An Adaptive Kerberos Authentication Protocol With Digital Envelop Technique

M. E. El-Hennawy
El-shorouk Academy, Institute of Computers and Information Technology
mhennawy@ad.gov.eg

M. M. Kouta
Professor of Computer Science
Arab Academy for Science and Technology

Abstract: *The Kerberos Authentication Protocol, developed at MIT, has been widely adopted by other organizations to identify clients for network services across an insecure network to protect the privacy and integrity of communications with those services. While Version 5 has been invented (specified in RFC1510) to overcome Version 4 environmental shortcomings and technical deficiencies, it has still some issues to be managed. One of the basic initials of the Kerberos protocol in the inter-realm environment is the process of sharing keys among realms to guarantee interoperability between them. This paper provides a proposal for simplifying this process and allowing shared inter-realm keys through the employment of the digital envelop technique.*

1. INTRODUCTION

The past two decades have seen an enormous increase in the development and use of networked and distributed systems, providing increased functionality to the user and more efficient use of distributed resources. To obtain the benefits of such systems parties will cooperate by exchanging messages over networks. The parties may be users, hosts or processes; they are generally referred to as principals. A distributed system - a collection of hosts interconnected by a network - poses some intricate security problems. A fundamental concern is authentication of local and remote entities in the system. In a distributed system, the hosts communicate by sending and receiving messages over the network. Various resources (like files and printers) distributed among the hosts are shared across the network in the form of network services provided by servers. Individual processes (clients) that desire access to resources direct service requests to the appropriate servers **[1]**. A distributed system is susceptible to a variety of threats mounted by intruders as well as legitimate users of the system. We identify two general types of threats. The first type, host compromise, refers to the subversion of individual hosts in a system. The second type, communication compromise, includes threats associated with message communications. We subdivide these into: eavesdropping of messages transmitted over network; arbitrary modification, insertion, and deletion of exchanged messages; and replay of old messages.

A considerable number of authentication protocols have been specified and implemented to overcome such threats. An authentication protocol is a sequence of message exchange between principals that either distributes secrets to some of those principals or allows the use of some secrets.

Kerberos, as an authentication protocol, involves a user and four computer principals: a client; a server with whom wishes to communicate; and two trusted servers. The first is known as a Ticket-Granting Server and the second is the Key Distribution Centre. The full protocol has three parts each consisting of two messages between the client and each of the servers in turn.

This paper includes 6 sections. Section 2 describes the Kerberos model. Section 3 discusses the inter-realm administrative environment as it is crucial for exchange between realms. Furthermore, section 4 introduces the digital envelop technique. Section 5 states the proposed inter-realm process that can be used to solve the problem of inter-realm key exchange by means of digital envelop technique. In section 6 the paper concludes with expected future work.

2. KERBEROS MODEL

The Kerberos Authentication Service was developed by the Massachusetts Institute of Technology (MIT) to protect the emerging network services provided by Project Athena. The goal of Project Athena is to create an educational computing environment based on high-performance workstations, high-speed networking, and servers of various types. Researchers envisioned a large-scale (10,000 workstations to 1,000 servers) open network computing environment in which individual workstations can be privately owned and operated **[1]**. Versions 1 through 3 were used internally. Although designed primarily for use by Project Athena, Version 4 of the protocol has achieved widespread use beyond MIT. Version 5 of the Kerberos protocol incorporates new features suggested by experience with Version 4, making it useful in more situations **[2]**.

Kerberos provides a means of verifying the identities of principals, (e.g. a workstation user or a network server) on an open (unprotected) network. This is accomplished without relying on authentication by the host operating system, without requiring physical security of all the hosts on the network, and under the assumption that packets traveling along the network can be read, modified, and inserted at will. Kerberos performs authentication under these conditions as a trusted third party authentication service by using conventional (shared secret key) cryptography **[3]**. To achieve this, the client (initiating party) conducts a three-party message exchange to prove its identity to the server (the contacted party). The client proves its identity by presenting to the server a ticket (shown in figures as $T_{c,s}$) which identifies a principal and establishes a temporary encryption key that may be used to communicate with that principal, and an authenticator (shown in

figures as $A_{c,s}$) which proves that the client is in possession of the temporary encryption key that was assigned to the principal identified by the ticket. A principal is the basic entity that participates in authentication. Each principal is uniquely named by its principal identifier. The authenticator prevents an intruder from replaying the same ticket to the server in a future session.

