RAMAIAH
Institute of Technology

M.S. Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of Computer Science and Engineering

Course Name: Cryptography and Network Security
Course Code - CSE643
Credits - 3:0:0
UNIT - 4
Term: March 2022 – July 2022

Prepared by: Dr. Sangeetha. V
Assistant Professor

Textbooks

1. Behrouz A. Forouzan, Debdeep Mukhopadhyay: Cryptography and Network Security, 2nd Edition, Special Indian Edition, Tata McGrawHill, 2011.

2. William Stallings, Cryptography and Network Security, Fifth Edition, Prentice Hall of India, 2005.

Reference Book:

1. Josef Pieprzyk, Thomas Hardjono, Jennifer Serberry Fundamentals of Computer Security, Springer, ISBN 978-3-662-07324-7.

OUTLINE - (Text 2)

Message authentication:

(Chapter 12.1 to 12.3)

- Authentication Requirements
- Authentication Functions
- Message Authentication Codes

Digital signatures: (Chapterr 13.1)

- Digital Signatures
- Digital Signature Algorithm

Key management and distribution:

(Chapter 14.3,14.4)

- Distribution of public keys
- X.509 certificates

Kerberos (Chapter 15.3)

Introduction

Authentication is the process of verifying who someone is.

Example : validating your credentials like User Name/User ID and password to verify your identity.

Authorization is the process of verifying what specific applications, files, and data a user has access to.

Introduction

Integrity involves maintaining the consistency, accuracy and trustworthiness of data over its entire lifecycle.

Data integrity is the assurance that digital information is uncorrupted and can only be accessed or modified by those authorized to do so.

Introduction

- In information security, Message Authentication is a property that a message has not been modified while in transit (data integrity) and that the receiving party can verify the source of the message.
- Message authentication is typically achieved by using message authentication codes(MACs), authenticated encryption (AE) or digital signatures.

