The use of Encryption in Kerberos for Network Authentication

John T. Kohl
Digital Equipment Corporation
MIT Project Athena
Cambridge, Massachusetts

Abstract

In a workstation environment, the user often has complete control over the workstation. Workstation operating systems therefore cannot be trusted to accurately identify their users. Some other method of authentication is needed, and this motivated the design and implementation of the Kerberos authentication service.

Kerberos is based on the Needham and Schroeder trusted third-party authentication model, using private-key encryption. Each user and network server has a key (like a password) known only to it and the Kerberos database. A database server uses this knowledge to authenticate network entities to one another.

The encryption used to achieve this authentication, the protocols currently in use and the protocols proposed for future use are described.

1 Introduction

This paper gives a brief overview of Kerberos, an authentication system developed at Project Athena at M.I.T., and describes the rationale behind and uses of encryption in Kerberos to achieve its goals. More complete descriptions can be found in [8, 4], and in a forthcoming Request For Comments.

It begins with a quick overview of the message scheme used to achieve authentication, then describes the use of encryption in the current protocols (including its flaws), and finishes by describing modifications proposed for the next version of the Kerberos protocols.

2 Terminology

Throughout this paper, we use certain terms relating to Kerberos which may be unfamiliar to the reader. Below is a definition of such terms.

principal An entity which shares a private key with some Key Distribution Center (KDC), and therefore can participate in authentication exchanges. A principal's name is bound to a private encryption key in the KDC's database. The current implementation allows two-part names for principals, consisting of a name field and an instance field. The realm of a principal is determined by the realm name assigned to the KDC with which it shares a private key.

server A principal which provides a Kerberos-mediated service to other principals.

client A principal which desires to use a service.

G. Brassard (Ed.): Advances in Cryptology - CRYPTO '89, LNCS 435, pp. 35-43, 1990.

realm An autonomous unit of authentication authority. All principals in a realm share a key with that realm's KDC. Realms may share keys with each other to allow authentication between principals in different realms.

session key A randomly-generated encryption key contained in a Ticket or in Credentials.

Ticket A data structure cryptographically sealed under a server's private key. The ticket contains information necessary for a principal to verify another's identity based on the trust of the KDC. Tickets can be re-used until they expire.

Credentials A data structure composed of a Ticket and the information needed by a client to use that Ticket.

Authenticator A data structure cryptographically sealed under a temporary key. The authenticator contains information used to aid in replay detection. An authenticator may only be used once.

3 Kerberos overview

Kerberos provides a means for two principals (for example, a workstation user and a network server) to verify each other's identities in the context of an open (i.e. unprotected) network system. This must be accomplished without relying on authentication by the workstation operating system or host addresses, without requiring physical security of all the hosts on the system, and under the assumption that packets traveling along the network can be read and inserted at will. Kerberos performs authentication under these conditions as a trusted third-party authentication service using private key encryption.

Kerberos is based on protocols described by Needham and Schroeder [5], Voydock and Kent [9], Denning and Saco [1], and Watson [10]. A central Key Distribution Center (KDC) maintains a database of principals and private keys (currently only DES keys are supported). When a principal desires to authenticate with some service, it sends a request to the KDC, which responds with a Ticket and other control information encrypted in the principal's private key. The principal decrypts the reply, stores the contents for possible future use, and then forwards the Ticket plus a freshly-constructed Authenticator to the service. The service can verify the identity of the client by examining the Ticket (which itself is encrypted in the service's private key), and verifying its contents with the information contained in the Authenticator.

The current protocols are known as "Version 4" (there were lower-numbered prototype protocols); the protocol revision underway will yield Kerberos protocol "Version 5".

4 Version 4 Protocol

4.1 Encryption

The basic encryption algorithm used in the current version of Kerberos is the U.S. National Institute of Standards and Technology (NIST)[1] Data Encryption Standard (DES) [6]. DES is a block cipher, operating on 64-bit blocks.

The standard mode of encryption under DES is called Electronic Code Book (ECB). ECB mode is not used by Kerberos because it has deficiencies when applied to successive

[1]Formerly the National Bureau of Standards.

blocks of data. When block-aligned repetitive data are encrypted using ECB, they can be recognized as identical ciphertext blocks (e.g. an array of zeros larger than several blocks will show up as a set of identical ciphertext blocks). While this does not directly reveal the encrypted data, it does put them at greater risk to discovery through cryptographic analysis.

