Key Exchange Protocol Supporting Mobility and Multihoming

Mohammed A. Tawfiq, Sufyan T. Faraj, and Abdul-Karim A-R. Kadhim
College of IT, Nahrain University, P.O. Box 64065
Al-Jaderiya, Baghdad, Iraq
Correspondence Email: sufyantaih@ieee.org

ABSTRACT:

In this work, a new key exchange protocol for IP-based mobile networks is introduced. This protocol is called KEPSOM (**K**ey **E**xchange **P**rotocol **S**upporting M**o**bility and **M**ultihoming). The goals of designing KEPSOM are to develop key exchange protocol proposal characterized by its secrecy, simplicity, efficiency, resistivity, and its ability to support mobility and multihoming. The protocol requires only two roundtrips. The design limits the private information revealed by the initiator. An old security association (SA) can be replaced with a new one by rekeying without the need of restarting the protocol with a new session. On the other hand, the changes in IP address due to mobility or multihoming need not to restart the protocol with a new SA session. The proposed protocol can also support key exchange in hybrid wireless network, in which the mobile node can operate in both Ad Hoc and Base Station-oriented wireless network environments using different transmission modes. KEPSOM has been analyzed and proven secure. Several tests have been done to measure and evaluate the performance of the protocol. In these tests, it is found that the required time for rekeying is about 27% of the total required time for exchanging the keys. And the required time to detect and update the change in IP address, which may occur due to mobility or multihoming, is less than 10% of the total required time to establish a new SA sessions.

Keywords: Authentication, IP, Key Exchange, Mobility, Networks, SIGMA.

بروتوكول لتبادل مفاتيح التشفير يدعم التنقل وتعدد مفاتيح الارتباط

محمد علي توفيق و سفيان تايه فرج و عبد الكريم عبد الرحمن كاظم
كلية تكنولوجيا المعلومات- جامعة النهرين

الخلاصة:

يتم في هذا البحث تقديم بروتوكول جديد لتبادل مفاتيح التشفير للتطبيق في الشبكات النقالة المبنية على أساس بروتوكول الأنترنت (IP-based)، حيث قمنا بإطلاق مختصر "كيبسوم" (KEPSOM) كتسمية لهذا البروتوكول المقترح. إن أهداف تصميم "كيبسوم" هو لتطوير بروتوكول تبادل مفاتيح يتميز بالسرية، والبساطة، والكفاءة، والمقاومة العالية، إضافة إلى القدرة على دعم الحركة وتعدد نقاط الارتباط ضمن الشبكة؛ حيث يتطلب هذا البروتوكول جولتين كاملتين من التبادل فقط. كما يمكن تعويض رابطة أمنية (SA) قديمة بأخرى جديدة من خلال تجديد المفاتيح دون الحاجة لأعادة بدء البروتوكول من جديد، وكذلك هو الأمر بالنسبة لتغير عنوان التشبيك.

ويمكن تطبيق هذا البروتوكول أيضا في الشبكات اللاسلكية الهجينة ذات أنماط البث المختلفة.. ولقد تم تحليل "كيبسوم" واثبات أمنيته، كما قمنا بإجراء عدة اختبارات لقياس أدائه، حيث تبين من هذه الاختبارات أن البروتوكول يحتاج في حالة تجديد المفاتيح إلى حوالي ٢٧% فقط من الوقت اللازم لإعادة التبادل من جديد ، كما انه يحتاج في حالة تغير عنوان التشبيك إلى ١٠% فقط من الوقت اللازم لبناء رابطة أمنية جديدة..

1. INTRODUCTION

Many key exchange protocols already exist and have been implemented for a variety of applications and environments. Several have been proposed for the IPsec protocol suite. IKE (Internet Key Exchange protocol) is one of them. IKE has a number of deficiencies; the three most important are: the number of rounds is high (which increases vulnerability to denial-of-service (DoS) attacks), the complexity of the protocol, and its specifications. IKEv2 represents the new proposal to replace the IKE protocol [1]. Another protocol proposal called “Just Fast Keying” or JFK has also been introduced [2]. It performs many of the same functions as IKEv2. The authors of this protocol aimed to make JFK simpler than IKE, which in turn would make it more secure and easier to allow different implementations of the protocol work with each other. They also wanted to make it resistant to DoS attacks. The IKEv2 and JFK protocols have chosen to address a particular feature the same way, and for many other features, the two proposals differ.

This work presents a new protocol proposal called “**K**ey **E**xchange **P**rotocol **S**upporting M**o**bility and **M**ultihoming” (KEPSOM). KEPSOM combines the strength features of each of IKEv2 and JFK, modifies some, and add others. The proposed protocol supports key exchange in mobile and multihomed IP-based networks. It can also support key exchange in hybrid wireless network, in which it combines infrastructure network system, and noninfrastructure (Ad Hoc WLAN) system.

The rest of this paper is organized as follows: the main design features of KEPSOM are introduced in Section 2. While the protocol details are presented in Section 3. The issues of mobility and multihoming in KEPSOM are considered in Serction 4. Next, Section 5 is dedicated to consider the issue of applying KEPSOM in the environments of hybrid wireless networks. The security of KEPSOM is formaly analyzed in Section 6. Then, some exprimental results are presented in Section 7. Finally, the work is concluded in Section 8.

2. DESIGN OF KEPSOM

The design of a new proposal for key exchange protocol must have the following characteristics:

1- Security: the resulting key should be cryptographically secure.
2- Simplicity: it must be as simple as possible.
3- It must be efficient with respect to computation, bandwidth, and number of rounds.
4- it must resist any type of DoS attacks.
5- It must support mobility and multihoming environments.

To achieve these characteristics for the proposed protocol, the key exchange is accomplished within a minimal number of rounds in order to resist DoS attacks, and results in greater simplicity in implementation, debugging, and operation.

It is recommended that the proposed protocol should be negotiable. Negotiation is important in order to fixe errors, delete SA’s (security association's) payloads, rekeying, update addresses, and so on. Thus, the protocol proposal is divided into two parts. The first part represents all the features that can complete the key exchange operations. While, the second part represents the negotiations which can be established between the communicating endpoints using control message exchanges.

