An Authentication Protocol Based on Kerberos 5

Eman El-Emam [1], Magdy Koutb [2], Hamdy Kelash [3], and Osama S. Faragallah [3]
(Corresponding author: Osama S. Faragallah)

Egyptian Space Program, National Authority for Remote Sensing and Space Sciences [1]
El-Nozha El-Gedeeda, Cairo, Egypt
Department of Industrial Electronics & Control, Faculty of Electronic Engineering, Menouf, Egypt [2]
Department of Computer Science & Engineering, Faculty of Electronic Engineering, Menouf, Egypt [3]
(Email: {ademahmo, S.J.Shepherd}@bradford.ac.uk)
(Received Aug. 6, 2009; revised and accepted Feb. 12, 2010)

Abstract

We introduce some modifications to the widely deployed Kerberos authentication protocol. The principle's secret-key will be independent of the user password to overcome the weak passwords chosen by the network principal that are susceptible to password guessing attacks, the main drawback of the Kerberos protocol. Instead, the Kerberos Distribution Center saves a profile for every instance in its realm to generate the principle's secret-key by hashing the profile, and encrypting the output digest. Besides, the lifetime of the secret-key is controlled using the system clock. Triple-Des is used for encryption, SHA-256 for hashing, and Blum Blum Shub for random number generation.

Keywords: Access control, authentication protocols, authorization, computer network security, Kerberos

1 Introduction

An elaborate set of protocols and mechanisms have been created to deal with information security issues. The technical means to achieve information security in an electronic society are provided through cryptography. The cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, access control, and authentication. Confidentiality is a service used to keep the contents of information from all but those authorized to have it. There are numerous approaches to provide confidentiality, e.g. the mathematical algorithms which render data unintelligible. Data integrity is a service that addresses the unauthorized alteration of data. To assure data integrity, one must have the ability to detect data manipulation by unauthorized parties. Data manipulation includes insertion, deletion, and substitution. Access control is the ability to limit the access to authorized users and applications. To achieve this, each entity trying to gain access must first be identified, or authenticated, so that access rights can be assigned to individuals. Authentication is a service related to identification. It is a fundamental building block for a secure networked environment. If a server knows the identity of a client, it can decide whether to provide the service, whether the user should be given special privileges, and so forth. In other words, authorization and accounting schemes can be built on top of authentication resulting in the required security to the computer network system.

Protocols play a major role in cryptography and are essential in meeting cryptographic goals. We need protocols to apply cryptographic algorithms and techniques among the communicating parties. Encryption schemes, hash functions, and random number generation are among the primitives which may be utilized to build a protocol. A cryptographic protocol is a distributed algorithm defined by a sequence of steps precisely specifying the actions required of two or more entities to achieve a specific security objective. The whole point of using cryptography in a protocol is to detect or prevent attacks.

We will begin with describing the motivation for the Kerberos approach and its environment in Section 2. Then, we will present a brief overview of the related work in Section 3. After that, we will outline the Kerberos messages exchange and we will analyze the publicly released versions of Kerberos Version 4 and Version 5 in Section 4. While in Section 5, we will discuss the Kerberos drawbacks. Then, in Section 6, we will examine the details used in our proposed implementation, address its associated database, comparing it with the previous versions, list its security properties and describe our testing environment and our testing results. Finally, we will summarize our conclusions and our future work in Section 7.

2 Motivation

Today, more common in computer network architecture is a distributed architecture consisting of dedicated user workstations (clients) and distributed or centralized

servers. In this environment, network connections to other machines are supported. Thus, we need to protect user information and resources housed at the server. The authentication service in these environments can be achieved by using Kerberos. It is one of the most widely used authentication protocols. It addresses an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network.

Kerberos employs one or more Kerberos servers (KDC: Kerberos Distribution Center) to provide an authentication service. The overall scheme of Kerberos is that of a trusted third party that uses a protocol based on that proposed by Needham and Schroeder [12]. It is trusted in the sense that clients and servers trust Kerberos to mediate their mutual authentication.

Kerberos relies exclusively on symmetric encryption, making no use of public-key encryption. Most of the secure routing protocols rely on public key infrastructures (PKI) to authenticate communicating nodes. Although PKI is secure, it is based on asymmetric cryptography and hence requires excessive processing and communication resources [8]. This resource hungry feature makes PKI based systems more susceptible to Denial of Service attacks. In contra

3 Related Work

3.1 Kerberos History

Massachusetts Institute of Technology (MIT) developed Kerberos to protect network services provided by Project Athena. Several versions of the protocol exist; versions 1-3 occurred only internally at MIT. Many members of Project Athena contributed to the design and implementation of Kerberos [13]. In [5] there is a dialogue that was written in 1988 to help its readers understand the fundamental reasons for why the Kerberos V4 protocol was the way it was. It was amazing how much this dialogue was still applicable for the Kerberos V5 protocol. Although many things were changed, the basic core ideas of the protocol have remained the same. Miller and Neuman are the primary designers of Kerberos Version 4 with contributions from Saltzer and Schiller [22]. They published that version in the late 1980s, although they had targeted it primarily for Project Athena. Version 5, designed by Kohl and Neuman, appeared as RFC 1510 in 1993 [11] (made obsolete by RFC 4120 in 2005 [14]), with the intention of overcoming the limitations and security problems of version 4.

In 2007, MIT formed the Kerberos Consortium along with some of the major vendors and users of Kerberos such as Sun Microsystems, Apple, Google, Microsoft, etc., to foster continued development. The MIT Kerberos Consortium was created to establish Kerberos as the universal authentication platform for the world's computer networks.

3.2 Kerberos Security

Security of Kerberos has been analyzed in many works, e.g. [1, 2, 3, 10, 21, 23] and [24]. Most commonly analyzes identify certain limitations of Kerberos and sometimes propose fixes. This leads to the evolution of the protocol when a new version patches the known vulnerabilities of the previous versions. The current version Kerberos V5 is already being revised and extended [14, 18], and [17]. Butler, Cervesato, Jaggard, and Scedrov have analyzed portions of the current version of Kerberos and have formally verified that the design of Kerberos' current version meets the desired goals for the most parts [6]. Boldyreva and Kumar at 2007 take a close look at Kerberos' encryption and confirm that most of the options in the current version provably provide privacy and authenticity [4].