FIPS 81 [7] defines the Cipher Block Chaining (CBC) mode of DES to alleviate this problem. The ciphertext of the previous block is bitwise exclusively or'ed (XOR'ed) with the cleartext before encryption, so that block-aligned data are masked (see Figure 1). For the first block, the encryption key is used as the Initialization Vector (IV), which is treated as the previous ciphertext block and XOR'ed into the first block before encryption. However, CBC does not provide any integrity assurance (which Kerberos desires). If a ciphertext block is modified, the error induced after decryption spans only the block modified and the following block. An integrity check can be added by computing a checksum on the cleartext before encryption, and encrypting it as part of the cleartext.

But the Kerberos designers wanted to do the encryption and integrity check in a single pass. A first pass to compute a checksum followed by a separate pass to perform the encryption was deemed too expensive for performance reasons [3]. Their design criteria specifically did not expect hardware DES encryption assist, and so they rely on software implementations of the encryption algorithms [4]. As as result, they were not constrained to the officially defined standard modes of operation (which they would have been limited to, had they assumed hardware assist). So they devised what they called "Plaintext Cipher Block Chaining" mode (PCBC) in which the cleartext of the previous block as well as the ciphertext of the previous block are XOR'ed into the current block before encryption (see Figure 2). The result of PCBC is that errors in the decrypted cleartext would propagate themselves to all successive blocks of the cleartext. This property allowed the use of PCBC without a separate integrity check (the encrypted messages contained enough predictable contents at the end of the cleartex to make detection of a modified block highly likely).

However, PCBC has a different deficiency: swapping two ciphertext blocks will foul the cleartext of the corresponding blocks (and all blocks between) upon decryption, but due to the nature of the XOR method with the cleartext and ciphertexts, the errors cancel out, and succeeding blocks are properly decrypted. So if the integrity checks look at the last few blocks to verify message integrity (as the current implementation does), the checks can be fooled into accepting a partially garbled message.

4.2 Cryptographic checksums

In addition to the use of encryption to seal and protect messages, a quadratic checksum algorithm is available in an optional application protocol to achieve a lower-cost assurance of integrity (without assurance of privacy). The algorithm is modified from Jueneman [2] (The modifications have not been analyzed with respect to cryptographic security.). The checksum is computed with the session key used as a seed. However, in the current protocol the checksum is not encrypted when transmitted, leaving the session key exposed to possible attack by inversion of the algorithm. If the checksum were encrypted, an attacker would have to discover the session key through cryptanalysis on the seeded checksum.