2.1 Main Features Exchange (MFE)

Only two round trips are required to exchange the main features that can establish any SA. Each round trip consists of two messages. Moreover, each message contains some payloads. These payloads are indicated by names. Payloads shown in brackets are optional. The main feature exchange (MFE) is shown in **Figure (1)**, where:

HDR: is the header payload, which contains the SPIs, flags of various sorts, and lifetimes.
N_i, N_r: are Initiator and Responder nonces, respectively.
KE_i, KE_r: are payloads that contain the Initiator's and Responder's Diffie-Hellman values.
CERT : is payload that contains an indication by the Initiator to the Responder as to which ID (and corresponding key material) the latter should use to authenticate to the former, (e.g., certificate information).
SA: the cryptographic and services properties of the security association that the Initiator and Responder want to establish.
$S_x(M)$: is the digital signature of message M with the private key belonging to principal x (Initiator or Responder).
SPI : is the security parameter index.
k_e, k_a : a shared key used respectively to encrypt and authenticate messages (3) and (4).
RK : is an optional payload. It is used for purpose of rekeying.
ID_i, ID_r : identification payloads which allow peers to assert an identity to each other.
$H_k(M)$: keyed hash of message M using key k.
HKR : a transient hash key private to the responder.
TS_i, TS_r : traffic selector payload. It allows peer to identify packet flows.

I→R: HDR , N_i , KE_i , CERT

R→I: HDR , N_i , N_r , KE_r , ID_r , $SA_{r1,}$ S_r (SPI_r , N_i, KE_r , ID_r) , H_{HKR} (KE_r , N_r , N_i)

I→R: HDR , [H_{HKR} (KE_r , N_r , N_i)], E_{kei}([RK] , N_i , KE_i , SA_i , [ID_i , TS_i , TS_r, S_i (SPI_i , N_r, KE_i , ID_i)]),[H_{kai}(E_{kei}(N_i , KE_i , SA_i , ID_i , TS_i , TS_r, S_i(SPI_i , N_r , KE_i , ID_i)))]

R→I: HDR , E_{ker}([RK] , N_r , KE_r , SA_{r2} , [ID_r , TS_i , TS_r, S_r(SPI_r, N_i , KE_r, ID_r)]) , [H_{kar}(E_{ker}(N_r , KE_r , SA_{r2} , ID_r , TS_i , TS_r , S_r(SPI_r , N_i , KE_r, ID_r)))]

Fig. 1. Main feature exchange of KEPSOM

The four messages used for MFE in KEPSOM are described as follows:
Message *(1):* The initiator's first message in MFE contains the HDR payload, the initiator nonce, the KE payload which sends the Diffie-Hellman value, and the CERT payload which inform the responder as to what authentication information the latter should use. The HDR payload contains the initiator's SPI for the SA to be established. It

also contains the lifetime values of the initiator and the responder that are determined by the initiator. The nonce in payload N_i serves two purposes; first, it allows the initiator to reuse the same KE value in multiple instances of the protocol. This will insure that the resulting session key will be different. Secondly, it can be used to differentiate between different parallel sessions.

Message (2): The responder replies with N_i and his random nonce N_r, the KE payload that sends his Diffie-Hellman value KE_r, his identity, and his SA_{r1} payload, which states the cryptographic algorithms that the responder supports for the SA. The message contains a signed copy of his SPI_r, initiator's nonce, his Diffie-Hellman value, and his identity (certificates or a string identifying his public key). He replies also with an authenticator calculated from a pseudo random function (PRF) known to the responder. The authenticator is computed over the responder's Diffie-Hellman value, and the two nonces.

Message (3): This message can be used for two options. The first option is to complete the establishment of the key exchange for a new SA, while the second option is used for rekeying the current SA. These two options depend on the existence or absence of the optional payload (RK). In the absence of the RK payload, the message repeats back the data sent by the responder, including the authenticator. The authenticator is used by the responder to verify the authenticity of the returned data, and to protect the integrity of the second message. The authenticator also confirms that the sender of the message (3) uses the same address as in message (1). This can be used to detect and counter a "cookie jar" DoS attack, which involves an attacker that is willing to reveal the address of one subverted host so as to acquire a valid cookie (or number of cookies) that can then be used by a large number of other subverted hosts to launch a DoS attack using the valid cookie(s) [2]. The message also includes the initiator's nonce, initiator's Diffie-Hellman value, SA_i, the initiator's identity, the traffic selectors for both initiator and responder, and a signature computed over SPI_i, the responder's nonce, the initiator's Diffie-Hellman value, and the initiator's identity. This latter information is all encrypted and authenticated under keys k_{ei} and k_{ai} derived from the Diffie-Hellman computation and the nonces N_i and N_r. The encryption and authentication use algorithms specified in SA_i. The signature is protected by the encryption. This is necessary for identity protection, since every thing signed is public except SA. The payload RK is used in the case of rekeying the current SA, otherwise it must be omitted. If RK is exists, then all the other optional payloads in the message must be removed.

Message (4): The responder replies with message like message (3) but for responder entities, every thing is encrypted and authenticated under k_{er} and k_{ar}. The encryption and authentication algorithms are specified in SA_{r2} in which it matches SA_i. The payload (RK) is used for rekeying the existing SA, otherwise it must be omitted.

The core cryptographic design of the proposed protocol follows that of SIGMA [3] (the explanation of SIGMA protocol is delayed to Section 6), JFK [2], and IKEv2 [1]. The authors of JFK and IKEv2 have stated that they are quite open to adding or changing major parts of their protocols [4]. However, some features are so much part of the core of a protocol that they cannot be changed. Thus, in the design of KEPSOM, some features of the above mentioned proposals have been matched and/or mixed. While some other new features have been built and added.

2.2 Control Message Exchange (CME)

Peers may desire to convey control messages to each other regarding errors or notifications of certain events. This can be accomplished by using a control message exchange. JFK protocol does not provide any notification messages. The control functions that might be used are provided from other protocols. IKE protocol provides many control messages. These messages have been modified and some extensions are added. Also new extensions are suggested and are under development.

To make efficient and secure control message exchange (CME) in the proposed protocol the following considerations must be taken into account:

1- CME may only occur after MFE has been established, and are cryptographically protected using the negotiated keys.
2- Messages in CME may contain zero or more notification, delete, and configuration payloads.
3- For each CME request, the recipient must send some response (else, the sender will retransmit the message assuming that it was lost in the network unless lifetime is finished).
4- To delete an SA, a CME message with delete payload is sent listing the SPIs of the SAs to be deleted. The recipient must close the designated SAs. The response will contain delete payload in the other direction.

Figure (2) defines general form of the CME messages, in which, the notation [ ] indicates that this payload is optional. Indeed, N means the notification payloads, and D is the delete payload. The processing of a control message exchange is determined by its component payloads.

