87-20-15 Private Keys, Trusted Third Parties, and Kerberos

Alex Bidwell

Payoff

Shared private key cryptography and public key cryptography can provide solutions to such issues as protection of information, E-mail privacy, user authentication, integrity of information, information authenticity, and all the issues involved in secure commerce. Authentication protocols such as Kerberos, and such E-mail applications as privacy enhanced mail and Pretty Good Privacy, illustrate that key management is one of the most important issues confronting the successful deployment of these technologies. Trusted third parties are always required. Assessing whether they are secure(can be trusted) and readily accessible is an essential part of the infrastructure needed to provide secure communications and commerce on the Internet. Even though a technology is familiar, its implementation can be flawed. Equally important, cryptography is not magic; security basics still apply.

Introduction

The information superhighway is a reality. Organizations must decide whether to get on, where to go when they do get on, what to do when they get there, and how best to do it. Perhaps knowing what not to do when they get there is just as important. This article provides information to help users stay on the safe part of the highway.

Cryptography Basics and Goals

Basic ciphers have existed for hundreds of years. In times of war, the phrase “knowledge is power” has an obvious meaning. Keeping critical information secret is crucial. The critical issues are secrecy, integrity, and authenticity (i.e., whether the information obtained has been tampered with and who that information is from). The technologies that address these issues provide solutions to authenticating users of the system. Encryption technology has advanced so far that all users on the Internet can download programs to their PC to encrypt their files or E-mail at no cost. However, the federal government will not allow anyone to sell or use these implementations outside the United States, except when the algorithms are sufficiently weak that encrypted messages using them can be broken in a relatively short time.

The basic function of cryptography is to convert information into something that is not understandable by those who do not have the authority or means to read it. Readable information is called plaintext, and encrypted information is called ciphertext. The function of encryption is the transformation of plaintext tociphertext and the function of decryption is to recover the original plaintext from the ciphertext received as shown in Exhibit 1. People and organizations want their private information to remain private, whether it is an E-mail message, a letter, or a file of financial or sensitive data accessed by mistake or by deception.

Encryption and Decryption

If an effective cipher algorithm is used, the plaintext will not be easily recovered from the ciphertext unless the receiver of the message knows how to decrypt it. Users of ciphers have several goals:

- **Privacy/Secrecy.** The sender does not want an eavesdropper to able to decrypt and read the ciphertext sent or stored. The loss of the secret to“the enemy” could be damaging.

- **Integrity.** The receiver of the encrypted information does not want attackers to be able inject false information without detection. The parties using the information do not want to be fooled by false information sent in place of the original message.

- **Authenticity.** The receiver of the information would like to be able to determine from where and from whom the information actually came. Thus, the strength and features of the cipher mechanism are important.

Secure Hash Functions

The values produced by secure hash functions are used to verify the integrity of a message directly. hash values are also used as a surrogate for information that needs to be kept secret (e.g., system passwords). A shadow password file is a file of the Hash values of each user's password, and a system sign-on is verified if theHash value of a user's password compares with that stored on the shadow file for that user's ID. Hash functions can be used with encryption techniques to provide digital signatures, which verify not only the integrity of a message (or document) but also its authenticity. Hash functions H{M} generate a short fixed-length (i.e., typically 128 bits) representation of a message. Secure Hash functions have several properties. They must produce unique output values for each arbitrary input M, such that given any other message

M' M, H{M'} H{M}.

In addition, it must be impossible to reconstruct M. The likelihood of a mathematical collision or reconstruction of M should be nil.

The Hash functions MD2 and MD5, designed by Rivest, and the secureHash algorithm (SHA) of the National Institute of Standards and Technology (NIST) are considered secure enough to be adopted by various standards bodies and used in many implementations. MD2 is the slowest of the three. MD5 was created after some partially successful cryptoanalytic attacks on MD4 were reported. SHA has been modeled after MD4 with improvements including a 160-bit Hash instead of a 128-bit output.

Private Or Secret Key Cryptography

Symmetric cryptosystems use the same private or secret key for both encryption and decryption. As shown in Exhibit 2, this key is shared between parties wishing to keep their communications private.

Encryption and Decryption Using Shared Secret Key

The plaintext message (M) is encrypted into ciphertext © using the algorithm directed by key (K),(K)C K{M};

and plaintext (P) is reproduced when © is decrypted or mapped using an inverse algorithm directed by (K), which is indicated by(K-1). P K-1{C} = K-1{K{M}} = M

An example clarifies which of the user's goals can be satisfied by private symmetric key cryptography and illustrates why these goals are important. Consider the simple substitution cipher. According to some predetermined scheme, one set of characters in the alphabet is used in place of other characters. If the sender uses the 36 alphanumeric

Previous screen

characters formed by capital letters only and numbers, and the key is zero (0), no substitution is made; if the key is+1, B is used in place of A, C for B, 0 for Z, and A for 9.Examine the following ciphertext when the plaintext is encrypted with a key of -2.