Tickets are issued by a trusted third party Key Distribution Center (KDC). The KDC, proposed by Needham and Schroeder **[4]**, is trusted to hold in confidence secret keys known by each client and server on the network. The key shared with the KDC forms the basis upon which a client or server believes the authenticity of the tickets it receives. A Kerberos ticket is valid for a finite interval called its lifetime. When the interval ends, the ticket expires; any later authentication exchanges require a new ticket from the KDC. Each installation comprises an autonomously administered realm and establishes its own KDC. Clients in separate realms can authenticate to each other if the administrators of those realms have previously arranged a shared secret.

2.1. THE INITIAL TICKET EXCHANGE

Figure 1 shows the messages required for a client to prove its identity to a server. The basic messages are the same for Versions 4 and 5 of Kerberos though the details of the encoding differ. In the first message the client contacts the KDC, identifies itself, presents a nonce (a timestamp or other non-repeating identifier for the request), and requests credentials for use with a particular server. Upon receipt of the message, the KDC selects a random encryption key $K_{c,s}$, called the session key, and generates the requested ticket. The ticket identifies the client, specifies the session key $K_{c,s}$, lists the start and expiration times, and is encrypted in the key K_s shared by the KDC and the server. Because the ticket is encrypted in a key known only by the KDC and the server, nobody else can read it or change the identity of the client specified within it. The KDC next assembles a response, the second message, which it sends to the client. The response includes the session key, the nonce, and the ticket. The session key and nonce are encrypted with the client's secret key K_c (in Version 4 all fields are encrypted in K_c). Upon receiving the response the client decrypts it using its secret key (usually derived from a password). After checking the nonce, the client caches the ticket and associated session key for future use **[5]**.

In the third message the client presents the ticket and a freshly-generated authenticator to the server. The authenticator contains a timestamp and is encrypted in the session key $K_{c,s}$. Upon receipt the server decrypts the ticket using the key it shares with the KDC (this key is kept in secure storage on the server's host) and extracts the identity of the client and the session key $K_{c,s}$. To verify the identity of the client, the sever decrypts the authenticator (using the session key $K_{c,s}$ from the ticket) and verifies that the timestamp is current. Successful verification of the

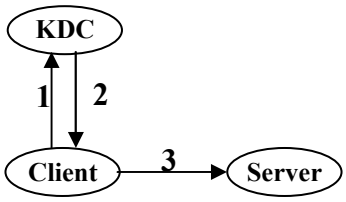
authenticator proves that the client possesses the session key $K_{c,s}$, which it only could have obtained if it were able to decrypt the response from the KDC. Since the response from the KDC was encrypted in K_c, the key of the user named in the ticket, the server may reasonably be assured that identity of the client is in fact the principal named in the ticket. If the client requests mutual authentication from the server, the server responds with a fresh message encrypted using the session key. This proves to the client that the server possesses the session key, which it could only have obtained if it was able to decrypt the ticket. Since the ticket is encrypted in a key known only by the KDC and the server, the response proves the identity of the server. For greater detail on the messages in Version 4 of Kerberos the reader is referred to **[5-7]**.

