*International Journal of Engineering, Management & Technology (IJEMT)*
*www.ijemt.com, Volume 1 Issue X, December 2022, PP 47-55, ISSN (Online): 2583 – 4517*

# The Kerberos Authentication Service

Sachin Agrawal [*1], Dr. Rishi Kumar Agarwal [2]

[1](*Department of Computer Application, DPBS P.G. College Anoopshahr-203390, India)*

[2](*Department of Mathematics, DPBS P.G. College Anoopshar-203390, India)*

**ABSTRACT:** *The Kerberos Authentication Service, developed at MIT, has been widely adopted by other organizations to identify clients of network services across an insecure network and to protect the privacy and integrity of communication with those services. While Version 4 was a step up from traditional security in networked systems, extensions were needed to allow its wider application in environments with different characteristics than that at MIT. This paper discusses some of the limitations of Version 4 of Kerberos and presents the solutions provided by Version 5.*

**KEYWORDS:** *Authentication, Cryptosystem, Encryption, Kerberos, Session key.*

## 1. Introduction

The Kerberos Authentication Service was developed by the Massachusetts Institute of Technology (MIT) to protect the emerging network services provided by Project Athena. 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. Models for administration and use of computer services differ from site to site and some environments require support that isn't present in Version 4. Version 5 of the Kerberos protocol incorporates new features suggested by experience with Version 4, making it useful in more situations. Version 5 was based in part upon input from many contributors familiar with Version 4.

### Terminology and conventions

A principal is the basic entity that participates in authentication. In most cases a principal represents a user or an instantiation of a network service on a particular host. Each principal is uniquely named by its principal identifier.

Encryption is the process of transforming data into a form that cannot be understood without applying a second transformation. The transformation is affected by an encryption key in such a manner that the second transformation can only be applied by someone in possession of the corresponding *decryption key.*

A secret-key cryptosystem such as that defined by the Data Encryption Standard (DES) [FIPS46] uses a single key for both encryption and decryption. Such an encryption key is called a *secret* key.

* **Corresponding Author: Sachin Agrawal**
**Manuscript published on www.ijemt.com : December 29, 2022**

A public-key cryptosystem such as RSA [Riv78] uses different keys for encryption and decryption. One of the keys in the pair can be publicly known while the other must be kept private. These keys are referred to as *public* and *private* keys respectively.

Plaintext is a message in its unencrypted form, either before the encryption transformation has been applied, or after the corresponding decryption transformation is complete. Cipher text is the encrypted form of a message, the output of the encryption transformation.

In figures, encryption is denoted by showing the plaintext surrounded by curly braces ({}) followed by a key (K) whose subscript denotes the principals who possess or have access to that key. Thus, "abc" encrypted under c's key is represented as {abc}Kc.

## 2. The *Kerberos* Model

Kerberos was developed to enable network applications to securely identify their peers. 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 Tc,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 Ac,s) which proves that the client is in posses-sion of the temporary encryption key that was assigned to the principal identified by the ticket. 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 [Nee78], is trusted to hold in confidence secret keys known by each client and server on the network (the secret keys are established out-of-band or through an encrypted channel). 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.

### 2.1. The initial ticket exchange

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 Kc,s, called the *session key,* and gen-erates the requested ticket. The ticket identifies the client, specifies the session key Kc,s, lists the start and expiration times, and is encrypted in the key Ks 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 Kc (in Version 4 all fields are encrypted in Kc).

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.

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 Kc,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 Kc,s. To verify the identity of the client, the sever decrypts the authenticator (using the session key Kc,s from the ticket) and verifies that the timestamp is current.

Successful verification of the authenticator proves that the client possesses the session key Kc,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 Kc, 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.

## 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.

### Environmental shortcomings

Kerberos Version 4 was targeted primarily for Project Athena [Cha90], and as such in some areas it makes assumptions and takes approaches that are not appropriate universally:

**Encryption system dependence:** The Version 4 protocol uses only the Data Encryption Standard (DES) to encrypt messages. The export of DES from the USA is restricted by the U.S. Government, making truly widespread use of Version 4 difficult.