Figure 1: The Cipher Block Chaining (CBC) mode of DES

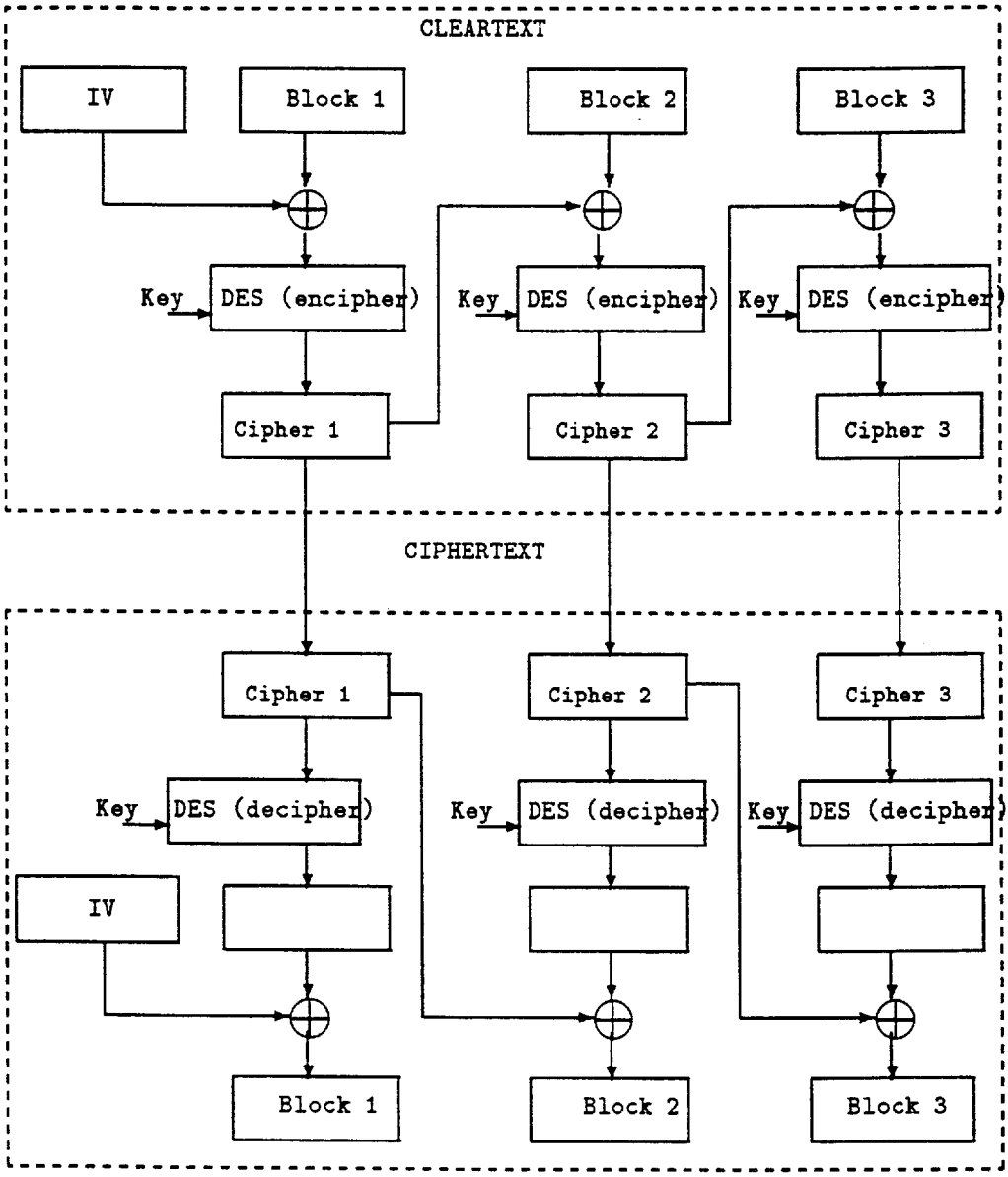
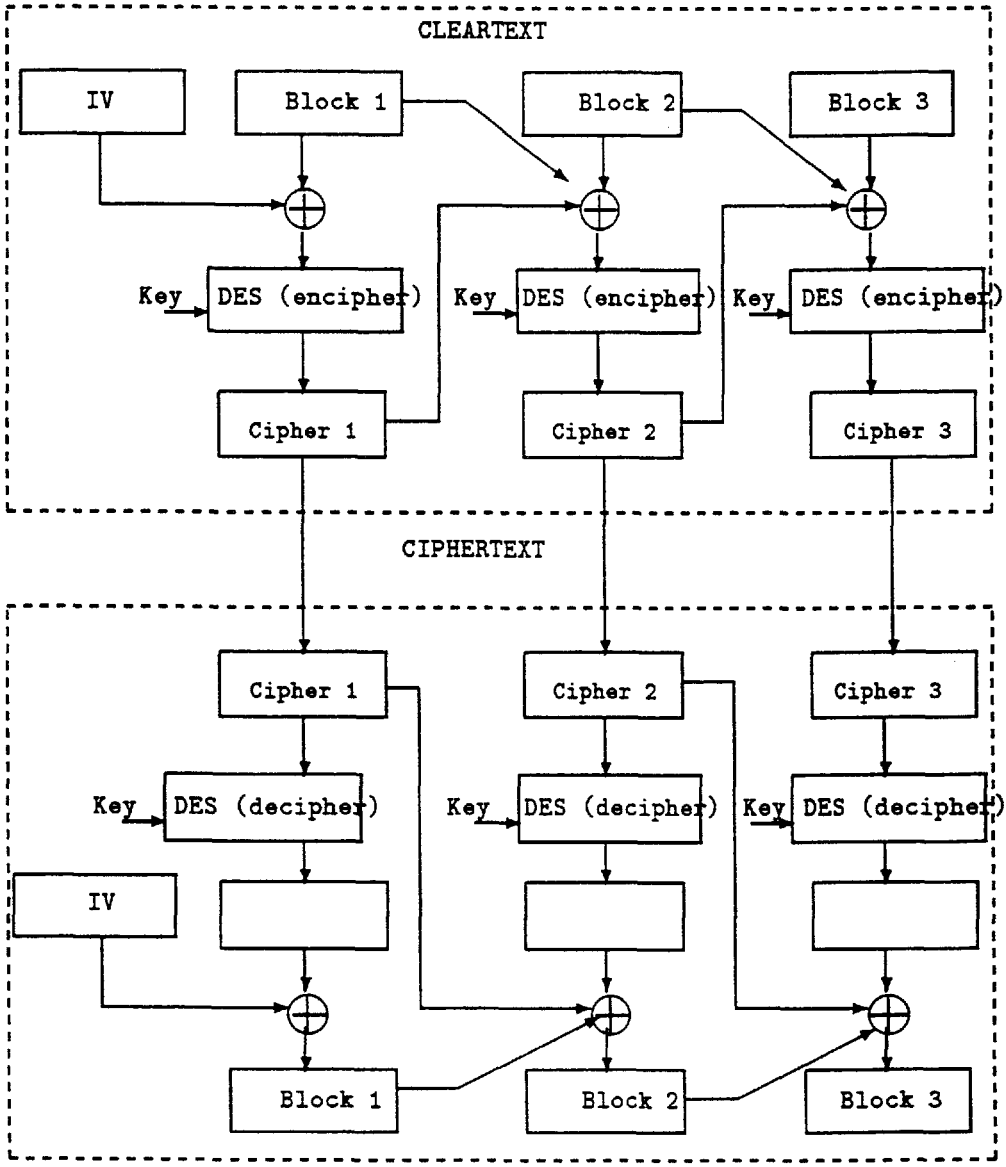