Plaintext: MEET ME AT 4PM

Ciphertext: KCCR KC 8R 2NK

How well does this cipher meet the user's goals? At best, this cipher preserves secrecy for a short time only, because the enemy merely needs to examine a few paragraphs to break it. Moreover, this cipher does nothing to indicate the integrity of the message. If receivers were to use a different cipher algorithm or key with all those with whom they communicate, they might have a clue as to who sent the message. Senders could confuse the enemy by changing the cipher algorithm. For example, they could choose an 8-bit key and XOR it with each successive 8 bits of the message. This cipher can be made stronger by generalizing the algorithm and letting the key indicate which part of the algorithm to use and what to do. For example, the substitution code could be changed based on which character is being operated on in a block of plaintext, for example, eight characters. The logical operation, XOR, can also be performed with parts of the key on different blocks of plaintext. Because each unique key makes the algorithm operate differently on the message, changing the key effectively changes the algorithm.

It is much easier to generate new keys than to invent new cipher algorithms; thus, this article addresses the complexity of only that problem. Consider the number of cipher keys required to communicate securely versus the number of parties involved. To communicate with two other people who should not be able to decipher each other's messages, two unique keys are needed. To communicate with three others three unique keys are needed. However, six unique keys will be involved in this case for the four parties, assuming that all parties communicate and each pair of senders and receivers uses a unique key. The total number of keys involved for N parties will be N(N - 1)/2. Thus, if 20 parties need to communicate securely, 190 unique key pairs are involved, which is difficult to manage.

Key Management

How can all of those keys be securely generated and distributed among the parties involved? A lot of trusted couriers will be kept busy, especially if the secret keys are changed often. It is important to limit the time that an enemy has to decrypt messages intended only for the user. It is also necessary to be able to revoke keys whose security has been compromised. The security implications of key management are almost as difficult as those for generating secure ciphers.

Users need the required number of keys to be generated and distributed securely to all the parties involved. The key manager must be able to generate and securely distribute keys to all the parties involved, account for how and when keys were delivered, and certify who received a particular key. Key revocation is equally important. If an encrypted path is compromised, all parties using the keys involved must be notified in a manner they can authenticate.

Attacker's Goal

Decrypting the message requires knowledge of the cipher algorithm and the key. The attacker must discover or guess which cipher is being used and try to recover the key to break the cipher.

Strong Cipher Goal

It is desirable that the cipher algorithms be so strong that even when they are in the public domain, for example, Data Encryption Standard (DES) or International Data

Encryption Algorithm (IDEA), a key is of sufficient length that guessing is very difficult, even with the most powerful computers. No matter how much plaintext orciphertext is captured, it should be impossible to recover, short of an exhaustive search by trying all possible key values. With strong ciphers, the effort (i.e., time) required to crack the code is usually an exponential function of key length.

Cryptoanalysis

Cryptoanalysis is the science of recovering plaintext from ciphertext without prior knowledge of the key. Any of the attack techniques described here, as well as analysis to discover weaknesses in key generation or in the algorithm, are sufficient to accomplish this recovery. The news media reported “weaknesses in Netscape's encryption” in the fall of 1995. At the heart of this report was some basic detective work. From the literature provided by Netscape explaining how it generated random numbers for generating keys, it became clear that a large portion of the key space was never used. This unused space significantly reduced the time of a brute force attack to hours instead of years using only a powerful personal computer. Practical cryptoanalysis refers to stealing the key by any means. When someone discovers the keys to encrypted links, these links are compromised.

Types of Attacks

Attacks are either passive or active. In addition to good detective work, active attacks include two basic modes: an exhaustive search of the key space based on an assumed key size and cipher algorithm to find the key that converts the ciphertext to understandable plaintext, and a more clever approach, such as tricking a person or machine into including some attacker's chosen plaintext with the user's normal plaintext. The chosen plaintext and resulting ciphertext are then included in the attacker's codebook with such information as the date, day of the week, and day of the year for later correlation. If these plaintext-ciphertext pairs show up again, indicating a key value is being reused, the enemy can mount a codebook attack to decipher the values in the message that have previously been included in the codebook.

Passive Attack or Eavesdropping

Eavesdropping can be accomplished by listening to passing traffic to capture plaintext or ciphertext or to learn about the habits of the messaging parties, their systems, and the protocols that they use. Attackers can use a sniffer on a local area network to observe E-mail sent to and from the server.