**Internet protocol dependence:** Version 4 requires the use of Internet Protocol (IP) addresses, which makes it unsuitable for some environments.

**Message byte ordering:** Version 4 uses a "receiver makes right" philosophy for encoding multi-byte values in network messages, where the sending host encodes the value in its own natural byte order and the receiver must convert this byte order to its own native order. While this makes communica-tion between two hosts with the same byte order simple, it does not follow established

conventions and will preclude interoperability of a machine with an unusual byte order not understood by the receiver.

**Ticket lifetimes:** The valid life of a ticket in Version 4 is encoded by a UNIX timestamp issue date and an 8-bit lifetime quantity in units of five minutes, resulting in a maximum lifetime of 211⁄4 hours. Some environments require longer lifetimes for proper operation (e.g. a long-running simulation which requires valid Kerberos credentials during its entire execution).

**Authentication forwarding:** Version 4 has no provision for allowing credentials issued to a client on one host to be forwarded to some other host and used by another client. Support for this might be useful if an intermediate server needs to access some resource with the rights of the client (e.g. a print server needs access to the file server to retrieve a client's file for printing), or if a user logs into another host on the network and wishes to pursue activities there with the privileges and authentication available on the originating host.

**Principal naming:** In Version 4, principals are named with three components: name, instance, and realm, each of which may be up to 39 characters long. These sizes are too short for some applications and installation environments. In addition, due to implementation-imposed conventions the normal character set allowed for the name portion excludes the period (.), which is used in account names on some systems. These same conventions dictate that the account name match the name portion of the principal identifier , which is unacceptable in situations where Kerberos is being installed in an existing network with non-unique account names.

**Inter-realm authentication:** 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 $O(n2)$ key exchanges to interconnect $n$ realms . Even with only a few cooperating realms, the assignment and management of the inter-realm keys is an expansive task.

**Authenticators and replay detection:** Kerberos Version 4 uses an encrypted timestamp to verify the freshness of messages and prevent an intruder from staging a successful replay attack. If an authenticator (which contains the timestamp) is out of date or is being replayed, the application server rejects the authentication. However, maintaining a list of unexpired authenticators which have already been presented to a service can be hard to implement properly (and indeed is not implemented in the Version 4 implementation distributed by(MIT).

**Password attacks:** The initial exchange with the Kerberos server encrypts the response with a client's secret key, which in the case of a user is algorithmically derived from a password. An intruder is able to record an exchange of this sort and, without alerting any system administrators, attempt to discover the user's password by decrypting the response with each password guess. Since the response from the Kerberos server includes verifiable plaintext [Lom89], the intruder can try as many passwords as are available and will know when the proper password has been found (the decrypted response will make sense).

**Session keys:** Each ticket issued by the KDC contains a key specific to that ticket, called a session key, which may be used by the client and server to protect their communications once authentication has been established. However, since many clients use a ticket multiple times during a user's session, it may be possible for an intruder to replay messages from a previous connection to clients or servers which do not properly protect themselves (again, MIT's Version 4 implementation does not fully implement this protection for the KRB_SAFE and KRB_PRIV messages). Additionally, there are situations in which a client wishes to share a session key with several servers. This requires special non-standard application negotiations in Version 4.

**Cryptographic checksum:** The cryptographic checksum (sometimes called a message authentication code or hash or digest function) used in Version 4 is based on the quadratic algorithm described in [Jue85]. The MIT implementation does not perform this function as described; the suitability of the modified version as a cryptographic checksum function is unknown.

## 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 [Koh92]. Version 5 addresses the concerns described above and provides additional functionality.

### Changes between Versions 4 and 5

### Use of encryption

To improve modularity and ease export-regulation considerations for Version 5, the use of encryption has been separated into distinct software modules which can be replaced or removed by the programmer as needed. When encryption is used in a protocol message, the cipher text is tagged with a type identifier so that the recipient can identify the appropriate decryption algorithm necessary to interpret the message.

Encryption keys are also tagged with a type and length when they appear in messages. Since it is conceivable to use the same key type in multiple encryption systems (e.g. different variations on DES encryption), the key type may not map one-to-one to the encryption type.