3.3 Kerberos Applications

Kerberos is also used in wireless applications. Erdem proposed a high speed 2G wireless authentication systems based on Kerberos [7]. He used DES, 3DES and AES as secret-key crypto algorithms. He also used SHA-1 message digest algorithm to hash the message blocks. Besides, Pirzada and McDonald discuss how Kerberos is used for authentication in mobile ad-hoc networks [16].

Kerberos is also introduced to be used in IPv6 networks. Sakane, Okabey, Kamadaz, and Esakix describe a method to establish secure communication using Kerberos in IPv6 networks [19]. They propose a mechanism to achieve access control using Kerberos and to deal with address resolution using Kerberos with modification.

Nitin et al. present an image based authentication system using Kerberos protocol at 2008 [15]. That paper is a comprehensive study on the subject of using images as a password and the implementation of Jaypee University of Information Technology (JUIT) Image Based Authentication (IBA) system called as JUIT-IBA using Kerberos protocol.

Kerberos has grown to become the most widely deployed system for authentication and authorization in modern computer networks. Kerberos is currently shipped with all major computer operating systems and is uniquely positioned to become a universal solution to the distributed authentication and authorization problem of communicating parties [9].

4 Kerberos Overview

4.1 Kerberos Authentication Dialogue

The Kerberos protocol allows a client to repeatedly be authenticated to multiple servers assuming that there is a long-term secret key shared between the client and Kerberos infrastructure. The client long-term secret key was generated using the client's password ([20] describes the password to key transformation technique that is presented by the standard specification). A simplified

overview of the Kerberos actions is shown in Figure 1. Exchange between the client and the Kerberos AS (Authentication Server) in Messages 1 and 2 are used only when the user first logs in to the system. Exchange between the client and the Kerberos TGS (Ticket Granting Server) in Messages 3 and 4 are used whenever a user authenticates to a new server. Message 5 is used each time the user authenticates itself to a server. And finally, Message 6 is the mutual-authentication response by the server.

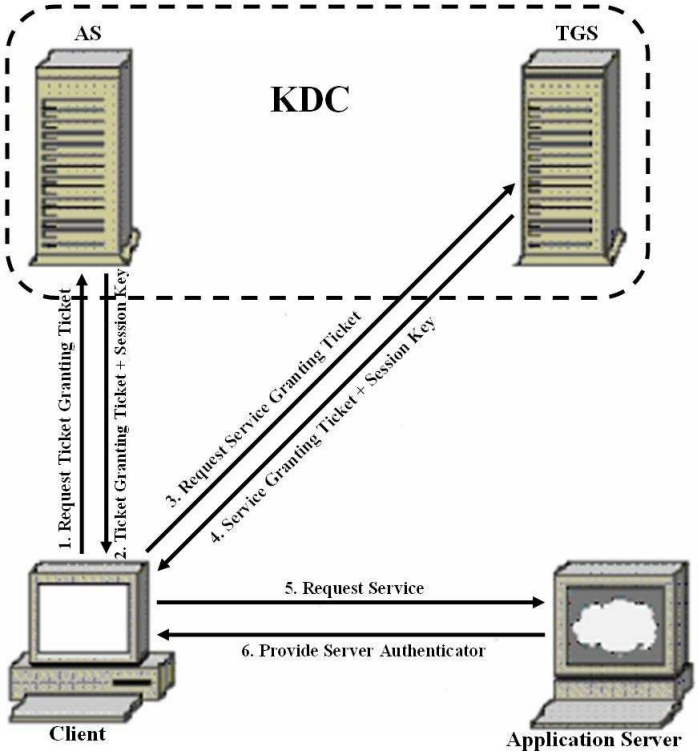

Figure 1: An overview of the Kerberos authentication dialogue

4.2 Kerberos 4 Message Exchange

If a client wishes to be authenticated to an application server, the standard run of Kerberos will accomplish this through three successive phases; the expected messages flow in these phases is shown in Figure 2 and proceeds as follows:

- In the first phase, the client sends a request to the Kerberos Authentication Server (AS) requesting a ticket granting ticket $\text{ticket}_{\text{tgs}}$ to be used in the second phase with the Ticket Granting Server (TGS). The AS is expected to reply with a message consisting of a ticket granting ticket tickettgs of lifetime lifetime_2 and an encrypted component containing a fresh session key $K_{\text{c,tgs}}$ to be shared between the client and the TGS. Another copy of this session key is contained in the Ticket granting ticket and is encrypted using the long-term secret key of the TGS K_{tgs} which is shared between TGS and Kerberos infrastructure (the AS can access the database of Kerberos infrastructure). The information directed to the client is encrypted under the client's long-term secret key KC.

- In the second phase, the client forwards the ticket granting ticket, along with an authenticator $Authenticator_{\text{C1}}$ encrypted with the session key $K_{\text{c,tgs}}$ obtained in the first phase, to the TGS, requesting a service ticket to be used in the third phase with the application server. The TGS is expected to reply with a message consisting of an application server ticket ticketV of lifetime lifetime4 and an encrypted component containing a fresh session key $K_{\text{c,v}}$ to be shared between the client and the application server. Another copy of this session key is contained in the application server ticket $ticket_{\text{V}}$ and is encrypted using the long-term secret key of the application server K_{V} which is shared between the application server and the Kerberos infrastructure (the TGS can access the database of the Kerberos infrastructure). The information directed to the client is encrypted with the session key of the first stage $K_{\text{c,tgs}}$.

- In the third phase, the client forwards the application server ticket $ticket_{\text{V}}$, along with a new authenticator $Authenticator_{\text{C2}}$ encrypted with the session key obtained in the second phase $K_{\text{c,v}}$, to the application server, requesting certain service. The application server ticket plus the secret session key are the client's credentials to be authenticated to a specific application server. If all credentials are valid, the application server will authenticate the client and provide the service. The acknowledgement message from the application server is optional and used only when the system requires mutual-authentication by the application server.