2.2. THE ADDITIONAL TICKET EXCHANGE

To reduce the risk of exposure of the client's secret key K_c and to make the use of Kerberos more transparent to the user, the exchange above is used primarily to obtain a ticket for a special ticket-granting server (TGS). The client erases its copy of the client's secret key once this ticket-granting ticket (TGT) has been obtained,

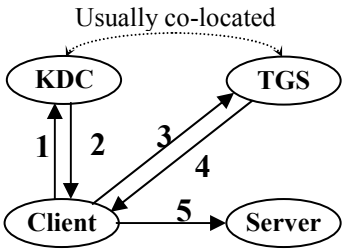
The TGS is logically distinct from the KDC which provides the initial ticket service, but the TGS runs on the same host and has access to the same database of clients and keys used by the KDC (see **Figure 2**). A client presents its TGT (along with other request data) to the TGS as it would present it to any other server (in an application request); the TGS verifies the ticket, authenticator, and accompanying request, and replies with a ticket for a new server. The protected part of the reply is encrypted with the session key from the TGT, so the client need not retain the original secret key K_c to decrypt and use this reply. The client then uses these new credentials as before to authenticate itself to the server, and perhaps to verify the identity of the server.

Once the authentication is established, the client and server share a common session key $K_{c,s}$, which has never been transmitted over the network without being encrypted. They may use this key to protect subsequent messages from disclosure or modification. Kerberos provides message formats which an application may generate as needed to assure the integrity or both the integrity and privacy of a message.



1. Client→ KDC: c, s, n
2. KDC → Client: $\{K_{c,s}, n\}K_c, \{T_{c,s}\}K_s$
3. Client→ Server: $\{A_c\}K_{c,s}, \{T_{c,s}\}K_s$

Figure 1: *Getting and using an Initial Ticket*



1. Client→ KDC: c, tgs, n
2. KDC → Client: $\{K_{c,tgs}, n\}K_c, \{T_{c,tgs}\}K_{tgs}$
3. Client→ TGS: $\{A_c\}K_{c,tgs}, \{T_{c,tgs}\}K_{tgs}$, s, n
4. TGS → Client: $\{K_{c,s}, n\}K_{c,tgs}, \{T_{c,s}\}K_s$
5. Client→ Server: $\{A_c\}K_{c,s}, \{T_{c,s}\}K_s$

Figure 2: *Getting a service ticket*

One of the basic facilities that Version 4 provides is the cooperation between authentication realms by allowing each pair of cooperating realms to exchange an encryption key to be used as a secondary key for the ticket-granting service. This pair-wise key exchange makes inter-realm ticket requests and verification easy to implement, but requires $O(n^2)$ key exchanges to interconnect n realms.

2.3. LIMITATIONS OF VERSION 4

Version 4 of Kerberos is in widespread use, but some sites require functionality that it doesn't provide, while others have a computing environment or administrative procedures that differ from that at MIT. As a result, work on Kerberos Version 5 commenced in 1989, fueled by discussions with Version 4 users and administrators about their experiences with the protocol and MIT's implementation. Version 4 suffers from both environmental shortcomings and technical deficiencies. Kerberos Version 4 was targeted primarily for Project Athena **[8]**, and as such in some areas it makes assumptions and takes approaches that are not appropriate universally. Environmental shortcomings can be referenced in **[2].** In addition to the environmental shortcomings, there are some technical deficiencies in Version 4 and its implementation **[9]**. Technical deficiencies can be referenced in **[10-13].**

2.4. CHANGES FOR VERSION 5

Version 5 of the protocol has evolved over the past two years based on implementation experience and discussions within the community of Kerberos users. Its final specification has reached closure, and a description of the protocol is available. Version 5 addresses the concerns described above and provides additional functionality. The Changes between Versions 4 and 5 can be referenced

in **[2] [14].** It covers the subjects of use of encryption, network addresses, message encoding, ticket changes, ticket lifetimes, and naming principals.

2.5 NEW PROTOCOL FEATURES IN VERSION 5

In addition to the changes discussed above, several new features are supported in Version 5.

Tickets: Version 5 tickets contain several additional timestamps and a flags field. These changes allow greater flexibility in the use of tickets than was available in Version 4. Each ticket issued by the KDC using the initial ticket exchange is flagged as such **[2]**.