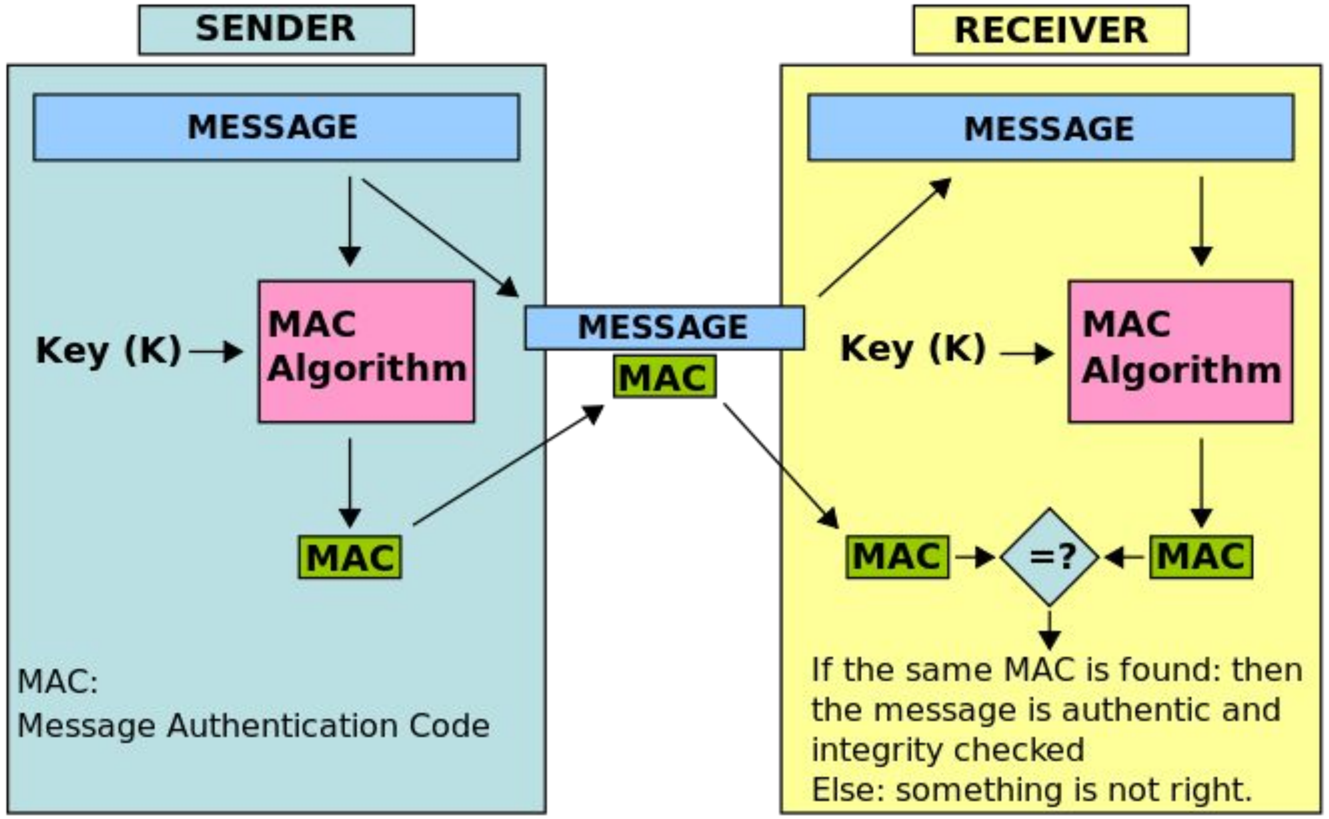
Introduction

- Message authentication can be provided using the cryptographic techniques that use secret keys.

- MAC algorithm is a symmetric key cryptographic technique to provide message authentication.

Introduction

- The MAC is used as an integrity check based on a secret key shared by two parties to authenticate information transmitted between them


SENDER
MESSAGE
Key (K)
MAC Algorithm
MAC
MAC:
Message Authentication Code
MESSAGE
MAC
RECEIVER
MESSAGE
Key (K)
MAC Algorithm
MAC
=?
MAC
If the same MAC is found: then the message is authentic and integrity checked
Else: something is not right.

Introduction

- Message Authentication Requirements
- Message Authentication Functions
- Message Authentication Codes

MESSAGE AUTHENTICATION FUNCTIONS

Message authentication mechanism has two levels of functionality.

- At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message.
- This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

RAMAIAH
Institute of Technology

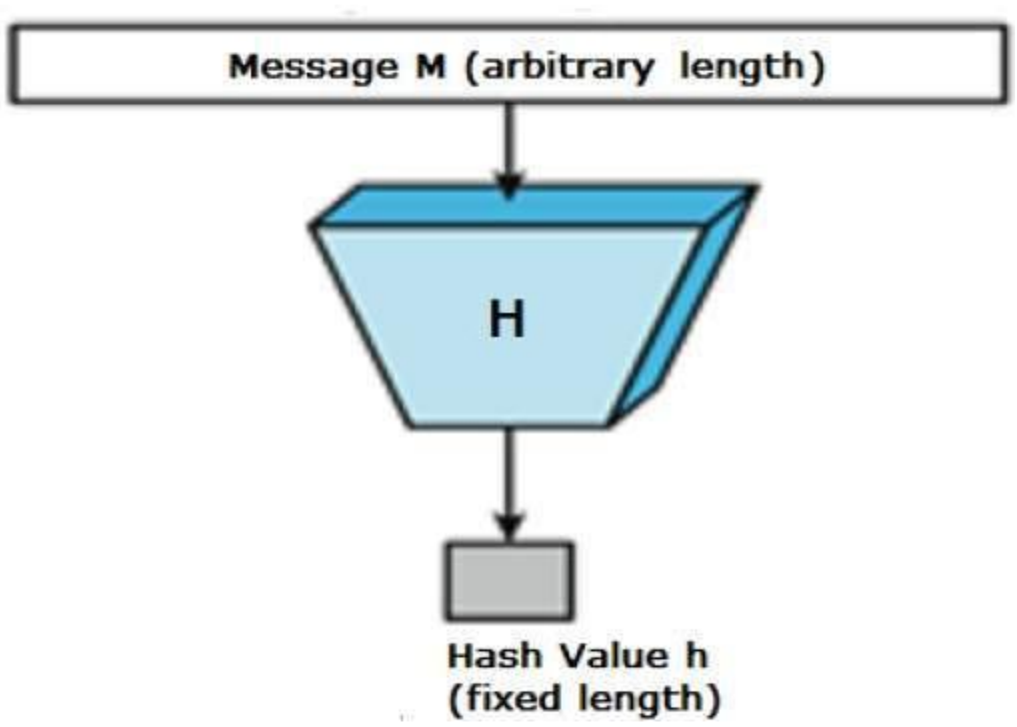
MESSAGE AUTHENTICATION FUNCTIONS

The types of functions that may be used to produce an authenticator are grouped into three classes.