Each encryption algorithm is responsible for providing sufficient integrity protection for the plaintext so that the receiver can verify that the cipher text was not altered in transit. If the algorithm does not have such properties, it can be augmented by including a checksum in the plaintext before encryption. By doing this, we can discard the PCBC DES mode, and use the standard CBC mode with an embedded checksum over the plaintext. It is important to consider the effects of chosen plaintext attacks when analyzing the message integrity properties of candidate encryption algorithms. Some potential weaknesses were found with encryption and checksum methods in initial drafts of the Version 5 protocol [Stu92]. These weaknesses were corrected in subsequent revisions.

**Network addresses**

When network addresses appear in protocol messages, they are similarly tagged with a type and length field so the recipient can interpret them properly. If a host supports multiple network protocols or has multiple addresses of a single type, all types and all addresses can be provided in a ticket.

**Message encoding**

Network messages in Version 5 are described using the Abstract Syntax Notation One (ASN.1) syntax [ISO8824] and encoded according to the basic encoding rules [ISO8825]. This avoids the problem of independently specifying the encoding for multi-byte quantities as was done in Version 4. It makes the protocol description look quite different from Version 4, but it is primarily the presentation of the message fields that changes; the essence of the Kerberos Version 4 protocol remains.

**Ticket changes**

The Kerberos Version 5 ticket has an expanded format to accommodate the required changes from the Version 4 ticket. It is split into two parts, one encrypted and the other plaintext. The server's name in the ticket is plaintext since a server with multiple identities, e.g. an inter-realm TGS, may need the name to select a key with which to decrypt the the remainder of the ticket (the name of the server is bookkeeping information only and its protection is not necessary for secure authentication). Everything else remains encrypted. The ticket lifetime is encoded as a starting time and an expiration time (rather than a specific lifetime field), affording nearly limitless ticket lifetimes. The new ticket also contains a new flags field and other new fields used to enable the new features described later.

**Naming principals**

Principal identifiers are multi-component names in Kerberos Version 5. The identifier is encoded in two parts, the realm and the remainder of the name. The realm is separate to facilitate easy implementation of realm-traversal routines and realm-sensitive access checks. The remainder of the name is a sequence of however many components are needed to name the principal. The realm and each component of the remainder are encoded as separate ASN.1 General Strings, so there are few practical restrictions on the characters available for principal names.

**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. This allows servers such as a password changing server to require that a client present a ticket obtained by direct use of the client's secret key Kc instead of one obtained using a TGT. Such a requirement

prevents an attacker from walking up to an unattended but logged in workstation and changing another user's password.

Tickets may be issued as renewable tickets with two expiration times, one for a time in the near future, and one later. The ticket expires as usual at the earlier time, but if it is presented to the KDC in a renewal request before this earlier expiration time, a replacement ticket is returned which is valid for an additional period of time. The KDC will not renew a ticket beyond the second expiration indicated in the ticket. This mechanism has the advantage that although the credentials can be used for long periods of time, the KDC may refuse to renew tickets which are reported as stolen and thereby thwart their continued use.

A similar mechanism is available to assist authentication during batch processing. A ticket issued as post-dated and invalid will not be valid until its post-dated starting time

passes and it is replaced with a validated ticket. The client validates the ticket by presenting it to the KDC as described above for renewable tickets.

Authentication forwarding can be implemented by contacting the KDC with the additional ticket exchange and requesting a ticket valid for a different set of addresses than the TGT used in the request. The KDC will not issue such tickets unless the presented TGT has a flag set indicating that this is a permissible use of the ticket. When the entity on the remote host is granted only limited rights to use the authentication, the for-warded credentials are referred to as a *proxy* (after the proxy used in legal and financial affairs). Proxies are handled similarly to forwarded tickets, except that new proxy tickets will not be issued for a ticket-granting service; they will only be issued for application server tickets.