Authorization data: Kerberos is concerned primarily with authentication; it is not directly concerned with the related security functions of authorization and accounting. To support the implementation of these related functions by other services; Version 5 of Kerberos provides a mechanism for the tamper-proof transmission of authorization and accounting information as part of a ticket **[2]**.

Pre-authentication data: In an effort to complicate the theft of passwords, the Kerberos Version 5 protocol provides fields in the initial- and additional-ticket exchanges to support password alternatives such as hand-held authenticators (devices which have internal circuitry used to generate a continually changing password) **[2]**.

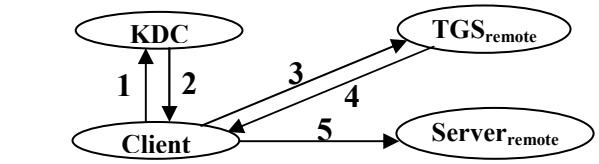
Subsession key negotiation: Tickets are cached by clients for later use. To avoid problems caused by the reuse of a ticket's session key across multiple connections, a server and client can cooperate to choose a new subsession key which is used to protect a single connection. This subsession key is discarded once the connection is closed **[2]**.

Sequence numbers: Kerberos provides two message formats for applications to protect their communications. iT uses a cryptographic checksum to insure data integrity and encryption to insure integrity and privacy **[2]**.

3. THE INTER-REALM ADMINISTRATIVE ENVIRONMENT

The Kerberos protocol is designed to operate across organizational boundaries. A client in one organization can be authenticated to a server in another. A realm is said to communicate with another realm if the two realms share an inter-realm key, or if the local realm shares an inter-realm key with an intermediate realm that communicates with the remote realm. An authentication path is the sequence of intermediate realms transverses in communicating from one realm to another **[3]**.

Version 4 provides cooperation between authentication realms by allowing each pair of cooperating realms to exchange an encryption key to be used as a secondary key for the ticket-granting service. A client can obtain tickets for services from a foreign realm's KDC by first obtaining a ticket-granting ticket for the foreign realm from its local KDC and then using that TGT to obtain tickets for the foreign application server (see **Figure 3**). This pair-wise key exchange makes inter-realm ticket requests and verification easy to implement, but requires $O(n^2)$ key exchanges to interconnect n realms (see **Figure 4**) **[5].**



1. Client→ TGS_{local}: $\{A_c\}K_{c,tgs}$,$\{T_{c,tgs}\}K_{tgs}$, tgs_{rem}
2. TGS_{local}→ Client: $\{K_{c,tgsrem}\}K_{c,tgs}$,$\{T_{c,tgsrem}\}K_{tgsrem}$
3. Client→ TGS_{remote}: $\{A_c\}K_{c,tgsrem}$,$\{T_{c,tgsrem}\}K_{tgsrem}$, S_{rem}
4. TGS_{remote}→ Client: $\{K_{c,srem}\}K_{c,tgsrem}$,$\{T_{c,srem}\}K_{srem}$
5. Client→ $Server_{remote}$: $\{A_c\}K_{c,srem}$,$\{T_{c,srem}\}K_{srem}$

Figure 3: *Getting a foreign realm service ticket*

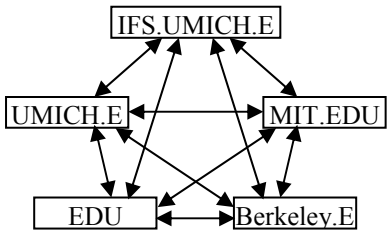


Figure 4: *Version 4 realm interconnections*

In Version 5, Kerberos realms cooperate through a hierarchy based on the name of the realm. A source realm is interoperable with a destination realm if it shares an inter-realm key directly with the destination realm, or if it shares a key with an intermediate realm that is itself interoperable with the destination realm. Each realm exchanges a different pair of inter-realm keys with its parent node and each child. This arrangement reduces the number of key exchanges to O(log(n)) **[2]**.