4.3 Kerberos 5 Message Exchange

Kerberos 5 authentication dialogue is shown in Figure 3. This is best explained by comparison with Version 4 (Figure 2). In Message (1), the following new elements are added:

- **Realm:** Indicates the realm of the client. Where the realm represents the nodes that are managed by a single KDC; i.e. share the same Kerberos database.

- **Options:** Used to request that certain flags be set in the returned ticket. These flags are an added feature in Kerberos 5.

- **Times:** Used by the client to request the following time settings in the ticket:

  1) **from:** the desired start time for the requested ticket.

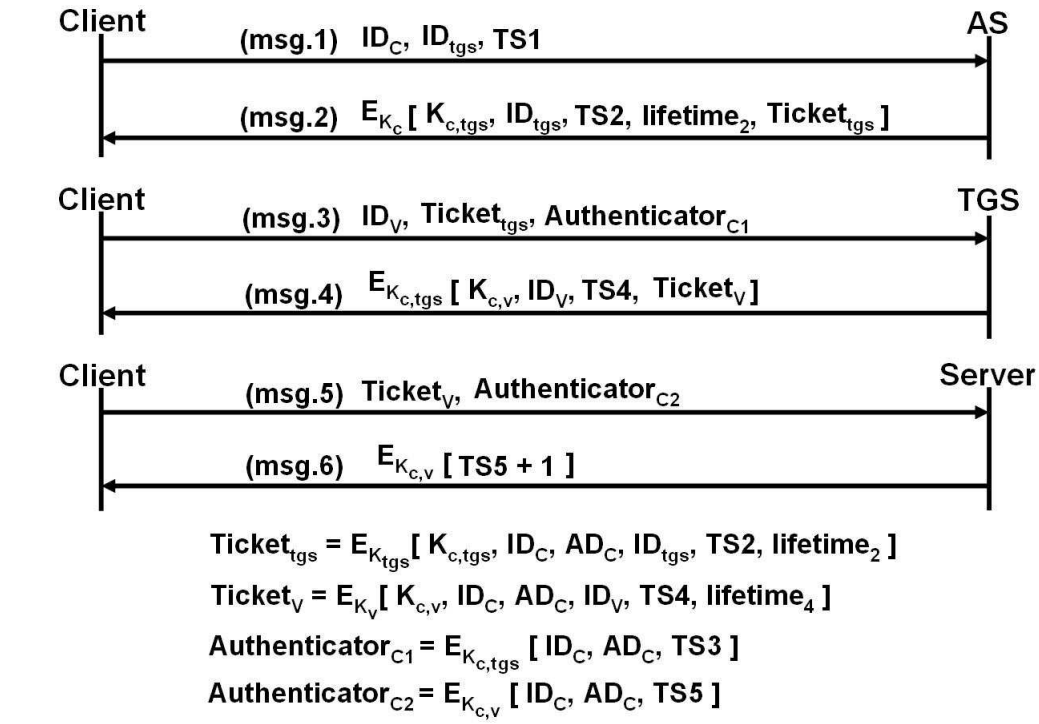

Figure 2: Kerberos 4 authentication dialogue

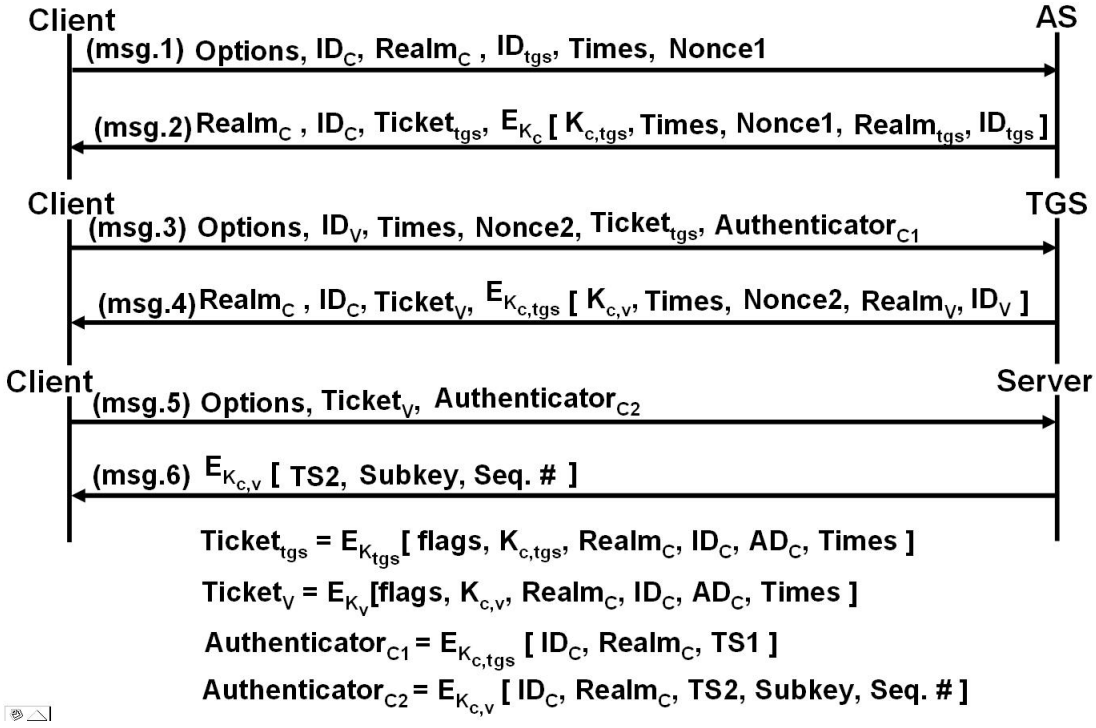

Figure 3: Kerberos 5 authentication dialogue

2) **till:** the requested expiration time for the requested ticket.