Active Attack

When mounting an active attack, the enemy inserts, deletes, or modifies legitimate messages with the purpose of deceiving the messaging parties or system administrators, thereby corrupting existing information to degrade performance or spoofing the system to gain unauthorized system entry. Exhibit 3 shows types of attacks and their effects. Attacks include:

- **Brute force attack.** This is an exhaustive search for key value. Advances in computer technology and parallel processing have greatly reduced the time to search for key values that will crack the code. However, single-lengthData Encryption Standard keys (i.e., 56 bits) are still considered to provide acceptable protection for most applications.

Previous screen

- **Known-plaintext attack.** This uses one or more pairs of known plaintext, and the corresponding ciphertext is assumed to be encrypted with the same key. Obtaining and identifying such pairs are the attacker's challenges and are usually accomplished by deception or selective garbage collecting. As more samples are collected and associated with unique but unknown keys, codebook attacks form the basis for plaintext recovery or possible determination of the key.

- **Chosen plaintext attack.** This tricks the enemy into encrypting messages or embedded passages chosen by the attacker. By observing the ciphertext of several messages, the encrypted equivalent of the chosen text may be determinable and added to the attacker's codebook. The chosen plaintext attack is one of the better methods for determining whether a previously used key value is currently being used.

- **Cut-and-paste.** This involves combining two or more messages encrypted with the same key to try to produce a new message. Users may be able to trick the enemy into some action without knowing the exact message content.

- **Time resetting.** In protocols using current time, senders try to confuse the enemy as to the current time, thereby making themselves open to replay or other attacks affecting authenticity or integrity.

- **Replay.** Attackers often insert a legitimate message into the network, at a time later than originally sent, to trick the receiving party into believing the message is authentic. Attackers can request secret codes or passwords to valuable resources, or they can repeat orders for transactions, which could lead to embarrassing situations.

- **Man-in-the-middle.** The enemy sits between the user and the party with whom the user wishes to communicate and impersonates each one to the other. The deception can be accomplished at a store and forward message node by performing unauthorized actions or by spoofing the parties into believing that the addresses or the keys that the enemy is using belong to the parties wishing to communicate.

Types of Attacks and Their Effects

Attack Type	Possible Effects		
	Secrecy	Integrity	Authenticity
Passive (Eavesdropping)	Yes	No	No
Active	Maybe	Yes	Yes
Brute force	Yes	No	No
Known plaintext	Yes	No	No
Chosen plaintext	Yes	Yes	Maybe
Replay	Maybe	Yes	Yes
Cut-and-paste	Maybe	Yes	Yes
Time resetting	Maybe	Yes	Yes
Man-in-the-middle	Maybe	Yes	Yes

Practical Private Or Symmetric Key Algorithms

Several algorithms are used today, and they provide excellent protection for most purposes. If more protection is needed, some techniques use double- or triple-length keys.

Previous screen

Data Encryption Standard (DES)

The Data Encryption Standard is the result of a proposal by scientists at IBM for a sophisticated and implementable encryption algorithm in response to several requests by the National Bureau of Standards (now the National Institute for Standards and Technology) in the early 1970s. One of the key requirements was that the strength of the cipher must reside in the key; the algorithm could be made public and was. The DES algorithm has been evaluated by the National Security Agency and standards bodies and has been certified as a secure cipher through 1998 by the National Institute for Standards and Technology.

DES is a block cipher and operates on 64-bit blocks of data at a time for both encryption and decryption operations using the same key. Hence, input is partitioned into 64-bit blocks of plaintext. Encryption yields 64-bit blocks of ciphertext and vice versa for decryption. The key length is 56 bits. It is usually expressed as a 64-bit number using every least significant eighth bit as a parity bit. The algorithm is an iterated block cipher using 16 key-directed iterations. It is based on key-directed substitutions, permutations, and logical XOR operations with previous parts of the data throughout the 16 iterative levels.

Its strength is based on the enemy's ability to choose which of the possible 256 72*1015 key values were used for this encryption. However, its strength really depends on whether the key generation facility generates truly random keys. If some of the random combinations become predictable, the cracking complexity can be greatly reduced. Export status has recently been given to versions ofDES using a 40-bit key specially generated from 56-bit keys using IBM's Commercial Data Masking Facility algorithm.

Decryption in DES is done with the same algorithm as encryption; however, the key is processed in reverse order. This is an internal process; the black box merely needs to be told whether to do encryption or decryption.

Triple DES.