4. DIGITAL ENVELOP TECHNIQUE

4.1 MOTIVATION

The asymmetric-key cryptosystems offer the digital signature, which fulfills the authentication, the non-repudiation, and the message integrity protection mechanisms. They offer, also, the key management, which is the key factor for any encryption algorithm. The disadvantage is the processing overload. The symmetric-key cryptosystems offer the performance and for a reasonable key length, the secrecy. The disadvantage is the key management. The hybrid cryptosystems offer an optimized way to utilize both symmetric-key and asymmetric-key cryptosystems together **[16]**.

The hybrid cryptosystems can be implemented by using the Digital Envelope Technique **[15] [16].** Several hybrid schemes standards have been developed for the exchange of encrypted e-mail, including those for Privacy Enhanced Mail (PEM) **[17]**, Pretty Good Privacy (PGP) **[5]**, and Secure/Multipurpose Internet Mail Extensions (S/MIME) **[18].**

4.2 THE PRINCIPLES OF THE DIGITAL ENVELOPE TECHNIQUE

The Digital envelope technique is based on the combination of the benefits of both the symmetric and the asymmetric cryptosystems **[5]**.

To send a message from a sender to a receiver, the sender generates a secret session key and encrypts the message using his selected symmetric-key cipher algorithm, after that the sender encrypts the secret session key using asymmetric-key cryptosystem with the receiver's public key. The sender sends to the receiver an envelope containing the identities of the receivers in a special formatting, the encrypted secret session key, and the encrypted message. When the receiver wants to read the message he opens the envelope. He decrypts the secret session key, using asymmetric-key cryptosystem with his private key, and then decrypts the message, using the recovered secret session key. **Figure 5** illustrates the implementation of the confidentiality through the Digital Envelope Technique **[18-19**].

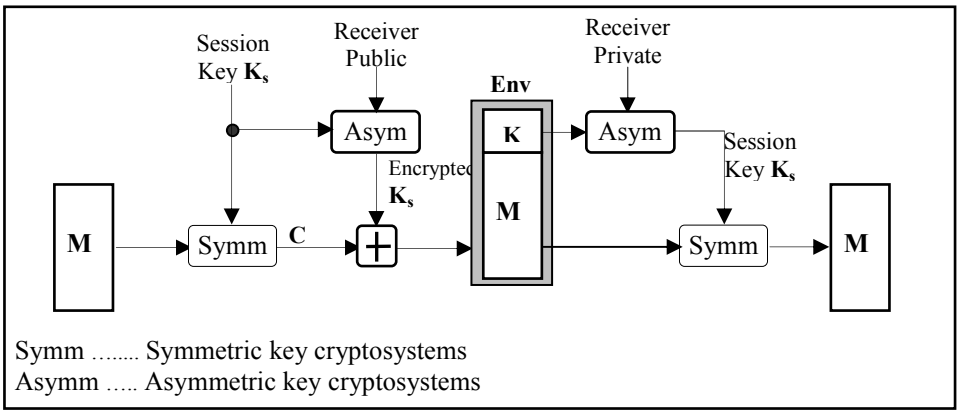


Figure 5: The Confidentiality through the Digital Envelop Technique

5. PROPOSED INTER-REALM KEY EXCHANGE PROPOSAL

Kerberos provides a mechanism for supporting inter-realm authentication. For two realms to support inter-realm authentication, Kerberos server in one realm should trust the Kerberos server of the other realm to authenticate its users. However, a requirement is: The Kerberos server in each interoperating realm

shares a secret key with the server in the other realm. However, as said before, the number of key that need to be exchanged between realms is at least O(log(n)).