1. Hash function
2. Message encryption
3. Message authentication code (MAC)

MESSAGE AUTHENTICATION FUNCTIONS

1. Hash function: A function that maps a message of any length into a fixed length hash value, which serves as the authenticator



MESSAGE AUTHENTICATION FUNCTIONS

2. Message encryption: The ciphertext of the entire message serves as its authenticator

    1. SYMMETRIC ENCRYPTION
    2. PUBLIC-KEY ENCRYPTION

3. Message authentication code (MAC): A function of the message and a secret key that produces a fixed-length value that serves as the authenticator.

MESSAGE AUTHENTICATION FUNCTIONS

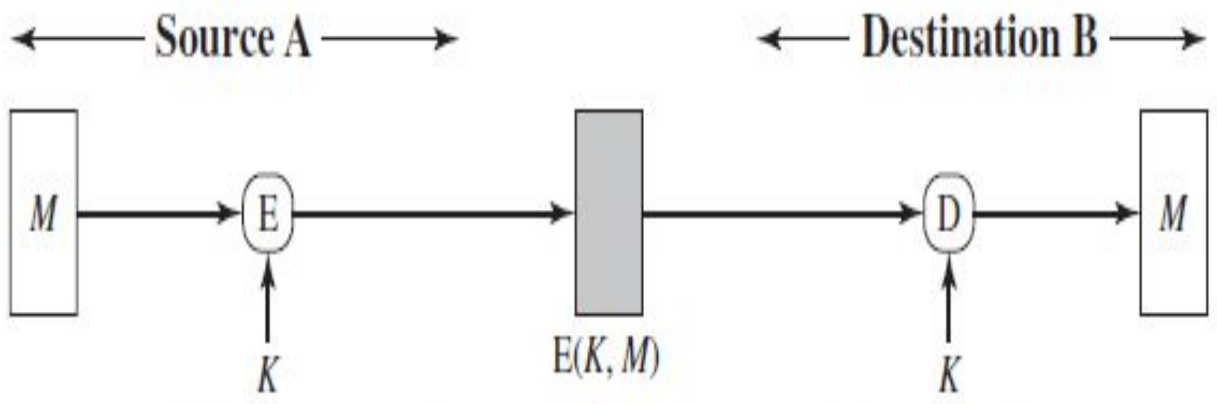
Message Encryption

1. SYMMETRIC ENCRYPTION
2. PUBLIC-KEY ENCRYPTION

MESSAGE AUTHENTICATION FUNCTIONS

Message Encryption - SYMMETRIC ENCRYPTION

- Symmetric encryption provides confidentiality.
- If no other party knows the key, then confidentiality is provided: No other party can recover the plaintext of the message.



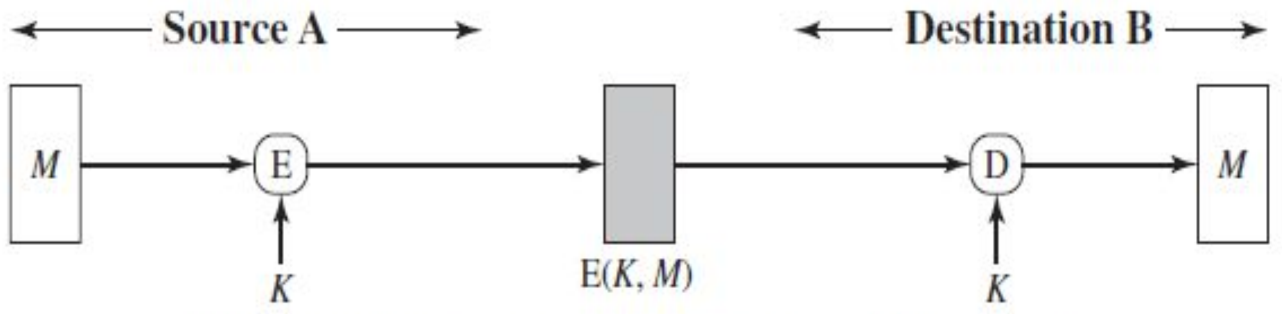
(a) Symmetric encryption: confidentiality and authentication