3) **rtill:** this field is only present in tickets that have the RENEWABLE flag set in the flags field. It indicates the maximum end-time that may be included in a renewal.

- **Nonce:** it is a random value to be repeated in Message (2) to assure that the response is fresh and has not been replayed by an opponent.

Let us now compare the ticket-granting service exchange for versions 4 and 5. We see that Message (3) in Figure 3 includes requested times and options for the ticket and a nonce, all with functions similar to those of Message (1).

Finally, for the client/server authentication exchange, several new features appear in Version 5. In Message (5), the client may request as an option that mutual authentication is required. The authenticator includes several new fields as follows:

- **Subkey:** The client's choice for an encryption key to be used to protect this specific application session. If this field is omitted, the session key from the ticket ($K_{c,v}$) is used. If the client selects a sub-session key, care must be taken to ensure the randomness of the selected key.

- **Sequence number:** An optional field that specifies the starting sequence number to be used by the server for messages sent to the client during this session (to detect replays).

After that, the server responds with Message (6). This message includes the timestamp from the authenticator. The subkey field, if present, overrides the subkey field of Message (5). The optional sequence number field specifies the starting sequence number to be used by the client.

5 Kerberos Drawbacks

The protocol weaknesses can be stated as follows:

1) Kerberos is vulnerable to password guessing attacks. The Kerberos message includes material encrypted with a key based on the client's password. An opponent can capture this message and attempt to decrypt it by trying various passwords. If the result of a test decryption is of the proper form, then the opponent has discovered the client's password and may subsequently use it to gain authentication credentials from Kerberos. Remember that when a user requests the ticket-granting ticket, the answer is returned encrypted with a key derived by a publicly-known algorithm from the user's password.

2) The system clocks of the hosts that are involved in the protocol should be synchronized. The tickets have a time availability period and if the host clock is not synchronized with the Kerberos server clock, the authentication will fail. In practice, Network Time Protocol daemons are usually used to keep the host clocks synchronized.

3) Kerberos requires continuous availability of the KDC. When the KDC is down, the system will suffer from the single point of failure problem. This can be mitigated by using multiple Kerberos servers.

4) There are no standards to explain the administration of the Kerberos protocol. This will differ between server implementations.

6 Contributions

6.1 Proposed Modifications to Kerberos

It is obvious that Kerberos is vulnerable to password guessing attacks. We introduce simple practical modifications to the KDC database to overcome these attacks. In our modified version of Kerberos, the long-term secret key of the network principle will be independent of the principle's password. Instead, the KDC will save a profile for every instance in the realm that it manages. The type of the profile data contents may be audio, video, image, or simply text data. The KDC database may have mixed types of profiles. The network principle may be a client or a server. Every principle in the network is registered in the KDC database by the principle ID. Then the KDC maps this ID to the proper profile where the profile is named with the principle's ID that belongs to that profile. In order to generate the principle's secret key, we apply a hashing algorithm to the principle's profile and then encrypt the output digest. We control the lifetime of the secret key using the current KDC system time that is appended to the principle's profile every predefined period (this period is a design parameter, i.e. a site constant). By this way, we change the input to the hashing function, and consequently, the output of the hashing function and the secret key will change too. The block diagram of Figure 4 illustrates our proposed scheme.

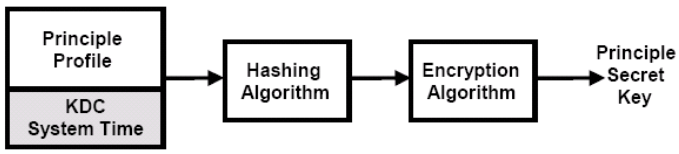

Figure 4: Secret key generation block diagram

6.2 Proposed Implementation

Our authentication dialogue is presented in Figure 5. The elements of each message in the proposed modified Kerberos are explained in Tables 1, 2, and 3.

Table 1: The elements of the proposed modified Kerberos (a)Authentication Service Exchange

Nonce1	TGS returns service-granting ticket
ID_{tgs}	Confirms that this ticket is for the TGS.
$ticket_{tgs}$	Key shared only by the client and the TGS.
K_{tgs}	Copy of session key accessible to client created by TGS to permit secure exchange between the client and the application server.
Flags	The times settings of the returned server ticket (from, till, renew_till).
$K_{c,tgs}$	Repeat for the random value of message 3.
ID_C	Confirms that this ticket is for the application server which identity is V.
AD_C	Reusable so that client does not need to request a new ticket from TGS for each access to the same server.
Times	Ticket is encrypted with key known only to TGS and server, to prevent tampering.

Table 2: The elements of the proposed modified Kerberos (b) Ticket-Granting Service Exchange

Message (3)	Client requests service-granting ticket
Options	Requests that certain flags be set in the returned application server ticket.
ID_V	Tells TGS the application server identity that the client requests access to.
Times	Requests certain time settings in the returned server ticket (from, till, renew_till).
Nonce2	Random value to be repeated in Message (4) to avoid replay attack.
$ticket_{tgs}$	Assures TGS that this user has been authenticated by AS.
$Authenticator_{C1}$	Generated by client to validate ticket. It assures TGS that the ticket presenter is the same as the client for whom the ticket was issued; has very short lifetime to prevent replay.
$K_{c,tgs}$	Authenticator is encrypted with key known only to client and TGS, to prevent tampering.
ID_C	Must match ID in the TGS ticket to authenticate ticket.
AD_C	Must match address in the TGS ticket to authenticate ticket.
TS1	Informs TGS of time this authenticator was generated to check the authenticator's validity.
Message (4)	TGS returns service-granting ticket
ID_C	Confirms that this ticket is for the TGS.
$K_{c,tgs}$	Key shared only by the client and the TGS.
$K_{c,v}$	Copy of session key accessible to client created by TGS to permit secure exchange between between the client and the application server.
Times	The times settings of the returned server ticket (from, till, renew_till).
Nonce2	Repeat for the random value of message 3.
ID_V	Confirms that this ticket is for the application server which identity is V.
$Ticket_V$	Reusable so that client does not need to request a new ticket from TGS for each
	access to the same server.
K_V	Ticket is encrypted with key known only to TGS and server, to prevent tampering.
Flags	The flags of the returned server ticket.
$K_{c,v}$	Copy of session key accessible to client; used to decrypt authenticator, thereby authenticating ticket.
ID_C	Indicates the rightful owner of this ticket.
AD_C	Prevents use of ticket from workstation other than one that initially requested the ticket.
Times	The times settings of the server ticket (from, till, renew_till).