To overcome the problem of Key exchange between various realms, the digital envelop technique can be applied under the following assumptions:

- Installing the identity and the public keys of all trusted realms in a local certification authority's database to allow using the public-key encryption.
- Introducing an initial sub-phase in the Kerberos protocol. It will be executed before stating inter-realm authentication can take place, and any time there is a need to refresh the shared keys between various trusted realms.
- In our proposal, it is not crucial to define the $ID_{remaote}$ in clear since it is encrypted inside the string
- The proposed sub-phase can be summarized as follows:

With these basic rules, a server wishing a service on a server in another realm needs a ticket for that server. The user's client follows the usual procedures as stated in **Figure 3**. Besides, after the first message received from client, the following steps are performed:

Sending Cycle:

- The local TGS generates the intended shared key (K_{Share}), and compose a message (shown in Figure 6 as M1) consisting of the remote realm identification address ID_{remote}, the local realm identification address ID_{local}, and a nonce, besides the K_{Share}.
- It encrypts the message using a generated session key (K_s), noting that this session key is generated to be used only once for this exchange and then destroyed.
- It encrypts the K_s using the remote realm public key ($E_{KPub\text{-}remote}$).
- It composes the digital envelop containing the encrypted session key (EK_s) and the M1, and sends it to the remote realm.

Receiving Cycle:

Remote TGS, when he received the digital envelop, to share the requested key with the local one could do the following:

- Extract the encrypted session key ($EK_{s)}$ and the encrypted message (M1) from the digital envelop.
- Recover the session key (K_s), by decrypting the received encrypted form (EK_s) using his private key ($E_{KPrive\text{-}remote}$).
- Recover the shared key ($\mathbf{K_{Share}}$), the timestamp (Nonce), ID_{local}, and the ID_{remote} (shown as M1 in Figure 6) using the recovered session key (K_s).

- Use ID_{remote} to verify that she is the proper receiver of the received message.
- Finally, he could use the recovered timestamp (nonce) for verifying the replay attack.
- Remote realm, then, can generate a message consisting of an encrypted message consisting of ID_{local} and f(Nonce+1) and encrypt it using the local public key $E_{KPub-local}$. The function f(Nonce+1) is an agreed upon function to assure the conjunction between the sent and received encrypted messages.
- The remote realm then builds-up the digital envelop and sends it to the local realm.
- The local realm verifies the nonce function, if verified the exchanges is guaranteed to be correct.

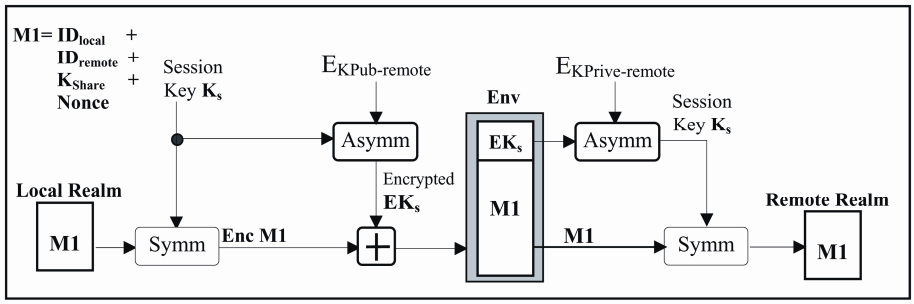


Figure 6: *The Shared key Exchange through the Digital Envelop Technique*

The exchanges can be described as followed:

1. $TGS_{local} \rightarrow TGS_{remote}$: { ID_{local}, ID_{remote} , K_{Share}, Nonce }$E_{KPub-remote}$
2. TGS_{remote}: [{ ID_{remote} , K_{Share}, Nonce }$E_{KPub-remote}$] $E_{KPrive-remote}$
3. $TGS_{remote} \rightarrow TGS_{local}$: { ID_{local} , f(Nonce+1) }$E_{KPub-local}$

6. CONCLUSIONS AND FUTURE WORK

We present in this paper the Kerberos Authentication Protocol, developed at MIT, has been widely adopted by other organizations to identify clients for network services across an insecure network to protect the privacy and integrity of communications with those services. While Version 5 has been invented (specified in RFC1510) to overcome

On the other hand, we believe the framework of the Kerberos Version 5 is flexible enough to adapt future requirements. we expect remote administration still need some investigation. The current protocol specifications do not specify an administrative interface to the KDC database. MIT's implementation provides a

sample remote administration program which allows administrators to add and modify entries and users to change their keys. We would like to standardize such a protocol. Some features we would like to add include remote extraction of server key tables, password "quality checks," and a provision for servers to change their secret keys automatically every so often.