Because the strength of DES encryption is dependent on key length, one effective way to increase key length and maintain compatibility with existing key management facilities is to consider encrypting two or three times by using two or three keys. The trade-offs are cryptographic strength and speed. Cryptoanalysts have demonstrated that double DES is no more secure than single Data Encryption Standard. One of the better ways to implement triple DES is to use two independent 56-bit keys (or one 112-bit key) in EDE mode. This three-step encryption process uses key 1 to encrypt, key 2 to perform decryption of the output from the first encryption, followed by encryption reusing key 1. The decryption step will not produce readable plaintext because key 2 is different than key 1. The decryption process is the converse: decrypt with key 1, encrypt with key 2, followed by decryption with key 1.

International Data Encryption Algorithm (IDEA)

The Interactive Data Extraction and Analysis was invented by James Massey and Xuejia Lai of Switzerland. It is an iterated block cipher with a structure similar to DES, but it uses only eight iterations compared with the 16 used byDES with a 128-bit key. Its software implementations are faster thanDES and much faster than triple Data Encryption Standard. Cryptographers believe that the mathematical theory on which IDEA is based makes it secure. IDEA also uses a longer key than even triple DES.

RC2

The algorithm for RC2 was designed by Ronald Rivest of Rivest_Shamir-Adleman Data Security, Inc. and has not been officially published, although details about the

Previous screen

algorithm are available to those who sign nondisclosure agreements with RSA. RC2 is a 64-bit block cipher. It accepts keys with variable lengths from 0 to the maximum supported by computers. The key is processed to generate an internal key of 1,024 bits. Special export status has been given to versions of RC2 using a 40-bit key.

RC4

The algorithm for RC4 was also designed by Ronald Rivest of RSA Data Security, Inc. and has not been officially published, although someone has placed the algorithm in the public domain by posting details about implementing the code on the Internet. Rivest_Shamir-Adleman still claims it is proprietary. Like RC2, RC4 uses a variable size key and is implemented as a stream cipher and works in output feedback (OFB)mode. Its characteristics are similar to those described for DES in OFB mode. Special export status has been given to versions of RC4using a 40-bit key.

Symmetric Key Cipher Modes of Operation

Key cipher modes are susceptible to codebook attack or propagation of errors. Thus, the following technical information may help users understand why one mode is preferred over another. Most of these modes are applicable to other symmetric key algorithms besides DES.

Electronic Codebook Mode (ECB).

ECB works best for messages with short and random data. DES is used on 8-byte blocks of data with no context information added; hence, whenever the same data blocks are encrypted, the same ciphertext results. Susceptible to codebook collection and substitution or block replay attacks, it is used primarily for sending keys and initialization vector, because the data are short and random.

Cipher Block Chaining Mode (CBC).

CBC is best for encrypting files stored locally. Each block ofciphertext is made to depend on its predecessor by XORing each block of plaintext with the previous block of ciphertext before encryption. This mode prevents codebook collection problems, but it remains susceptible to block replay attacks, because each block that passes through the cipher is modified by the block preceding it. It uses a shared initialization vectors and last-block padding. Error propagation is as follows: 1-bit ciphertext errors become 1-block plaintext errors; ciphertext synchronization errors (i.e., dropped bits) garble all subsequent plaintext output unless blocking is provided.

Cipher Feedback Mode (CFB).

CFB is best for encrypting a stream of data one character at a time(e.g., between a terminal and a host). It can be implemented as an*n*-bit stream cipher. With 8-bit CFB, encryption is able to start and operate on each 8-bit character. The last ciphertext block sent is fed back into the encryptor, which prevents codebook collection problems. DES is used only in encryption mode. It requires a unique initialization vectors for each message to avoid being vulnerable to replay attacks. Errors will propagate. For example, 8-bit CFB will produce 9 bytes of garbled plaintext if a 1-bit error has occurred in the ciphertext.

Previous screen

Output Feedback Mode (OFB).

OFB is best for error-prone environments. It is used for handlingasynchronous data streams (keyboard input) because most of the“cipher” is performed based on prior input and is ready when the last *n*-bit character is typed in. OFB requires an initialization vectors. This prevents codebook collection problems, but OFB remains susceptible to block replay attacks. Errors do not propagate or extend. However, several analyses have shown that unless the *n*-bit feedback size is the same as the block size, the algorithm strength will be significantly reduced.

Key Management Problems

The following problems often surface with the use of private, symmetric key-based ciphers.

Key Population Does Not Scale Well with User Growth.

(n)(n - 1)/2 keys are required for n people to communicate securely. For example, 19,900 keys are required for 200 people to communicate with each other securely and 1,999,000 keys are required for 2,000 people to so communicate.

A Secret Message Cannot Be Sent without Prior Arrangement.

It is impossible for operators to send a secret messages unless they already have the ability to send secret messages. In reality, senders must be able to trust the person (or their credentials) from whom they accept an encryption key. If they cannot trust them, they should not expect that messages encrypted with the key they delivered will remain secret.