3. KEPSOM DETAILS

The protocol proposal is designed to listens and sends on UDP port. Since UDP is unreliable protocol, then the design of the proposal must includes recovery transmission errors, including packet loss, packet replay, and packet forgery. The protocol is designed to function as long as:

1- At least one of a series of retransmitted packets reaches its destination before timing out.
2- The channel is not full of forged and replayed packets such that they exhaust the network or CPU capacities of either endpoints.

3.1 Retransmission timers

All messages in the protocol proposal will exist in pairs, a request and response. For each pair of messages one end of the SA is the initiator, and the other one is the responder. The rules in which initiator and the responder work for every pair of messages are as following:

1- The initiator is responsible for retransmission in the event of timeout.

2- The responder never retransmit a response unless it receives a retransmission of the request.
3- The initiator must remember each request until it receives the corresponding response.
4- The responder must remember each response until it receives a request.

These rules enable the protocol to be reliable in the sense that the initiator retransmit a request until either it receives a corresponding reply or it discards all state associated with SA.

I→R: HDR , E_k([N,] [D,] ….)

R→I: HDR , E_k([N,] [D,] ….)

Fig. 2. Control Message Exchange of KEPSOM

3.2 Sequence Number for Message ID

All the messages in the protocol contain a message identifier as part of its fixed header, called message ID. This identifier is used to control retransmission of lost packets and matching of requests and responses. It is essential to the security of the protocol, because it is used to prevent message reply attacks. It is crypto-graphically protected. The message ID is a 32 bits length. If the message IDs grow too large to fit in 32 bits, the MFE SA must be closed. Rekeying an SA will reset the sequence number.

3.3 Connection Timeouts

It is important when an endpoint either fails or reinitializes its state that the other point detects those conditions and not continue to waste network bandwidth by sending packets over discarded SAs. An endpoint must conclude that the other endpoint has failed according to the following rules:

1- When repeated attempts to contact the other endpoint have gone unanswered for a timeout period.
2- When a cryptographically protected MFE notification is received on a different SA to the same authenticated identity.
3- When an endpoint sends a CME to see whether the other endpoint is alive, with no response (acknowledgment) returned back.

Receipt of a cryptographically protected message on an SA assures liveness of that SA. According to these rules, if the endpoint presumes that the other endpoint is dead, then the associated SA and all of its components must be deleted. In this case the endpoint should send a delete payload indicating that it has closed the SA.

3.4 Nonces and Cookies

Each message in MFE of the proposed protocol contain nonce(s). these nonces are used as inputs to cryptographic functions and to insure creation of strong pseudorandom bits from the Diffie-Hellman key. The nonces used must be randomly chosen, with length of at least 128 bits. The first few bytes (2 bytes for IPv4 and 4 bytes for IPv6) of this nonces may contain cookies used by initiator and responder to avoid fragmentation-based DoS attacks.

3.5 Cryptographic Algorithm Negotiation

The SA payload indicates a proposal for a set of choices of cryptographic algorithms associated with the protocol. In IKE an SA consists of one or more proposals. Each proposal includes one or more protocols. Each protocol contains one or more transforms, each specifying a cryptographic algorithm. Using this approach will create the opportunity for an exponential explosion of proposals, assuming many of one type of algorithms worked with many of another type of algorithms. One another approach uses a single suite that is given by the responder as a take-it-or-leave-it, it is not negotiable. This approach is used in JFK protocol.

The approach that has been used in KEPSOM is that the responder can propose many suite options for different algorithms. The initiator then choose a single suite, which may be any subset of the responder's SA proposals. This approach is simple to encode.

3.6 Rekeying

Secret keys used in SAs should only be used for a limited amount of time to protect a limited amount of data. When the lifetime of the entire SA expires, the SA must not be used, and a new one must be established. Reestablishing a new SA to take the place of the expired one is referred to as "rekeying".

In general, rekeying in this work can be done without restarting the entire SA. However, if an SA is expired or is about to be expire and rekeying attempts fail, then SA must be closed and starting a new one. Messages (3) and (4) in MFE can be used to rekeying an existing SA. In this case the RK payload must be exists and all other options in these two messages must be omitted. For such a case message (3) and message (4) are going to be as shown in **Figure (3)**.

Message (3) :I→R: HDR , E_{kei}(RK , N_i , KE_i , SA_i)

Message (4) :R→I: HDR , E_{ker}(RK , N_r , KE_r , SA_{r2})

Fig. 3. The forms of messages (3) and (4) in the case of rekeying

In IKEv2 and JFK, each end of the SA is responsible for enforcing its lifetime policy on the SA and rekeying the SA when necessary. The two ends in IKEv2 and JFK may have different lifetime policies, or the same lifetime policy. In the case of different lifetime policies, the request for rekeying is the responsibility of the endpoint with the shorter lifetime. On the other hand, if the two ends have the same lifetime policy, it is possible that both ends will initiate a rekeying at the same time (which will results in redundant SAs). To reduce the probability of equal lifetime, some techniques must be used to ensure different lifetimes, which make this approach somehow more complex.

In KEPSOM, the technique used to overcome this problem is by making the initiator is responsible for enforcing two different lifetimes, one for the initiator it self, and the other is for the responder. In this technique, if an SA bundle has been inactive for a long time and if an endpoint would not initiate the SA in the absence of traffic, the endpoint may choose to close the SA instead of rekeying it when its lifetime expires. The node that initiated the surviving rekeyed SA should delete the replaced SA after the new one is established.

3.7 Traffic Selector

Traffic selector (TS) payloads allow endpoints to communicate some of the information from their security parameter database (SPD) to their peers. TS payloads specify the selection criteria for packets that will be forwarded over the newly set up SA. JFK offered a single selector having multiple address ranges and multiple port ranges. In IKE, the TS payloads offered multiple selectors, each selector has a source port, a destination port, and either a range of addresses, a subnet, or single address. The responder is allowed to narrow the choices by selecting a subset of the traffic.

The approach used by IKE has been used in this work with some modifications. Two TS payloads appear in each of messages (3) and (4) of the MFE (in IKEv2, these payloads appear in the messages that create a CHILD-SA pair, i.e., in phase II of IKE). Each TS payload contains one or more traffic selector. Each traffic selector consists of an address range (IPv4 or IPv6), a port range, and an IP protocol ID (e.g., UDP/TCP/ICMP). In KEPSOM proposal, the responder, just like IKEv2, can choose a subset of the traffic proposed by the initiator by eliminating or narrowing the range of one or more members of the set of traffic selectors, provided that, the set does not become null.