References

[1] T. Y. C. Woo, and S. S. Lam, "Authentication for Distributed Systems", Proceedings of IEEE Computer Conference, pp:39-52, 1992.

[2] J. T. Kohl, B. C. Neuman, and T. Y. Ts'o, "The Evolution of the Kerberos Authentication Service", Proceedings of the Springs EurOpen Conference, 1991.

[3] J. Kohl, and B. C. Neuman, "The Kerberos Network Authentication Service", MIT Project Athena, Network Working Group, June, 1991.

[4] Roger M. Needham and Michael D. Schroeder, ''Using Encryption for Authentication in Large Networks of Computers,'' Communications of the ACM **21**(12), pp. 993-999 (December, 1978).

[5] William Stallings, "Network and Internetwork Security, Principles and Practice", Printic-hall, 1995.

[6] J. G. Steiner, B. C. Neuman, and J. I. Schiller, ''Kerberos: An Authentication Service for Open Network Systems,'' pp. 191-202 in Usenix Conference Proceedings, Dallas, Texas (February, 1988).

[7] S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Saltzer, Section E.2.1: Kerberos Authentication and Authorization System, M.I.T. Project Athena, Cambridge, Massachusetts (December 21, 1987).

[8] George A. Champine, Daniel E. Geer, and William N. Ruh, ''Project Athena as a Distributed Computer System,'' IEEE Computer **23**(9), pp. 40-50 (September 1990).

[9] S. M. Bellovin and M. Merritt, ''Limitations of the Kerberos Authentication System,'' Computer Communications Review **20**(5), pp. 119-132 (October 1990).

[10] Mer90. Ralph C. Merkle, ''Fast Software Encryption Functions,'' in Crypto '90 Conference Proceedings, International Association for Cryptologic Research, Santa Barbara, CA (August 1990).

[11] FIPS81. National Bureau of Standards, U.S. Department of Commerce, ''DES Modes of Operation,'' Federal Information Processing Standards Publication 81, Springfield, VA (December 1980).

[12] Koh89. John T. Kohl, ''The Use of Encryption in Kerberos for Network Authentication,'' in Crypto '89 Conference Proceedings, International Association for Cryptologic Research, Santa Barbara, CA (August 1989).

[13] Lom89. T. Mark A. Lomas, Li Gong, Jerome H. Saltzer, and Roger M. Needham, ''Reducing Risks from Poorly Chosen Keys,'' Operating Systems Review **23**(5), pp. 14-18 (December 1989).

[14] Stu92. Stuart G. Stubblebine and Virgil D. Gligor, ‘‘On Message Integrity in Cryptographic Protocols,’’ in Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, California (May 1992).

[15] Atkins D., Buis P., Hare C., Kelly R., Nachenberg C., Anthony B. Nelson, Phillips P., Ritchey T., Sheldon T., and Snyder J., “Internet Security, Professional Reference”, New Riders Publishing, Indianapolis, IN, (1997).

[16] M. E. El-Hennawy, M. M. El-Gendy, Y. H. Dakroury, F. S. Helail, M. M. Kouta, " A Proposal for A New Unconditionally Secure Hybrid Cryptosystem" Institute of Statistical Studies & Reserch Conference, November, (2000).

[17] Schneier, B., "Applied Cryptography Protocols, Algorithms and Source Code", In C. 2 ed. John Wiley and Sons, Inc., New York, (1996).

[18] (See [RFC-2612, -2631, -2632, -2633and -2634]).

[19] Bellovin, S. and Merritt, M. "Limitations of the Kerberos Authentication System", Computer Communications Review, October 1990.