Key Revocation Is Difficult.

There are no mechanisms for revoking keys that have been lost or compromised (without a centric based system).

Such issues can be resolved by using secure protocols with a trusted centralized key distribution center.

Cryptographic Protocols

The purpose of using a protocol with cryptography is to prevent or detect eavesdropping or cheating. Thus the parties involved can authenticate themselves to each other and establish a secret shared key with which they can encrypt their messages to preserve secrecy. If a new encryption key is established each time they need to communicate or is given a reasonably short lifetime, the amount of time a replay attack can be successfully mounted is limited. An effective protocol does not rely on the fairness or trustworthiness of the parties or the message network. Today there are several secure protocols that have been implemented experimentally. MIT's Project Athena was one of the first to tackle the tough problems associated with key management for private (symmetric secret) key cryptographic systems with a secure protocol called Kerberos. Versions of Kerberos code are available via the Internet or as supported products from several vendors. Other protocols that also perform user authentication and key management and are worth considering are Krypto-Knight, SESAME (Secure European System for Applications in a Multivendor Environment), S-Key, Yaksha, and Radius.

Previous screen

Kerberos

The goal of Kerberos, a trusted third party protocol, is to provide unforgeable credentials to identify principals and to provide them with shared symmetric cipher keys to preserve the privacy of any transactions during their session. The principals are users, services(applications) on servers, or hosts. The Kerberos protocol assumes all principals sit on an untrusted network, except the trusted Kerberos servers, which must reside in a secure environment. Kerberos provides two major services as a trusted third party: authentication of principals and key management. Kerberos keeps a data base of each principal's secret—a hash function of each principal's password. When client machines can be used by more than a single user, network nodes or services can and should also be authenticated so session audits can indicate which workstation was involved in a user session.

Authentication of Principals.
Trusted third party functions include the following:

- **Identity-based authorization model.** Because Kerberos knows every principal's identity, it can create messages to convince one principal of another principal's identity.
- **A trusted third party performs the authentication.** The Kerberos servers are the linchpin. They must be physically and logically secure, with effective security and administration policies and procedures, to ensure they cannot be compromised. Their implementation must provide high availability.
- **Symmetric secret key exchange.** Kerberos provides one-time random session keys to both principals with a finite lifetime to limit exposure to replay attacks.
- **Principal.** This designation identifies users (user ID) or services (service ID) to Kerberos. (For discussion purposes, the principal is commonly referred to as the user.)

Key Management.
The functions of key management include the following:

- **Key distribution between users and services.** Session keys are securely distributed to both parties involved.
- **Key creation.** The Kerberos implementation provides excellent random key generation. It is questionable whether individual users would be able to provide such a facility.
- **Key revocation.** If a user ID has been compromised or a session key has expired(without a renewal request), Kerberos is able to revoke all outstanding session keys established on behalf of a user.
- **Single sign-on.** Users need only remember their Kerberos ID and pass phrase. Kerberos eliminates the burdens of remembering the IDs and passwords for multiple systems by passing tokens representing previously authenticated users to any system the user wishes to access.

The services provided by Kerberos are provided by an authentication server (AS) and a ticket-granting server (TGS), which reside in a secure environment as shown in Exhibit 4.

Kerberos Message Exchange

Kerberos Message Exchange

To obtain access to a server, s, a user must have a valid ticket and must be authenticatable to the server. To initiate the process, users identify themselves in a request to an AS to obtain credentials from the TGS. This request is represented as message 1 in Exhibit 4 and is not encrypted. The message flow and action scenario that follows is the same for Kerberos versions 4 and 5; the message formats shown are for version 5. Only messages 2 and 4 are different, where double encryption of the ticket-granting ticket (TGT) has been eliminated. Exhibit 5 provides a key to the Kerberos nomenclature used in the following steps.

