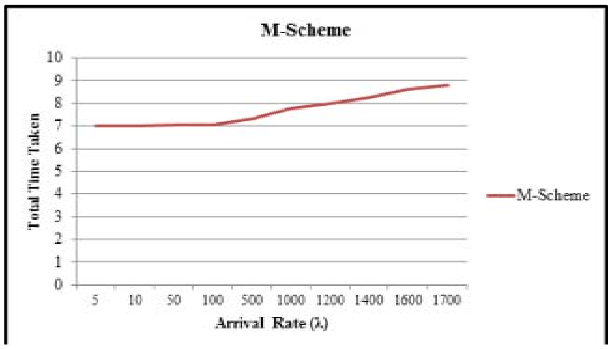

Fig. 3. M-Scheme

In H-Scheme the whole complexity lies at *H*. Due to this factor, this scheme is likely to be suitable for mobile users with limited capability and in comparison with M-Scheme, it may experience more delay in setup time.

In M-Scheme, since the complexity is pushed towards the edge of the network (mobile nodes), its performance is better than H-Scheme (as shown in these figures). This is as a result of less waiting time in the request queue at *H*. Less delay at *H* reduces the setup time and improves the efficiency and utilization. This scheme is more suitable for the mobile users with higher capability.

6. Conclusion

We presented two schemes for generating a common key independently. The key generation schemes are strong and secured. The information used in key generation is highly dynamic that further enhances the security of key generation. In our future work, we plan to compare the performance of our schemes with other schemes through a detailed simulation model to demonstrate cost-saving and security strength. We also intend to investigate potential breach of security in Hand-Off.

7. References

[1] A. Bharathidasan, and V. Ponduru, "Sensor Networks: An Overview", IEEE Infocom, Hongkong, 7-11 March 2004.

[2] A. Perrig, J. Stankovic, and D. Wagner, "Security in Wireless Sensor Networks", Communications of the ACM, June 2004.

[3] C. Haowen, and A. Perrig, "Security and Privacy in Sensor Networks", IEEE Computer Society, October 2003.

[4] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security Protocols for Sensor Networks", IEEE Infocom, Anchorage (Alaska), 22-26 April 2001.

[5] S.A. Camtepe, and B. Yener, "Key Distribution Mechanisms for Wireless Sensor Networks: A Survey", Technical Report TR-05-07 Rensselaer Polytechnic Institute, Computer Science Department, March 2005.

[6] L. Eschenauer, and V. D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks", In 9th ACM Conference on Computer and Communications Security, Washington DC, 18-22 November 2002.

[7] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks", In IEEE Symposium on Research in Security and Privacy, California, 11-14 May 2003.

[8] D. Liu, and P. Ning, "Location-Based Pairwise Key Establishment for Static Sensor Networks", In 1st ACM Workshop on Security of Ad Hoc and Sensor Networks, Fairfax, 31 October 2003.

[9] W. Du, J. Deng, Y. Han, and P. Varshney, "A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks" Proc. 10th ACM Conf. Computer and Comm. Security (CCS), Washington DC, 27-30 October 2003.

[10] D. Liu, and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks", In 10th ACM Conference on Computer and Communications Security CCS, Washington DC, 27-30 October 2003.

[11] S. Zhu, S. Setia, and S. Jajodia, "Leap: Efficient security mechanisms for large-scale distributed sensor networks", In 10th ACM Conference on Computer and Communications Security (CCS), Washington DC 27-30 October 2003.

[12] http://www.sunspotworld.com

[13] www.sunspotworld.com/docs/Purple/SunSPOT-OwnersManual.pdf

[14] N. Aakvaag, M. Mathiesen and G. Thonet, Timing and Power Issues in Wireless Sensor Networks" An Industrial Test Case, In Proceedings of the 2005 International Conference on Parallel Processing Workshops, 2005.

[15] Fips pub 180-3, federal information processing standards publication, secure hash standard (shs).

[16] Security in the GSM system: Jeremy Quirke © AusMobile.

[17] The design of Rijndael: AES--the advanced encryption standard: By Joan Daemen, Vincent Rijmen

[18] On the design of Security Protocols for Mobile Communications: Yi Mu, Vijay Varadharajan

[19] V Varadharajan, Security for personal mobile networked computing:.

[20] R morris and K. Thompson, "Password security: A Case History." Communications of the ACM, January 1997.

[21] Mobile Database Systems, Vijay Kumar, Wiley Publications.

[22] K. Boman, G. Horn, P. Howard and V. Niemi, "UMTS Security", Electronics and Communication Journal, October 2002.

[23] Dan Hirsch and Sanjay Madria, "A Cooperative Game Theoretic Approach for Data Replication in Mobile Ad-Hoc Networks".