Table 3: The elements of the proposed modified Kerberos (c) Client/Server Authentication Exchange

Message (5)	Client requests service
Options	Requests mutual authentication from the server.
TicketV	Assures server that this user has been authenticated by TGS.
$Authenticator_{C2}$	Generated by client to validate ticket. It assures server that the ticket presenter is the same as the client for whom the ticket was issued; has very short lifetime to prevent replay.
$K_{c,v}$	Authenticator is encrypted with key known only to the client and the application server.
ID_C	Must match ID in the server ticket to authenticate ticket.
AD_C	Must match address in the server ticket to authenticate ticket.
TS2	Informs server of time this authenticator was generated and used to check the authenticator's validity.
Subkey	The client's choice for an encryption key to be used to protect this specific application session. If this field is omitted, the session key from the ticket $K_{c,v}$ is used.
Seq. ♯	An optional field that specifies the starting sequence number to be used by the application server for messages sent to the client during this session to detect replays.
Message (6)	Optional authentication of application server to client
$K_{c,v}$	Assures the client that this message is from the application server which identity is V.
TS2	Assures the client that this is not a replay of an old reply.
Subkey	The server's choice for an encryption key to be used to protect this specific application session. If this subkey present, it overrides the subkey field of Message (5).
Seq. ♯	An optional field that specifies the starting sequence number to be used by the client for messages sent to the application server during this session to detect replays.

Table 4: Comparison between Kerberos 4, Kerberos 5, and our proposed implementation

Comparison Item	Kerberos 4	Kerberos 5	Our Proposed Implementation
Password attack	Vulnerable	Vulnerable	Keys are independent of password
Times	No times	From, till, renew_till	From, till, renew_till
Encyption technique	DES	Encryption key is tagged with type & length	Triple-DES
Double encryption in message 2 & 4	Found	Not found	Not found
DES mode of operation	PCBC (not standard)	The standard CBC mode	The standard CBC mode
Session key	1/lifetime for sub-session key (1/connection)	Client & server may negotiate for sub-session key (1/connection)	Client & server may negotiate for sub-session key (1/connection)
Network address	IPv4	Any (network address is tagged with type)	IPv4
Ticket lifetime	1280 minutes maximum	Arbitrary (determined by start & end times)	Arbitrary (determined by start & end times)

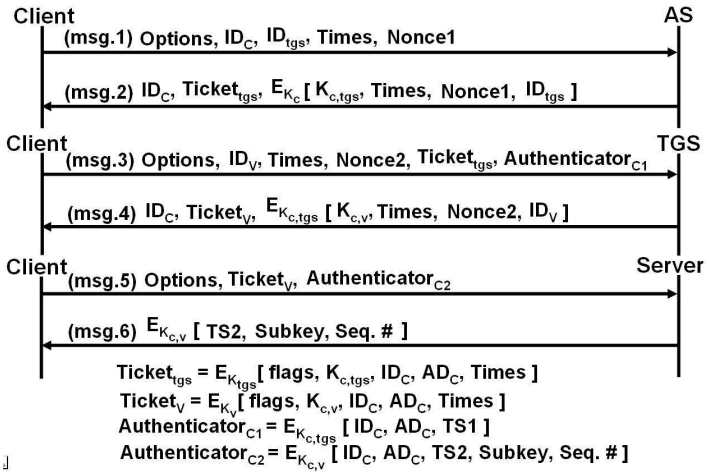

Figure 5: Authentication dialogue for the proposed modified Kerberos

6.3 Comparison Between our Implementation And Previous Versions

Our implementation complies with Kerberos 5 in almost all parts except that the principle's long-term secret key is independent of its password. A comparison between Kerberos 4, Kerberos 5 (the publicly released version of Kerberos) and our proposed implementation is depicted in Table 4:

- Both Kerberos 4 and Kerberos 5 are vulnerable to password guessing attack but our implementation overcomes this problem and that is the main contribution of our work.

- In both Kerberos 5 and our implementation, the client requests certain time settings from the server by the message element "Times" which is subdivided into three subelements (from, till, and renew_till). The "Times" message element is absent in Kerberos 4.

- Version 4 requires the use of DES. In Version 5, ciphertext is tagged with an encryption type identifier so that any encryption technique may be used. We used the DES encryption technique in our implementation.

- Double encryption: Note in Figure 2 that tickets provided to clients in Messages (2) and (4) are encrypted twice, once with the secret key of the target server and then again with a secret key known to the client. The second encryption is not necessary and is computationally wasteful. It is avoided in Version 5 (Figure 3) and in our implementation (Figure 5).

- Besides, encryption in Version 4 makes use of a non-standard mode of DES known as Propagating Cipher Block Chaining (PCBC) ([20] describes this mode of operation). Security problems have been demonstrated in that mode [14]. Version 5 makes use of the standard CBC mode for encryption and our implementation used that mode too.

- Each ticket includes a session key that is used by the client to encrypt the authenticator sent to the service associated with that ticket. In addition, the session key may subsequently be used by the client and the server to protect messages passed during that session. However, because the same ticket may be used repeatedly to gain service from a particular server, there is the risk that an opponent will replay messages from an old session to the client or the server. In both Version 5 and our implementation, it is possible for a client and server to negotiate a sub-session key to be used only for that one connection. A new access by the client would result in the use of a new sub-session key.

- Version 4 requires the use of Internet Protocol (IPv4) addresses. In Version 5, network address is tagged with type and length. This allows any network address type to be used. Our implementation makes use of the IPv4 network address.