3.8 Signature and Authentication

Endpoints are signed and/or authenticated by applying a sign or a MAC (message authentication code) on a certain block of data. For the responder, the block of data to be signed contains the following:

- The first octet of the first SPI in the header of the second message.
- The value of the initiator's nonce.
- The value of PRF(k_{ar}, ID_r-), where ID_r- is the responder's ID payload excluding the fixed header.

Similarly, for the initiator, the block of data to be signed is:

- The first octet of the first SPI in the header of the first message.
- The value of the responder's nonce.
- The value of PRF(k_{ai}, ID_i-), where ID_i- is the initiator's ID payload excluding the fixed header.

Noting that neither the nonces (N_i, N_r) nor the value PRF(k_{ar}, ID-) are transmitted.

CERT in message (1) represents an indication by the initiator to the responder as to what authentication information (e.g., certificate) that the responder should use. In messages (2) and (3), H_{HKR} is a pseudorandom function. This also implies that H is a secure MAC function . H is computed over the responder's exponential, the two nonces, and the initiator's network address, using a secret key HKR. In messages (3) and (4), the encryption is followed by MAC (H) authentication with the symmetric key k_a. The H function here is computed over the ciphertext.

3.9 Error Handling

In some cases, either of the protocol participants may detect an error that can be recovered. Recovering these errors sometimes needs to send an error message to the peer. In these situations, the protocol is designed to support the transmission of error messages.

In general, there are two kinds of error messages: error messages exchanged in the course of the key exchange protocol itself. And error messages after the protocol exchange has been completed. In IKEv2, there are two error messages that are not protected: errors in response to message (1) in the exchange (requesting a cookie, or indicating an unacceptable SA payloads), and notifications sent when a host receives as ESP (Encapsulating Security Payload) or AH (Authentication Header) packet with an unknown SPI (possibly because of a reboot). Other messages are protected under an IKE-SA, and their delivery is confirmed. In JFK, an error notification returned as a response to message (1) cannot be protected. Error in message (3) can be protected, and errors in message (4) are authenticated under key material already exchanged. JFK itself does not provide any other error or notification messages[4]. In general, message (1) in all key exchange protocols can not be protected, because no secret keys are available between participants. In KEPSOM, the problem with message (1) is exist, just like IKEv2 and JFK, while all other error messages are designed to be protected and their delivery is confirmed.

Errors that occur before a cryptographically protected SA is established must be handled very carefully. The way in which this protocol handles error messages that are received at a node is as follows:

1- If the message is marked as a request, the node may send a response after auditing this message. The response will be sent to the IP address and port from whence it came with the same SPIs containing notify payload indicating invalid (error) message. The response will not cryptographically be protected.

2- If the message is marked as a response, the node must audit the suspicious event without any responding.

3- If the received message is unprotected Notify payload, the node will treat such a message as a hint that there might be problems with SAs to that IP address. The node will not respond or change the state of any existing SAs, because this message may be a forgery.

4. MOBILITY AND MULTIHOMING IN KEPSOM

In some cases, the IP address pair used during the protocol execution when establishing the SA may be changed due to mobility or multihoming devices. This problem can be solved by rekeying the SA after the IP address has changed. However, this can be

problematic and an impractical solution. Thus, a mechanism to update the IP address without requiring a new SA is needed. This mechanism should consider the following:
1- Notifying the change of IP address of the other endpoint.
2- Updating the SA endpoint address according to the notification.
3- Using the new IP address if the old one does not work any more.
4- Ensuring that the given new address belongs to the peer.

There are different scenarios in which the IP address can be changed [5][6]. Changing the IP address due to mobility or multihoming in most of the existing key exchange protocols, such as JFK, needs to establish new SA. A new protocol extension to IKEv2, which support mobility and multihoming, is under development [7][8][9]. This protocol extension is called MOBIKE. From MOBIKE's point of view, multihoming support is integrated by supporting a peer address set rather than single address and protocol mechanisms to change to a new IP address are proposed. The preferred mechanism to MOBIKE is to use the Notify payload. The notification payload of IKEv2 can only store one SPI per payload. If there are large number of IPsec SAs, then large number of notify payloads must be established.

In the design of KEPSOM, the following design decisions have been taken as follows:

1- Storing a single or multiple IP addresses:

In storing a multiple IP addresses, if peer **A** provides a peer address set with multiple IP addresses to peer **B**, then when **B** detects that the used path does not work any more, it can try an IP address from **A**'s peer address set. Using this approach can recover the situation that both peers happen to lose their IP address at the same time. On the other hand, the problem with this approach is mostly in its complexity. How many addresses in the IP address set? What are the priorities of the addresses and what technique should be used to determine it? How to check the availability of the appropriate address?

The alternative approach is storing a single IP address. In case that this address can not be used any more, then an address update is sent to the other peer changing the primary address. The main advantage of this approach is that it makes retransmission easy, and it also simplifies recovery procedure. The only problem with this approach is when both peers lose their IP address at the same time. As mentioned previously, in mobile IPv4/IPv6, the home agent can be used to solve this problem.

Thus, the design decision has been made here to choose the approach of storing a single IP address, so that, peer **A** sends a single IP address to peer **B**. peer **B** can only change its own address, and it has to wait for an address update from peer **A** with a new IP address. This approach is suitable for multihoming and mobility.

2- Direct or Indirect indication:

In indirect approach, a mechanism is needed to detect or test the path of the other peer address in order to assure the change of that address. This mechanism may cause confusion with the detection of liveness of that peer, and may needed to try all possible local IP addresses for each remote address. While, in direct indication, only what is needed is to send a message with the address change.

Thus, the our design decision here was to choose the direct indication approach. The technique used for direct indication in KEPSOM is to send a CME message including Update_Address payload, which contains the new IP address. This payload must be authenticated to verify that this change is actually legitimate change done by the other peer.

3- Disconnecting:

If it is happened that any node working in any scenario is going to be disconnected for some time, then there must be a way in which no data traffic should be sent during the disconnecting time to this node. To solve this problem, it is possible that this node send a message to inform the responder that it will going to be disconnected for some time. The responder would then temporarily disable the SA for a specified time until the disconnected end sends a new message with its address during the allowed time. Otherwise, the responder should delete the session of that SA. The details of the technique used in KEPSOM to solve the disconnecting case is as follows:

- A message should be sent from the disconnected peer to the other end. This message contains a payload called "Update_Address".
- The Update_Address payload contains zero address value, and includes the allowed time for disconnecting (the tagged lifetime value in the lifetime field in HDR payload, which is explained latter, is used here as a default value).
- When the disconnected peer wants to reconnect again within the allowed time, then a new Update_Address payload must be sent.
- The new Update_Address payload must contain the old or the new IP address.
- An address flag field of one bit length must be set if the address is a new one, otherwise it must be cleared.
- The lifetime field must be zero for the payloads of a new IP address.