- User is prompted for ID; client sends plaintext message to the AS requesting a TGT (TGT = Tc,tgs).
- as_req = c, tgs, expiry, n
- AS looks up the Hash of the user's password from the Kerberos data base to form Kc to encrypt a response message and send it to the user. The response message contains a secret key, Kc,tgs, to be used to encrypt ticket request messages between the client and the TGS and the TGT, Tc,tgs, encrypted with the TGS's secret key, Ktgs.
- as_resp = Kc{Kc,tgs},Ktgs{Tc,tgs}.
- The user is prompted for a password, which the client uses to form Kc and decrypt the message. The value, chksm, included in the message is compared with a checksum calculated by the client. If they are the same, it is assumed that the password provided by the user is correct. If it is not correct, the user must try again.
- The secret key, Kc,tgs, is decrypted using Kc. Kc,tgs and the encrypted TGT, Ktgs{Tc,tgs}, are stored on the client so a request for services from server s can be made to the TGS. The user's password and the key Kc are erased from the client's memory to minimize exposure to snooping attacks at the client.
- The user's password was never exposed to the network. The client was able to decrypt the response from the AS because they shared a secret: knowledge of the user's password.
- Kerberos principals obtain tickets for server services from the TGS. Principals are either clients or services on servers. Tickets are encrypted in the key of the service, s.
- Encrypted Ticket Ks{Tc,s} = Ks{c, s, IPc, tstmp, chksm, expiry, Kc,s}
- A ticket is used to securely pass the name of the client requesting service to the server. To guard against replay attacks, all tickets are sent to servers with an authenticator encrypted with the session key shared by the client and the server.
- Encrypted Authenticator Kc,s{Ac} = Kc,s{c, IPc, tstmp, chksm}
- All messages contain a checksum which is checked to determine whether the message has been modified by transmission errors or malicious intent.

c	= client	K_i	= i's secret key
s	= server	$K_{c,s}$	= session key shared by c and s
IP_c	= client's network address	$K_s\{M\}$	= message M, encrypted in s's key
n	= nonce (random number)	$T_{c,s}$	= c's ticket to use s
tstmp	= time stamp of current time	$A_{c,s}$	= authenticator from c to s
chksm	= checksum	$K_s\{T_{c,s}\}$	= c's ticket to use s encrypted in s's key
expiry	= ticket start and end times	$K_{c,s}\{A_{c,s}\}$	= encrypted authenticator from c to s

Previous screen

- When the user decides that access is required on server, s, the client does the following:
- Sends a message containing the encrypted authenticator and the encrypted TGT requesting service from servers to the TGS.
- tgs_req = s, expiry, n, Ktgs{Tc,tgs},Kc,tgs{Ac}.
- The TGS is able to decrypt the TGT because it is encrypted in its secret key. If the message chksm is correct, and the other information it previously generated is correct, the TGS extracts the client-TGS session key, Kc,tgs, from the decrypted TGT and decrypts the client's authenticator. The TGS then compares and checks the chksm and whether the tstmp is current enough to be considered valid. If it is valid, the following occurs:
- The TGS sends the client an encrypted ticket to access the server, s, and an encrypted session key, Kc,s, for the client/server session. The ticket is encrypted with a key formed from the server's password which the server shares with the TGS. The encrypted ticket and the session key Kc,s is encrypted with the session key Kc,tgs, used by the client and the TGS.
- tgs_resp = Kc,tgs{Kc,s},Ks{Tc,s}.
- The client then sends its authenticator and ticket to the server, s, which uses the authenticator to verify that the ticket came from the client, c, and recovers the client/server session key Kc,s from ticket, Tc,s.
- s_req = Ks{Tc,s},Kc,s{Ac}
- When mutual authentication is required, the server sends a message to the client encrypted with the session key containing the tmstmp generated by the client incremented by 1. This could not have been done correctly if the server did not generate the secret key, Ks, based on the server's password and shared by the server and the TGS to decrypt the ticket. The ticket contains the client/server session key.
- s_resp = Kc,s{timestmp+1}
- TGSs should cache authenticators that are still valid according to their time stamp and expiry of their TGT, so they can be checked for new requests for service to guard against replay attacks.

Kerberos Nomenclature

Kerberos Shortcomings

The Kerberos protocol demonstrates several shortcomings.

- Because the AS responds without authenticating the user(based on the assumption that only the valid user would know the shared secret), an eavesdropper could record request-response pairs to form the basis of a dictionary attack. However, the nonce, *n*, is supposed to limit an attacker's ability to do this because it should change the appearance of the encrypted response message. Primarily, because the nonce is sent in the request message, an attacker's valid time to calculate the key is limited to the life of

Previous screen

the nonce. This time should be much shorter than the time predicted to determine the key.

- Access control (i.e., a user's privileges on server, s) must be provided and maintained on each server that the user accesses, instead of maintained centrally.

- Most Kerberos implementations have served a relatively small population (thousands) when implemented by a university or corporation. Each implementation can be considered a realm. Interrealm authentication can only be provided as a special nonstandard feature by a few vendors. A single Kerberos server with a large user population could present problems as a performance bottleneck and a single point of failure, both of which could prevent network access. Improvements in newer versions of Kerberos will include interrealm authentication.

- Workstations used by more than one user are prime candidates for swiping session keys and TGT for current replay attacks or later dictionary attacks if a user has access to a debugger program. If the client Kerberos code is bugged so that all IDs and passwords are written to hidden files for later examination of the system, these accounts will be compromised.