Figure 2: The Plaintext Cipher Block Chaining (PCBC) mode of DES (non-standard)



4.3 Cryptanalysis

The Kerberos protocols and their current implementation were designed with the assumption that the cryptosystem was secure. Very little analysis of cryptographic attacks was performed.

4.4 Application protocols

4.4.1 Authentication Service

In order to obtain a Ticket to present to a service, the client sends a cleartext message to the KDC, containing its name, instance, and realm, the client host's time of day, a requested lifetime for the Ticket, and the service name and instance for the desired Ticket.

The KDC retrieves the records for both the client and server, constructs a Ticket and associated credentials information, encrypts the Ticket under the server's key, encrypts the credentials and encrypted Ticket under the client's key, and returns the encrypted data along with some cleartext control information to the client.

The Ticket contains the client's name, the client's host's network address, a session key, a lifetime, the time at which the Ticket was issued, and the server's name, instance, and realm.

The credentials information contains a copy of the session key contained in the Ticket, the server's name, instance, and realm, the lifetime of the Ticket, the key version number of the server's private key used to create the Ticket, the length of the Ticket, the Ticket itself, and the KDC's time of day.

It should be noted that the Ticket itself need not be encrypted along with the rest of the credentials in the response. The Ticket is usually passed over the network from the client to the server, and since the Ticket is encrypted in a secret key, the session key contained therein is safe from release.

4.4.2 Client to Server

After obtaining a Ticket and associated Credentials, the client constructs an Authenticator (which contains the client's name, host network address, and timestamp, to be used as anti-replay information, all encrypted in the session key) and sends the Ticket, Authenticator, and (possibly) other application-protocol information to the application server.

The server decrypts and verifies the Ticket using its private key. If successfully verified, it uses the enclosed session key to decrypt the Authenticator and verifies the anti-replay information.

This achieves authentication of the client to the server; if the client requires the server to authenticate in return, the server can use the session key to generate a reply proving that it has access to the session key. This serves to authenticate the server to the client, since we assume only the correct server knows the private key and could decrypt the Ticket and obtain the session key.

4.4.3 Ticket-Granting Service

There is a special service provided on the KDC which acts like most Kerberos-mediated services, but has access to the KDC database. This service, dubbed the "Ticket-granting service" (TGS) can issue new Tickets without requiring the client to have its private key

available (which typically would require a client workstation to store or repeatedly request the user's password).