From technical point of view, this technique could save the bandwidth, since there is no need to transmit any data traffic during the disconnecting time. The technique is also quite suitable for multihomed laptop scenario, especially when it is required to disconnect the laptop for some time in order to reconnect it with a new IP address. The design takes in its consideration that the allowed time for disconnecting could be a few minutes, a few hours, or even more.

5. USING KEPSOM IN HYBRID WIRELESS NETWORKS

Hyprid wireless networks are becoming more widely available. This created the need for securing such networks to provide users with authentic communications, secure and robust information exchange, and efficient security mechanism. Basically, there are two types of wireless networks [10]:

1- Infrastructure WLANs: This is also called Base-Station (BS) oriented networks, in which mobile hosts (MH) communicate with base stations (single-hop).
2- Noninfrastructure WLANs (Ad Hoc WLANs): In this type of structure the MHs communicate with one another directly. Ad Hoc networks do not have any centralized administration nor standard support services regularly available on the network to which the hosts may normally be connected.

There have been many techniques or concepts proposed for supporting hybrid wireless network communication protocols. Chang et. al.[10] proposed hybrid wireless network communication protocol to combine the advantages of BS-oriented and Ad Hoc wireless networks. The approach allows two mobile hosts to communicate directly (one-hop direct-transmission) or through another mobile host (two-hop direct-transmission) within the BS-oriented networks. According to the communicating techniques proposed by this approach, some features have been added to KEPSOM to make it suitable for key exchange in hybrid wireless networks. In BS-oriented networks, BS manages all the MHs within the cell area and handles handoff procedures [11]. If BS fails for any reason all MHs under that BS could not communicate with others. Therefore, to increase the reliability and efficiency of the BS-oriented network, the approach adds MH-to-MH direct transmit capability, and restricts it to at most two hops to insure not to increase the complexity of the protocol too much.

In Ad Hoc networks, the problem is that, building or maintaining a connection is not easy. The connection will be disrupted any time one MH moves out of the connection range [12] [13] [14]. Each MH forwards the packets hop by hop until the packet arrive the destination MH_n. If any MH moves out of the coverage area, then the packets must be forward through another path. Since, MH moves randomly, the frequent rerouting will decrease the throughput and increase the delay. Forced termination would occur if there is no MH found to act as intermediate node for forwarding the packets to the destination.

To maintain the communication between the MHs two different methods are proposed; one-hop direct-transmission and two-hop direct transmission within BS-oriented environments. In these two methods, the MHs could communicate with each other over the wireless media, without any support from the infrastructure network components within the signal transmission range. If the transmission range is less than the distance between the two MHs, then it could be changed back to the BS-oriented system. Therefore, the MH is able to operate in both Ad Hoc and Bs-oriented WLAN environments. Direct transmission defines the situation where two MHs communicate directly or use a third MH as a relay without the help of BS. When the initiator broadcasts the connection request message, both the BS and the MH within the initiator signal covering area would receive this message. Each MH receiving the message will check the destination ID. If the destination ID match itself, the transmission would use the one-hop direct-transmission mode; else, each receiving MH must check its neighbor database to see if the destination is currently a neighbor of itself. Otherwise, the BS would be used for connection. **Figure (4)** illustrates the flow chart which represents this procedure. When the destination moves out of the covering area, the BS would have to take over. On the other hand, when the MH moves into the covering range, the responder stop going through the BS and changes to one-hop direct-transmission.

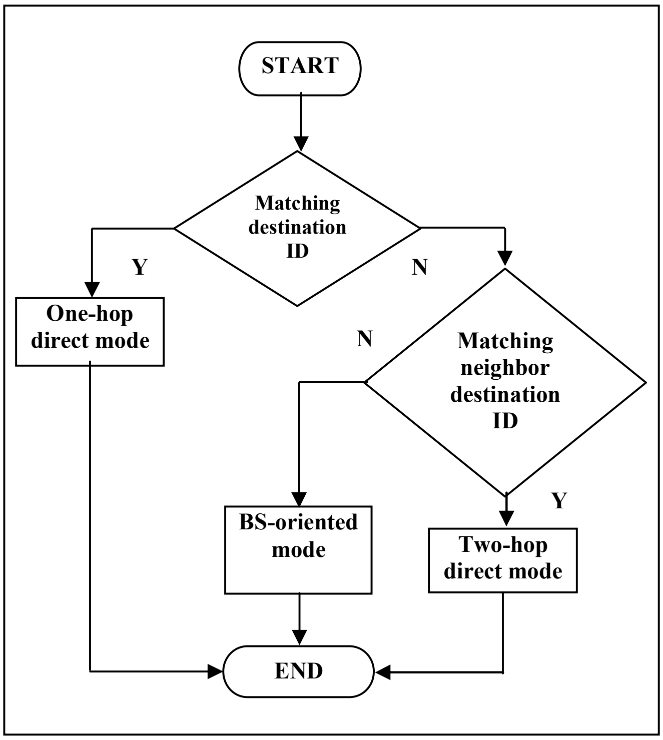

Fig. 4. Connection mode flow chart

All the messages in KEPSOM include HDR payload as described earlier. The format of the HDR payload contains a reserved field of 1 octet length. KEPSOM can make use of that field in order to be suitable for key exchange in hybrid networking. Thus, the reserved field has been changed to be "Mode Type". The Mode Type field is one octet length. This field represents the mode of the connection in the hybrid network.

Key exchange can be established using KEPSOM with one-hop direct-transmission, two-hop direct-transmission, or BS-oriented mode in hybrid wireless network. There is no need to make any modification to any message in KEPSOM. The same MFE messages and the same CME messages can be used. They are authenticated to verify that the changes in the mode is actually legitimate change done by the other MH, and cryptographically protected with the negotiated key. The only modified payload is the HDR payload, by making use of a reserved field in this payload to indicate the mode of the transmission. Thus, KEPSOM can be used to exchange keys in hybrid wireless network as long as the mobile hosts can communicate each other directly or through an intermediate MH or through a base station.

6. SECURITY ANALYSIS

The core cryptographic design of KEPSOM follows that of SIGMA ("SIGn-MAc") family of key exchange protocols [15]. The security proof of Basic-SIGMA protocol (Σ_o) has been introduced in [16]. So, to analyze the security of KEPSOM it is needed to extend the proof and the analysis of Σ_o to deal with the proposed protocol.