- The Kerberos servers present a single point of failure which could prevent user access or whose security, if breached, could compromise the security of all user sessions. Clearly, a server implementation with high availability attributes is necessary. Alternatives utilizing distributed authentication servers may be able to limit security breaches to a smaller portion of the user population.

Conclusion

Protocols have been invented that provide solutions to many of the problems found with private (symmetric) key-based ciphers. Most of these protocols, such as Kerberos, are complex, and their implementations may not solve all of the problems. How does Kerberos rate in terms of the problems outlined earlier?

- **Key population does not scale well with user growth.** This problem was only partially solved. Kerberos must be able to serve n users and keeps only their n hashed passwords. It issues secret session keys only when required. However, if all users required simultaneous sessions, n(n-1)/2 secret keys would still be required—an unlikely scenario. Growth across realms remains a problem.

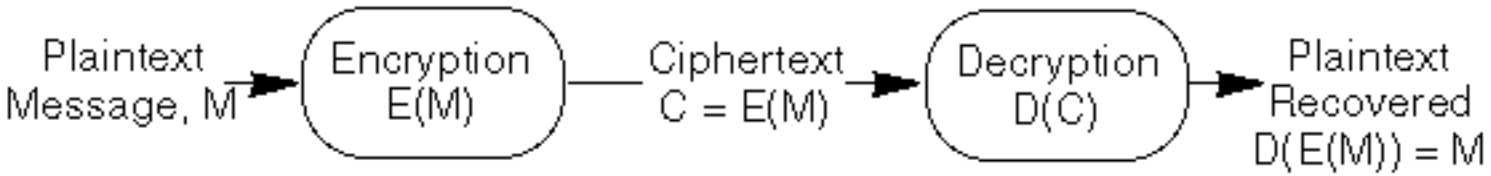
- **A secret message cannot be sent without prior arrangement.** This problem was solved once the “change the initial easy password” process was set in motion.

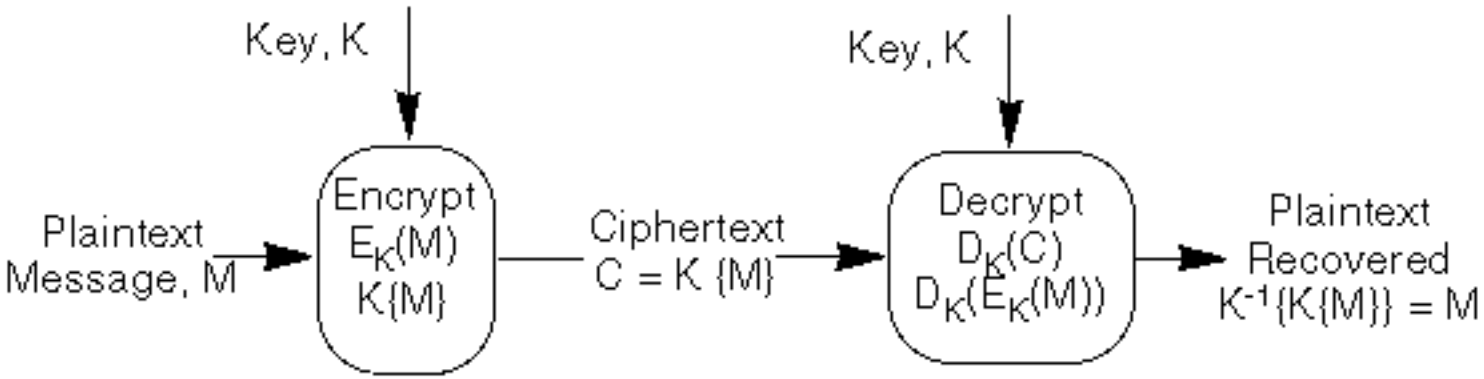
- **Key revocation difficult.** Kerberos solves this problem by administrator intervention.

- **Message originator cannot be uniquely identified.** The originator of a message cannot be identified based on knowledge of the key; it can only be narrowed down to those having knowledge of the key. The Kerberos protocol solved this problem by using a client authenticator. However, the solution was not generalized.