When a user logs in, the workstation software requests a Ticket for this TGS, using the normal Authentication Service protocol. The user types his password (during the login process), and it is used to decrypt the response. The Ticket and associated Credentials are cached on the workstation. The lifetime of this Ticket is usually short (about 8 hours at MIT Project Athena), so that the exposure of leaving the Ticket and session key stored on the workstation and subject to theft and malicious or unauthorized use is limited to a short time span. If the user's password were stored, a thief could impersonate the user for a potentially much longer time (until the user changed his password).

When Tickets are required for additional service, the client workstation uses the standard client-to-server protocol to send its TGS ticket and an Authenticator, along with a client timestamp, requested lifetime, and the name of the service for which tickets are needed to the TGS. The TGS uses its private key (which it can fetch from the database) to decrypt the ticket, verify the Authenticator, and fulfill the request by constructing a new Ticket and associated Credentials. As in the Authentication Service protocol, the Credentials and ticket are encrypted (but in the session key from the TGS ticket, rather than the client's private key) and returned to the client, which then decrypts and caches the Ticket and Credentials.

4.4.4 Integrity-protected messages

The "KRB_SAFE" protocol message is used when a client and server want to verify the integrity of a message without requiring privacy or degrading performance by utilizing encryption.

The message contains user data, some control information, the sender's network address, and the sender's host's time of day, along with a quadratic "cryptographic" checksum (described above) of the entire message seeded with the shared session key. An incorrect checksum (as verified by the receiver) or incorrect control information indicates a modified or unauthentic message.

4.4.5 Privacy-protected messages

The "KRB_PRIV" protocol message is used when a client and server want to verify the integrity and protect the privacy of a message.

The message contains user data, some control information, the sender's network address, and the sender's host's time of day, encrypted (using PCBC mode of DES) under the session key. Upon decryption, a garbled message or incorrect control information indicates a modified or unauthentic message.

5 Planned version 5 changes

Project Athena plans to be able to support different encryption types in the version 5 protocol messages. We expect to implement only a DES-based version. We hope that other implementors will provide different encryption types.

Due to the above discussed problems with PCBC, we have decided to use the CBC mode of DES combined with a data checksum to provide integrity and privacy. The choice of a

checksum algorithm to use is still under discussion; we are seeking an algorithm that won't negatively interact with DES.

We also are seeking a better cryptographic checksum than the quadratic checksum (which doesn't have much analytical proof). The DES CBC checksum mode has better-studied properties, but is computationally much more expensive than the quadratic checksum. We would ideally like a computationally "cheap" checksum which is also reasonably secure.

We expect to fix the "KRB_SAFE" protocol by allowing user selection of a cryptographic checksum algorithm.

6 Conclusion

This paper has discussed the encryption used in the Kerberos protocol and the rationale and design decisions underlying some of the uses of encryption. It has noted deficiencies in the current implementation and protocols, and suggests changes to alleviate those problems in the next version of the protocol.

Kerberos has succeeded in its goal of using software encryption by limiting the amount data required to be encrypted for the base authentication protocols, and allows applications to choose appropriate levels of cryptographic integrity and privacy.

7 Acknowledgments

The author would like to thank Jon Rochlis and Steve Miller for their comments on drafts of this paper.

References

[1] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in Key Distribution Protocols. *Communications of the ACM*, 24(8):533–536, August 1981.

[2] R. R. Jueneman et al. Message Authentication. *IEEE Communications*, 23(9):29–40, September 1985.

[3] Steven P. Miller. Private communication.

[4] Steven P. Miller, B. Clifford Neuman, Jeffrey I. Schiller, and Jerome H. Saltzer. Section E.2.1: Kerberos Authentication and Authorization System. *Project Athena Technical Plan*, December 1987.

[5] Roger M. Needham and M. D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, Dec 78.

[6] National Bureau of Standards. Data Encryption Standard. *Federal Information Processing Standards Publication*, 46, 1977.

[7] National Bureau of Standards. DES Modes of Operation. *Federal Information Processing Standards Publication*, 81, 1980.

[8] Jennifer G. Steiner, B. Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An Authentication Service for Open Network Systems. *Usenix Conference Proceedings*, pages 183–190, February 1988.

[9] Victor L. Voydock and Stephen T. Kent. Security mechanisms in high-level network protocols. *Computing Surveys*, 15(2):135–171, June 1983.

[10] R. W. Watson. Timer-Based Mechanisms in Reliable Transport Protocol Connection Management. *Computer Networks*, 5, 1981.