The analysis of a key exchange protocol needs to formalize a security model for that protocol, and for the attacker, taking in consideration any additional important elements. The formalism based on the approach of [3] has been used in modeling and analyzing the security of KEPSOM. In this approach a general framework for studying the security of session-based-multi-port protocol over insecure channels was introduced and refined to better fit the needs of practical key exchange protocols. In [16], the authors modeled key-exchange (KE) protocols as multi-party protocols, in which each party runs one or more copies of the protocol. Each run at specified party results in a local procedure called a session. This session produces outgoing messages and processes incoming messages. In the case of key-exchange, a session is intended to agree on "session key" with one another party (the peer to the session) and involves the exchange of the messages with that party. The activation of a KE session at a party has three input parameters (P, s, Q), where, P: is the local party at which the session is activated, s: is a unique session identifier, and Q: the identity of the intended peer to the session.

There is also a fourth input tag field, specifying whether the party is the initiator or responder in the exchange, however, this field is ignored because it has no effects on the security requirements. The output of a KE session at a party P consists of a public triple (P, s, Q) that identifies the session, and a secret value called the session key. As mentioned earlier, the security analysis of Σ_o can be extended to the elements found in KEPSOM protocol and not included in the basic protocol Σ_o. In the rest of this section, the security of different variants of KEPSOM will be considered. These variants include the following:

a- Eliminating the initiator and responder tags:

In Σ_o the initiator and responder include under their signature and MAC a special tag "0" and "1", respectively. KEPSOM do not use these tags. So, it is needed to show that if a protocol, say Σ_o', is defined identically to Σ_o except for lack of these tags is still secure. The tag "1" which is includede under the signature of Q has been used to argue that SIG_Q("1", s, g^x, g^y) received by P in the response message of session s is the only signature that Q could have produced under session s . Hence, the responder message must have come from Q. Assume that the tag "1" is omitted from this signature, as it would be in Σ_o'. Then P could receive a response message $SIG_Q(s, g^x, g^y)$ which has been taken from the finish message sent by Q in a session (Q, s), where Q was activated as the initiator. In this case, however, we know that before sending the finish message, with the above signature, (Q, s) should have received a valid response message which includes a legal signature $SIG_J(s, g^y, g^x)$ from some party J, as well as corresponding $MAC_{k1}(s, J)$ for $k_1 = PRF_{g^{xy}}(1)$. Since we know that g^y was chosen by Q itself and g^x was chosen by P, then no uncorrupted party could have chosen g^x except for negligible (collision) probability.

Moreover, in the proof of Σ_o it was shown that, even without the tags, it is not feasible for the attacker to make (R_o, s_o) complete with ID other than I_o or R_o, since the only available MAC value are on I_o and R_o. Thus, if (I_o, s_o) completes with peer $R_o \neq I_o$ then it is needed to use signature-based argument to show that replaying $MAC_{k1}(s_o, R_o)$ in the finish message does not help. Indeed the finish message will also have to carry a signature $SIG_{Ro}(s_o, g^y, g^x)$ where the g^x is the DH exponent received by (R_o, s_o) in the start message. On the other hand, the only signature produced by R_o in session s_o is $SIG_{Ro}(s_o, g^x, g^y)$, where g^y chosen by R_o itself after receiving g^x and independently of this value. Therefore, except for negligible collision probability, $g^x \neq g^y$ and $SIG_{Ro}(s_o, g^y, g^x)$ can never be produced by R_o.

b- Putting the signature function under the authentication function:

In message (2) of KEPSOM, which represents the response to message (1), the signature payload and the authentication payload is sent separately, just like Σ_o. In the later two messages of KEPSOM the signature payload is computed under the MAC operation. That is, the sent message is not SIG( ), MAC( ), as in Σ_o, but rather sends the value MAC(…, SIG( )). Fortunately, the analysis of the protocol when the SIG function goes under the authentication function is essentially the same as the simplified Σ_o version. The analysis adaptation is straightforward and is based on the following simple fact.

Lemma: If SIG is a secure signature scheme and MAC is a secure message authentication function, then it is infeasible for an attacker to find two different messages M and M' such that for a randomly chosen secret MAC-key k_1 the attacker can compute $MAC_{k1}(SIG(M'))$ even after seeing $MAC_{k1}(SIG(M))$.

The implementation of the above changes preserves security, in which the SIG function is included under the authentication function, and the PRF function itself is used to implement the MAC function. Noting that KEPSOM also make sure that the keys generated, using PRF function differ from each other.

c- A four messages variants:

The protocol Σ_o uses only three messages in key exchange. While KEPSOM uses four messages for this exchange. So it is needed to show the effect of the fourth message on the security of the protocol. Let Σ_1 be a four message protocol, and is similar to Σ_o except that the responder delays its authentication to the fourth message. The protocol is:

I→R: s, g^x

R→I: s, g^y

I→R: s, ID_i, SIG_i("0", s, g^y, g^x), MAC_{k1}("0", s, ID_i)

R→I: s, ID_r, SIG_r("1", s, g^x, g^y), MAC_{k1}("1", s, ID_r)

The security analysis of Σ_1 is similar to that of Σ_o. It follows the same basic logic and structure of that proof but it requires some changes due to the addition of the fourth message, and the fact that the responder authenticates after the initiator. But this fact is not true in KEPSOM, since in message (2) the responder authenticates before the initiator. Thus, in KEPSOM the security analysis needs only to recognize what message four can do. The four-message variant adds important property to the security of the protocol. Since, whenever a part P completes a KEPSOM exchange with peer Q, it is guaranteed that Q has initiated an exchange with P and is aware of P's existence.

d- The nonces variants:

The other variant in KEPSOM is the use of nonces, which serve two purposes; first, it allows the initiator to reuse the same DH value across different sessions, while ensuring that the resulting session key will be different. Secondly, it can be used to differentiate between different parallel sessions. In addition KEPSOM adds a preliminary cookie mechanism for DoS protection.

e- Encrypting the identities:

The other variant in KEPSOM is the use of encryption algorithm to encrypt the information in message (3) and (4). The main goal of this use of encryption is to protect the identities of the peers from disclosure over the network. The addition of encryption preserves the security of the protocol. In message (3), the initiator uses the key derived by using PRF function from the two DH values and the two nonces to encrypt his identity and service request. The initiator's nonce is used to ensure that this session key is unique. The initiator can validate the responder's identity before sending his own one. Then the initiator identifying information is sent encrypted. Thus, privacy is protected from both passive and active attackers. In message (4) of KEPSOM, all the information is encrypted and authenticated using keys derived also from the two nonces and the two values of DH. The initiator can verify that the responder is present and participating in the session by decrypting the message and verifying the enclosed signature.