In certain situations, an application server (such as an X Window System server) will not have reliable, protected access to an encryption key necessary for normal participation as a server in the authentication exchanges. In such cases, if the server has access to a user's ticket-granting ticket and associated session key (which in the case of single-user workstations may well be the case), it can send this ticket-granting ticket to the client, who presents it and the user's own ticket-granting ticket to the KDC. The KDC then issues a ticket encrypted in the session key from the server's ticket-granting ticket; the application server has the proper key to decrypt and process this ticket. The details of such an exchange are presented in [Dav90].

**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 authori-zation and accounting information as part of a ticket. This information takes the form of restrictions on the use of a ticket. The encoding of each restriction is not a concern of the Kerberos protocol, but is instead defined by the authorization or accounting mechanism in use. Restrictions are carried in the *authorization data* field of the ticket.

When a ticket is requested, restrictions are sent to the KDC where they are inserted into the ticket, encrypted, and thus protected from tampering. In the protocol's most general form, a client may request that the KDC include or add such data to a new ticket. The KDC does not remove any authorization data from a ticket; the TGS always copies it from the TGT into the new ticket, and then adds any requested additional authorization data. Upon decryption of a ticket, the authorization data is available to the application server. While Kerberos makes no interpretation of the data, the application server is expected to use the authorization data to appropriately restrict the client's access to its resources.

Among other uses, the *authorization data* field can be used in a proxy ticket to create a capability. The client requesting the proxy from the KDC specifies any authorization restrictions in the authorization data, then securely transmits the proxy and session key to another party, which uses the ticket to obtain limited service from an application server. Neuman [Neu91] discusses possible uses of the *authorization data* field in detail.

The Open Software Foundation's Distributed Computing Environment uses the *authorization data* field for the generation of privilege attribute certificates (PACs). Privilege information is maintained by a privilege server. When a PAC is requested by a client the privilege server requests a Kerberos ticket identifying the privilege server itself, but restricting the groups to which the client belongs and specifying a DCE specific user ID. The ticket is then returned to the client which uses it to assert its DCE user ID and prove member-ship in the listed groups. In essence, the privilege server grants the client a proxy authorizing the client to act as the privilege server to assert the listed DCE user ID and membership in the listed groups. If the ticket did not include restrictions, it would indicate that the client was the privilege server, allowing the client to assert any user ID and membership in any group.

**Conclusion**

Version 5 of Kerberos is a step toward the design of an authentication system that is widely applicable. We believe the framework is flexible enough to accommodate future requirements. Some items we expect to add to Kerberos in the near future include:

**Public-key cryptosystems:** The encryption specifications in Kerberos Version 5 are designed primarily for secret-key cryptosystems, but we are considering support for public-key cryptosystems. One advantage of such support will be the ability to interoperate with the evolving certificate infrastructure for Privacy Enhanced Mail. There is also work proceeding on the development of a hybrid Internet

Authentication System (IAS) that will provide interoperability between Kerberos and public key

based systems such as Digital Equipment Corporation's DASS [Tar91].

**"Smartcards":** Several companies manufacture hand-held devices which can be used to augment normal password security methods, and there is strong interest within the industry to integrate one or more of these systems with Kerberos. Work is underway to use the pre-authentication data field to pass the additional information needed to use such devices.

In the more distant future it might also be possible to program a smartcard to directly take part in the Kerberos protocol. To do so would require special hardware to support communication between the smartcard and the workstation (so that the smartcard could communicate with the KDC). The advantage of such an approach is that the initial Kerberos exchange could take place without making the user's password available to a potentially untrusted workstation.

**Remote administration:** 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

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

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).

D. Borman, Editor, ''Telnet Encryption Option,'' Internet-Draft, *Internet Engineering Task Force, Telnet Working Group* (July 1991).

D. Borman, Editor, ''Telnet Authentication: Kerberos Version 5,'' Internet-Draft, *Engineering Task Force, Telnet Working Group* (February 1992).

William Stallings. Network Security Essentials Applications and standards, *Pearson Education.*

Behrouz A. Forouzan, Debdeep Mukhopadhyay. Cryptography and Netwok Security, *Tata Mc Graw Hill Education Private Limited.*

Atul Kahate. Cryptography and Netwok Security, *Tata Mc Graw Hill Education Private Limited.*