- The ticket lifetime in Version 4 is encoded in an 8-bits quantity in units of five minutes. Thus, the maximum lifetime that can be expressed is $256 \times 5 = 1280$ minutes. In Version 5 and in our implementation, the ticket includes an explicit start and end times (the "from" and the "till" of the message element "Times"), allowing tickets with arbitrary lifetimes.

6.4 Security Properties of our Implementation

The security properties of the proposed implementation can be stated as follows:

- The realm principles long-term secret keys are independent of the password, thus the proposed implementation will be susceptible to the password guessing attack.

- **Session key secrecy:** For any client and any server, if the TGS generates a symmetric session key $K_{c,v}$ for a certain client and certain server, then the intruder does not learn that session key.

- **Confidentiality of $K_{c,tgs}$:** If the intruder does not know the long term secret keys (K_C and K_{tgs}) used to encrypt the session key $K_{c,tgs}$ generated by the authentication server AS for use by the client and the TGS, then the intruder cannot learn $K_{c,tgs}$.

- **Authentication of ticket-granting ticket (TGS ticket) and the authenticator ($Authenticator_{C1}$):** If the intruder does not know the long term key used to encrypt a ticket-granting ticket, then if the TGS processes a request,

ostensibly from a client, containing the ticket-granting ticket and the session key $K_{c,tgs}$, then some Authentication Server created the session key $K_{c,tgs}$ for the client to use with the TGS and also generated this ticket-granting ticket. Furthermore, if the intruder does not know the long term key that the authentication server used to send $K_{c,tgs}$ to the client, then the authenticator $Authenticator_{C1}$ was created by the client.

- **Confidentiality of $K_{c,v}$:** If the intruder knows neither the long term secret key used by a TGS to encrypt the service ticket containing a new session key $K_{c,v}$ for a client to use with a server nor the session key used by the client to request the service ticket, then the intruder cannot learn $K_{c,v}$.

- **Authentication of the server ticket and the authenticator ($Authenticator_{C2}$):** If the intruder does not know the long term key used to encrypt a service ticket for the client to present to an application server, then if the server processes a request, ostensibly from the client, containing this service ticket and the session key $K_{c,v}$, then some Ticket Granting Server generated the session key $K_{c,v}$ for the client to use with the application server and also created the service ticket. Furthermore, if the intruder never learns the session key which the Ticket Granting Server used to encrypt $K_{c,v}$ when sending the service ticket to the client, then the client created the authenticator $Authenticator_{C2}$.

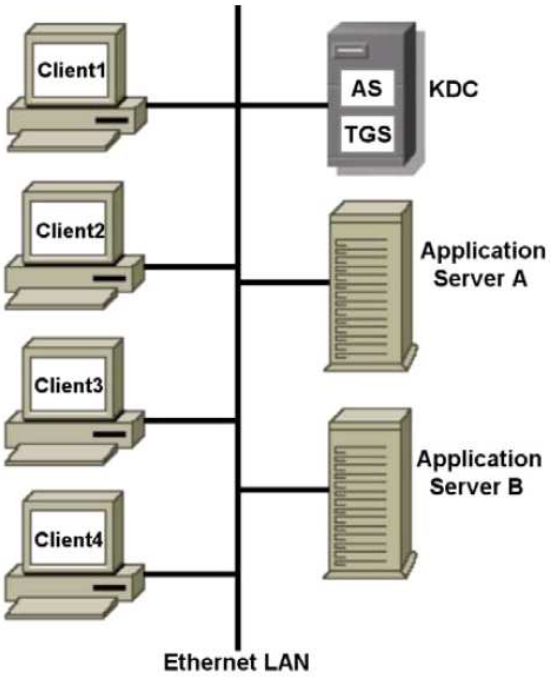

Figure 6: A schematic for the testing LAN

6.5 Testing Environment

Our testing LAN is presented in Figure 6 where the KDC is logically divided into the AS and the TGS. We have two application servers (Server A and Server B) and four clients (client1, client2, client3, and client4). We used a mixed profiles KDC database with 4 profiles corresponding to the four clients (1 audio + 1 video + 1 image + 1 text profiles).

In our implementation, we use Triple-DES in CBC mode as an encryption algorithm, SHA-256 as a hashing algorithm, and Blum Blum Shub as a random number generator algorithm. In our design, the lifetime of the long-term principle's secret key is 1 week, the lifetime of the TGS ticket is 1 day, the lifetime of the application server ticket is 8 hours, and the lifetime of the authenticator is 5 minutes.

We used different scenarios to test our implementation. We will examine one of them here. Client 1 (ID: ccc11) and Client 2 (ID: ccc22) are trying to be authenticated to Server A (ID: vvv11, IP: 10.0.0.2), while Client 3 (ID: ccc33) and Client 4 (ID: ccc44) are trying to be authenticated to server B (ID: vvv22, IP: 10.0.0.3). Figures 7, 8, 9, and 10 shows the clients edited values during the testing scenario. Now we will examine Figure 7. It has four main sections. They could be explained as follows:

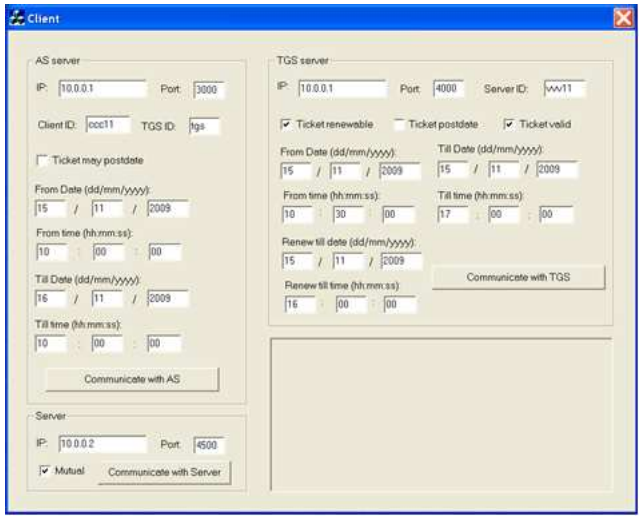

Figure 7: A screen shot for client 1 edited values

1) The first section in the upper left corner of the figure corresponds to the first communication phase during which an exchange between the client and the AS is happened. In that section the client enters some inputs. They are as follows:

   - The IP address and the communication port of the AS.
   - The client's ID.
   - The TGS ID.