7. EXPERIMENTAL RESULTS

The testbed used to implement KEPSOM is a wireless access network linked to Internet Service Provider. The wireless communication architecture which has been used in this experiment is to accommodate the requirements of key exchange in IP-based mobile network. The following cases have been tested experimentally:

a- KEPSOM has been configured and executed on different PCs, each PC access the wireless network through its own fixed IP address.

b- In the second case, the IP address of one of the communicated computers was changed manually during the key exchange operation. This case looks like a multihomed laptop scenario. Computers have multiple interface cards and therefore several ways to connect to a network. For example, a laptop can be disconnected from a fixed Ethernet, and then use a campus WLAN. Then later moving to other place using WLAN but with other IP address.

c- The third case represents a mobile network in a more realistic manner, in which the IP address is changed automatically due to mobility. In this case, the address of one of the communicating computers was changed remotely during the key exchange session without the need of restarting a new KEPSOM session. The automatic change in address has been occurred by asking the administrator to change the IP address of one end remotely during the key exchange operations.

In normal case, the change in IP address may cause the two communicated endpoints to disconnect. However, KEPSOM was able to check and detect the changes in IP address and to accept the connection with the new IP address in a secure manner, without the need to restart with a new key exchange session unless lifetime is finished. In all the test scenarios, if an SA bundle has been inactive for a long time and if an endpoint would not initiate the SA in the absence of traffic, then the endpoint close the SA when its lifetime expired.

In general, the traffic in networks depends tremendously on the speed of the network itself, the bandwidth, the congestion in the network, and the specification of the communicated equipments. Therefore, lifetimes should be chosen carefully.

The devices used in this experiment were:

- Pentium IV desktop PC of 1.7 GHz processor having RAM of 256 MB, with wireless LAN network adapter (802.11g Wireless LAN PCI). The operating system is Windows XP Service Pack2.
- Pentium IV laptop PC of 1.7 GHz processor having RAM of 128 MB. The operating system is Windows XP (Home Edition).
- 2.4 GHz band wireless access point, supporting 11 and 54 Mbps.

To measure and calculate the time delay for the key exchange, it is required to check the execution time needed to prepare the payloads of each message before sending it to the other end, and the time needed to analyze and check the received messages. **Table 1** shows the average required execution time for each MFE and rekeying message (preparing and analyzing) on both ends and the total length for each message.

Table 1. Average execution time for the MFE and rekeying messages

Message	Ts (msec)	Tr (msec)	Tt (msec)	T (msec)
Message 1	1.5	0.5	20	22
Message 2	1.5	1.5	27.8	30.8
Message 3	2.5	0.5	32.1	35.1
Message 4	2.5	0.5	22.9	25.9
Rekeying	1.0	0.6	29.4	31.0

For real time monitoring, the most important parameter to assess system performance is the time delay. The time delay defined in [17] as "the measure of the elapsed time (latency) between messages exchanged on the link". Thus, to check the time delay for each message it is required to define the following variables:

T_s = time required to prepare the message to be sent.
T_r = time required to analyze the received message.
T_t = time required to transmit the message.
T = total time = $T_s + T_r + T_t$

The time required for transmitting each MFE and rekeying message has been measured for several times. The average value for each message is indicated in **Table 2**, which includes also the total time required for each message. Noting that the bit rate of the wireless network used in the test is 2 Mbps.

Table 2. Total required time for the MFE and rekeying messages

Message	**Execution time (msec)**		**Length (octets)**
	Prepare to send	**Analyze on receive**	
Message 1	1.5	0.5	260
Message 2	1.5	1.5	400
Message 3	2.5	0.5	480
Message 4	2.5	0.5	312
Rekeying	1.0	0.6	448

From **Table 2** the total required time to exchange the keys is about 114 msec. While, the total required time for rekeying is about 31 msec. This represents about 27% of the total required time for exchanging the keys according to the two roundtrips of the MFE. **Figure (5)** shows the time delay verses information field of different sizes.

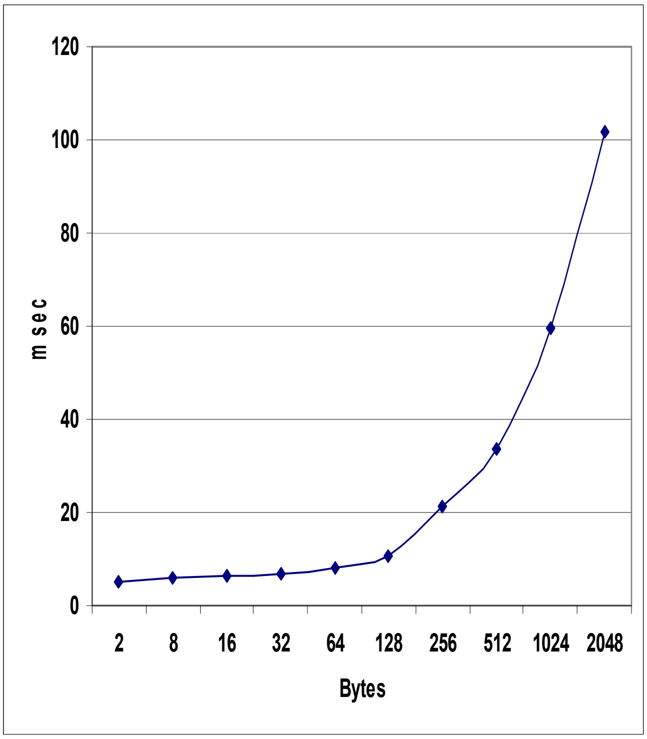

Fig. 5. Time delay vs. information field size

Moreover, the required time to detect the change in IP address and to update this change which occurred due to mobility or multihoming has been measured also. The total time required to detect and to update the IP address is about 2.2 msec. The average measured time to transmit the update message is about 9 msec. Thus, the total time required to complete the update process of the new IP address is about 11.2 msec. Comparing this time with the total time required to establish a new SA sessions, we can get the following:

$$\frac{\textbf{Time to update the IP address}}{\textbf{Total key exchange time}} \times 100 = \quad 9.8\%$$

This leads to the result that the performance of KEPSOM is much better than those protocols which need to establish a new SA session in each change in IP address.