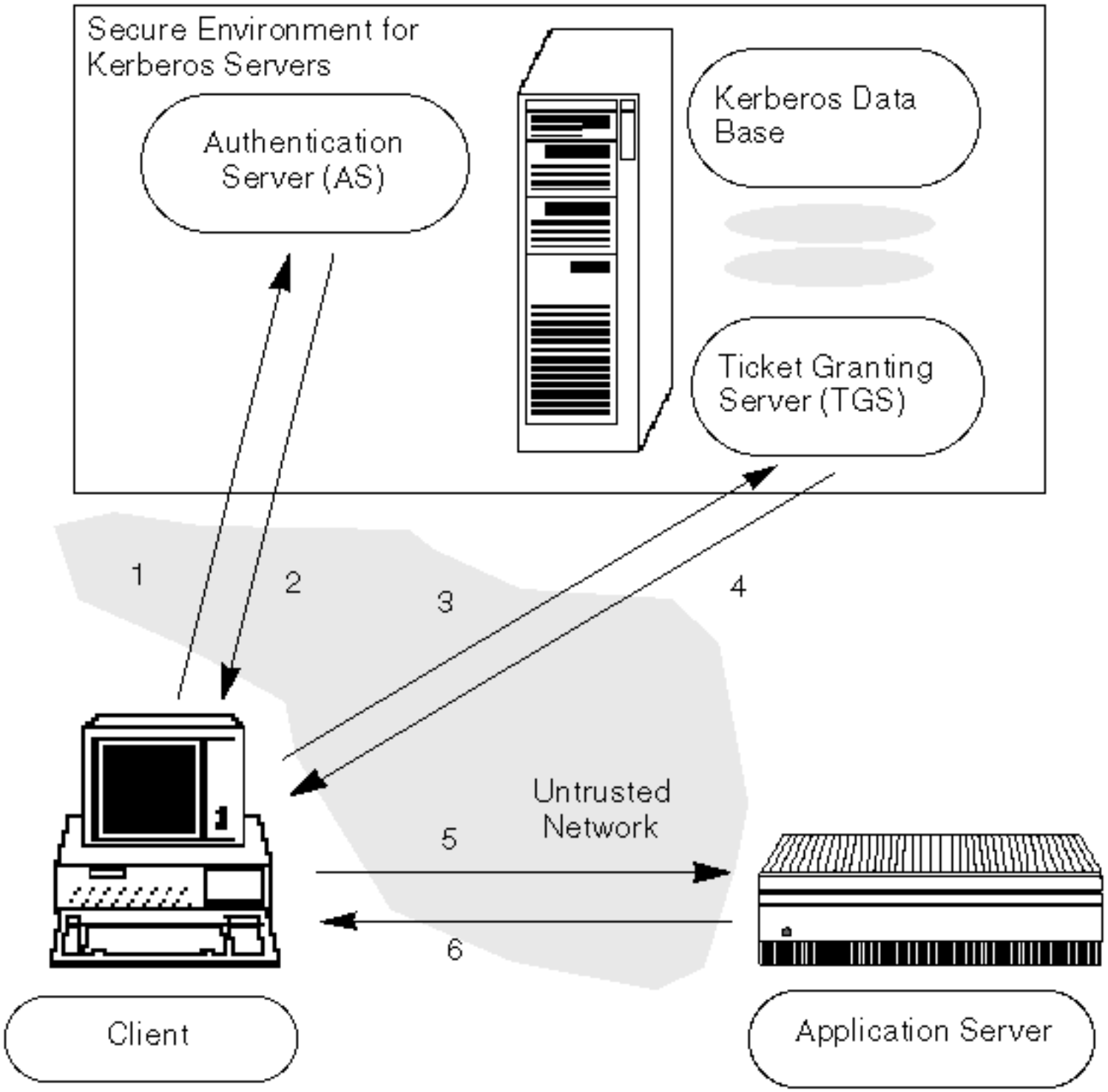
Author Biographies

Alex Bidwell
Alex Bidwell is president of CyberQuest, Inc.


Plaintext
Message, M
Encryption
E(M)
Ciphertext
C = E(M)
Decryption
D(C)
Plaintext
Recovered
D(E(M)) = M


Key, K
Key, K
Plaintext
Message, M
Encrypt
$E_K(M)$
K{M}
Ciphertext
C = K {M}
Decrypt
$D_K(C)$
$D_K(E_K(M))$
Plaintext
Recovered
$K^{-1}\{K\{M\}\} = M$

Attack Type	Possible Effects		
	Secrecy	Integrity	Authenticity
Passive (Eavesdropping)	Yes	No	No
Active	Maybe	Yes	Yes
Brute force	Yes	No	No
Known plaintext	Yes	No	No
Chosen plaintext	Yes	Yes	Maybe
Replay	Maybe	Yes	Yes
Cut-and-paste	Maybe	Yes	Yes
Time resetting	Maybe	Yes	Yes
Man-in-the-middle	Maybe	Yes	Yes


Secure Environment for
Kerberos Servers
Authentication
Server (AS)
Kerberos Data
Base
Ticket Granting
Server (TGS)
1
2
3
4
Untrusted
Network
5
6
Client
Application Server