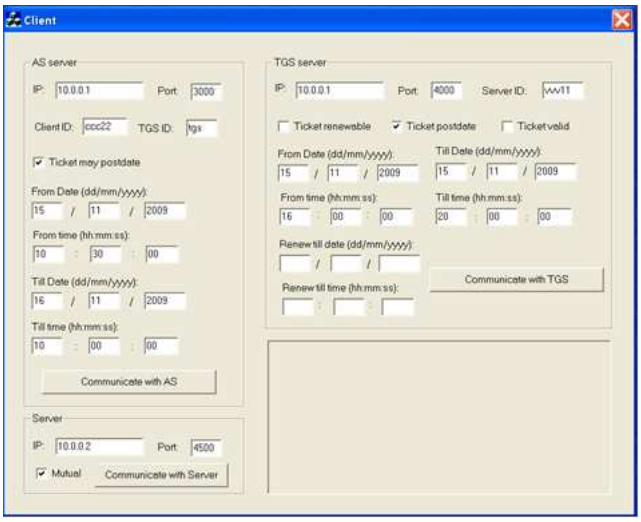

Figure 8: A screen shot for client 2 edited values

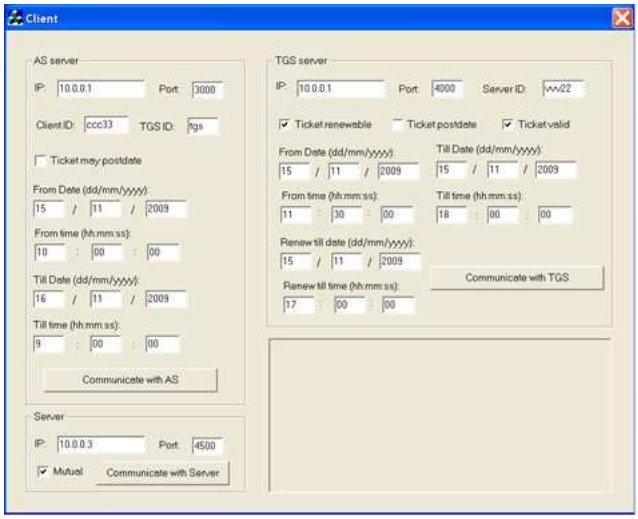

Figure 9: A screen shot for client 3 edited values

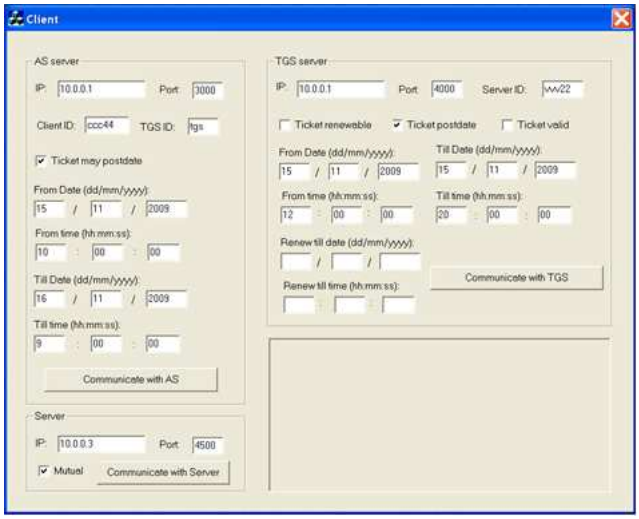

Figure 10: A screen shot for client 4 edited values

- If the client requests that a ticket granting ticket may postdated, the ticket may postdate option should be chosen. It is not chosen in our example.
- The requested times parameters for the ticket granting ticket. They are from date, from time, till date, and till time.

2) The second section in the upper right corner of the figure corresponds to the second communication phase during which an exchange between the client and the TGS is happened. In that section the client enters some inputs. They are as follows:

- The IP address and the communication port of the TGS.
- The application server ID to which the ticket is requested.
- Some options in the requested application server ticket. Here we request a valid renewable application server ticket. Note that we can not chose the "Ticket postdate" option since the "Ticket may postdate" option is not selected in the AS exchange during the first communication stage.
- The requested times parameters for the application server ticket. They are from date, from time, till date, till time, renew_till date, and renew_till time. Note that if the "Ticket renewable" option is not selected, the renew_till date and the renew_till time will be absent in the times parameters.

3) The third section in the lower left corner of the figure corresponds to the third communication phase during which an exchange between the client and the application server is happened. In that section the client enters some inputs. They are as follows:

- The IP address and the communication port of the application server.
- A mutual option is chosen to indicate that mutual authentication is requested from the server.

4) The fourth section in the lower right corner of the figure corresponds to the error message section. If an error occurred during any communication phase, an error is displayed in that section.

Notice that in Figure 8 when the renewable option is not chosen by the client the renew_till date and the renew_till time have no effect and are not edited by the client. Even if they are edited by the client, they will not be considered by our implementation. Besides, in Figure 8 the postdate option is chosen by the client in the TGS exchange and that is allowed since the may-postdate option is requested in the AS exchange.

7 Conclusions

The introduced modifications to the KDC database will enhance the performance of the protocol since the principle's long-term secret-key will be independent of the user password. Thus, our modified Kerberos version is no longer vulnerable to password guessing attacks. We tested our implementation on a small LAN and we are looking forward to extend our implementation to cover cross-realm operations.

References

[1] G. Bella, and E. Riccobene, "Formal analysis of the Kerberos authentication system," *Journal of Universal Computer Science*, vol. 3, no. 12, pp. 1337-1381, 1997.

[2] G. Bella, and L. Paulson, "Kerberos version IV: Inductive analysis of the secrecy goals," *ESORICS '98, Springer-Verlag*, 1998.