The effect of KEPSOM on the network has been checked using WINDOWS XP administrator performance tool. Several tests were made during the execution of KEPSOM to measure that effect. Because the protocol starts at one end and continues on other end, hence in each test, the following events have been taken into consideration: start connection (connect), send and receive the MFE messages (message 1, message 2, message 3, message 4), send and receive the initiator rekeying message and the responder rekeying message, and closing the connection (disconnect). For example, **Figure (6)** shows the IP sent datagrams. Datagrams Sent per second is the rate, in incidents per second, at which IP datagrams were supplied for transmission by local IP user-protocol. Another test is to measure the datagrams received per second which is the rate, in incidents per seconds, at which IP datagrams are received from the interfaces. In both tests, the rate of the datagrams sent and received is similar for all of the MFE messages and also for the rekeying messages. But it differs in both the "connect" and "disconnect" events. This is because the connection establishment needs to call some functions that are more complicated than the functions called in sending or receiving the messages.

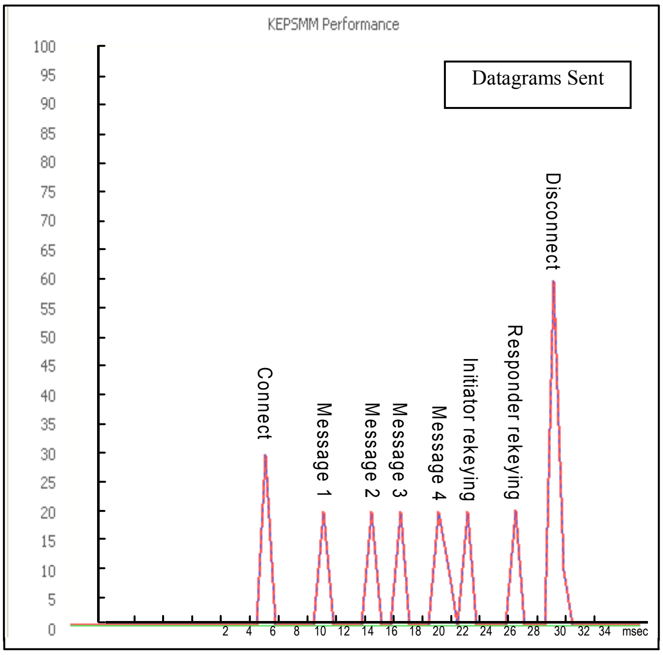

Fig. 6. IP Sent datagrams

8. CONCLUSION

The designed protocol supports any change in IP address due to mobility and multihoming. KEPSOM can also be used to support key exchange in hybrid wireless network, in which the mobile node can operate in both Ad Hoc and BS-oriented wireless network environments using three transmission modes; one-hop direct-transmission, two-hop direct-transmission, and BS-oriented transmission modes. From experimental tests, it was found that the total required time for rekeying is about 27% of the total required time for exchanging the keys. Also, the total time required to complete the update process, when an IP address changes (detect the new address, prepare and transmit the update message, and updating the address) was found to be less than 10% of the total required time to establish a new SA sessions. This leads to the conclusion that the performance of KEPSOM is much better than those protocols which need to establish a new SA session in each change in IP address.

REFERENCES:

[1] C. Kaufman, "Internet Key Exchange (IKEv2) Protocol", Internet draft, draft-ietf-ipsec-ikev2-12.txt, IETF, Work in Progress, Jan. 2004.

[2] W. Aiello, S. M. Bellovin, M. Blaze, R. Canetti, J. loannidis, A. D. Keromytis, and O. Reingold “Efficient, DoS-Resistant, Secure Key Exchange for Internet Protocols”, CCS'02. Nov. 18-22, 2002, Washington, DC USA Copyright 2002 ACM.

[3] R. Canetti and H. Krawczyk, “Analysis of key-exchange protocols and their use for building secure channels”, Proc. of the Eurocrypt conference, May 2001.

[4] P. Hoffman, “Features of proposed processors to IKE”, Internet draft (work in progress), (draft-ietf-ipsec-soi-features-01.txt), May 2002.

[5] P. Eronen, “Mobility Protocol Options for IKEv2 (MOPO-IKE)”, Internet draft, Network Working Group, draft-eronen-mobike-mopo-01.txt, Work in Progress, Oct. 2004.

[6] T. Kivinen, and H. Tschofenig, “Design of MOBIKE protocol”, Internet draft (work in progress), (draft-ietf-mobike-design-01.txt), Dec. 2004.

[7] T. Kivinen, “MOBIKE Protocol”, Internet draft, IKEv2 Mobility and Multihoming(MOBIKE), draft-kivinen-mobike-protocol-00.txt, Work in Progress, Feb. 2004.

[8] T. Kivinen, “Design of The MOBIKE Protocol”, Internet draft, IKEv2 Mobility and Multihoming (MOBIKE), draft-ietf-mobike-design-00.txt, Work in Progress, June. 2004.

[9] F. Dupont, “Address Management for IKE Version 2”, Internet draft, Network Working Group, draft-dupont-ikev2-addrmgmt-06.txt, Work in Progress, Oct. 2004.

[10] R. S. Chang, W. Y. Chen, and Y. F. Wen “Hybrid wireless network protocols”, IEEE Trans. On Vehicular Tech., Vol. 52, No. 4, Jul. 2003.

[11] D. B. Johnson, and D. A. Maltz, “Protocols for Adaptive Wireless and Mobile Networking”, IEEE Personal Communications, Vol. 3, No. 1, pp. 34-42, Feb. 1996.

[12] P. Bhagwat, and C. E. Perkins, “Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers”, SIGCOMM Conf. Process, pp. 234-244, Sep. 1994.

[13] C. C. Chiang, and M. Gerla, “Routing and multicast in multihop, mobile wireless networks”, Proc. of the IEEE ICUPC, pp. 546-551, 1997.

[14] D. B. Johnson, “Routing in Ad Hoc Networks of Mobile Hosts”, Proc. of the IEEE Workshop on Mobile Computing Systems and Applications, Dec.1994.

[15] H. Crawczyk, “SIGMA: the ‘SIGn-and-MAc’ Approach to authenticated Diffie-Hellman protocol”, In Proc. of the Crypto Conference, 2003.

[16] R. Canetti and H. Krawczyk, “Security analysis of IKE’s signature-based key-exchange protocol”, In Proc. of the Crypto Conference, Aug. 2002.

[17] W. T. Strayer, and A. C. Weaver, “Performance Measurements of data transfer services in MAP”, IEEE Network, vol. 2, no. 3, May 1988.