[3] S. Bellovin, and M. Merrit, "Limitations of the Kerberos authentication system," *SIGCOMM Computer Communication Review*, vol. 20, no. 5, pp. 119-132, 1990.

[4] A. Boldyreva, and V. Kumar, "Provable-security analysis of authenticated encryption in Kerberos," *IEEE Symposium on Security and Privacy (SP'07)*, pp. 1-21, May 2007.

[5] B. Bryant, *Designing An Authentication System: A Dialogue in Four Scenes*, Project Athena document, Feb. 1988. (http://web.mit.edu/Kerberos/ dialogue.html)

[6] F. Butler, I. Cervesato, A. D. Jaggard, and A. Scedrov, "A formal analysis of some properties of Kerberos 5 using MSR," *IEEE CSFW '02*, pp. 1-16, 2002.

[7] M. Erdem, "High-speed ECC based Kerberos authentication protocol for wireless applications," *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 3, pp. 1440-1444, Dec. 2003.

[8] Y. C. Hu, A. Perrig, and D. B. Johnson, "SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks," *Proceeding of IEEE Workshop on Mobile Computing Systems and Applications*, 2003.

[9] MIT Kerberos Consortium, "The Kerberos webpage," (http://www.Kerberos.org/index.html)

[10] J. Kohl, "The use of encryption in Kerberos for network authentication," *Crypto' 89*, pp. 35-43, Springer-Verlag, 1989.

[11] J. Kohl, and C. Neuman, *The Kerberos Network Authentication Service (V5)*, Network Working Group, RFC 1510, Sep. 1993. (http://www.ietf.org/rfc/rfc1510.txt)

[12] R. Needham and M. Schroeder, "Using encryption for authentication in large networks of computers," *Communications of the ACM*, pp. 993-999, Dec. 1978.

[13] C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," *IEEE Communications Magazine*, pp. 33-38, Sep. 1994.

[14] C. Neuman, T. Yu, S. Hartman, and K. Raeburn, *The Kerberos Network Authentication Service (V5)*, Network Working Group, RFC 4120, 2005. (http://www.ietf.org/rfc/rfc4120.txt)

[15] Nitin et al., "Security analysis and implementation of JUIT-image based authentication system using Kerberos protocol," *Proceedings of the 7th IEEE/ACIS International Conference on Computer and Information Science*, pp. 575-580, 2008.

[16] A. Pirzada, and C. McDonald, "Kerberos assisted authentication in mobile ad-hoc networks," *The* 27^{th} *Australasian Computer Science Conference*, 2004.

[17] K. Raeburn. *Advanced Encryption Standard (AES) Encryption for Kerberos 5*, Network Working Group, RFC 3962, 2005. (http://www.ietf.org/rfc/rfc3962.txt).

[18] K. Raeburn, *Encryption and Checksum Specifications for Kerberos 5*, Network Working Group, RFC 3961. (http://www.ietf.org/rfc/rfc3961.txt, 2005.)

[19] S. Sakane et al., "Applying Kerberos to the communication environment for information appliances," *IEEE Symposium on Applications and the Internet Workshops (SAINT-w'03)*, 2003.

[20] W. Stallings, *Cryptography and Network Security Principles and Practices, 4th edition,* Pearson Prentice Hall, pp. 401-419, pp. 433-435, 2006.

[21] S. Stubblebine, and V. Gligor. "On message integrity in cryptographic protocols," *IEEE Symposium on Security and Privacy' 92*, pp. 85, 1992.

[22] Wikipedia, "Kerberos (protocol)," (http://en.wikipedia.org/wiki/Kerberos_(protocol))

[23] T. D. Wu, "A real-world analysis of Kerberos password security," *NDSS '99*, The Internet Society, 1999.

[24] T. Yu et al., "The perils of unauthenticated encryption: Kerberos Version 4," *NDSS '04, The Internet Society*, 2004.

Eman M. El-Emam received the BSc in Electronics & Communication engineering from Ain Shams University, Cairo, Egypt in 2000. She received a Diploma in Computer Networking from the ITI (Information Technology Institute), Giza, Egypt in 2001. She has been a communication engineer in the ESP (Egyptian Space Program) at NARSS (National Authority for Remote Sensing and Space Sciences), Cairo, Egypt since 2001 till now. Her research interest includes channel coding, network protocols, and network security.

Magdy A. Koutb received the Eng. Degree from the University of Menofiya, Egypt, in 1977 and M.Sc.degree in 1981 from the university of Al-Azhar, Egypt, and the Ph.D. degree from the University of Silesia, Poland in 1985. He has been a Professor since 1997 at the Faculty of Electronic Eng., Menoufiya University. In 2003 he was appointed as Vice-Dean of Post-Graduate Studies and

Cultural affairs of the same faculty. He has extensive experience in Computer Engineering and Industrial Electronics. His main research interests include distributed systems, network security and intelligent control systems.

Hamdy M. Kelash received the Eng. Degree from the Institute of Electronic Engineering, Egypt in 1971, MSc degree from Faculty of Engineering Technology, Helwan University, Egypt, in 1979 and the PHD degree from Institute National Polytechnique (INP), France in 1984. He has been lecturer in 1984 at the Electronic Industry department, Faculty of Electronic Engineering, also a lecturer in 1987 at the Computer Sciences and Engineering department, and an Assistant Professor in 1993 and the Head of Computer Sciences and Engineering department, Faculty of Electronic Engineering, Menoufia University from 2001 to 2007. His main research interests include optical computing, artificial intelligence, network security, image processing, digital systems and parallel computing.

Osama S. Farag Allah received his BS in 1997, MSc in 2002, and PhD in 2007, all in computer science and engineering, from Menoufia University, Faculty of Electronic Engineering, Egypt. He was a demonstrator at the Department of Computer Science and Engineering, at Menoufia University, from 1997 to 2002, became an assistant lecturer in 2002, and was promoted to a lecturer in 2007. His research interests cover computer networks, network security, cryptography, Internet security, multimedia security, image encryption, watermarking, steganography, data hiding, and chaos theory.