MESSAGE AUTHENTICATION FUNCTIONS

Message Encryption - SYMMETRIC ENCRYPTION

- Symmetric encryption provides <u>authentication</u>

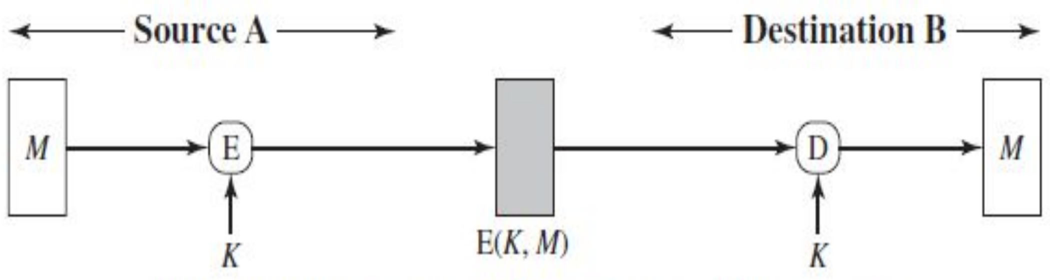
The message must have come from A, because A is the only other party that possesses K with the information necessary to construct ciphertext that can be decrypted with .



(a) Symmetric encryption: confidentiality and authentication

MESSAGE AUTHENTICATION FUNCTIONS

- Given a decryption function D and a secret key , the destination will accept any input X and produce output Y = D(K,X)
- If X is the ciphertext of a legitimate message produced by the corresponding encryption function, then Y is some plaintext message M. Otherwise Y is meaningless
- There may need to be some automated means of determining at B whether Y is legitimate plaintext received and therefore must have come from A.



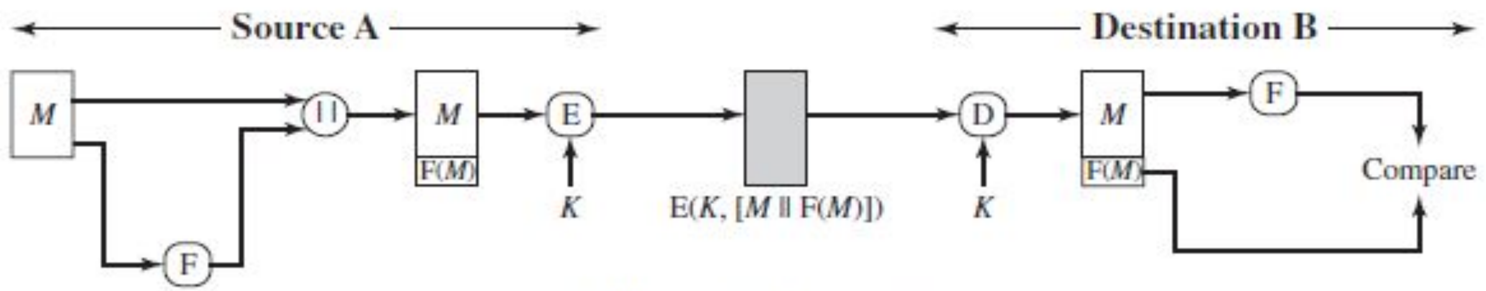
(a) Symmetric encryption: confidentiality and authentication

MESSAGE AUTHENTICATION FUNCTIONS

Solution : Force the plaintext to have some structure that is easily recognized and cannot be replicated without recourse to the encryption function.

Example, append an error-detecting code(frame check sequence (FCS)) or checksum, to each message before encryption.

The sequence is shown as internal error control

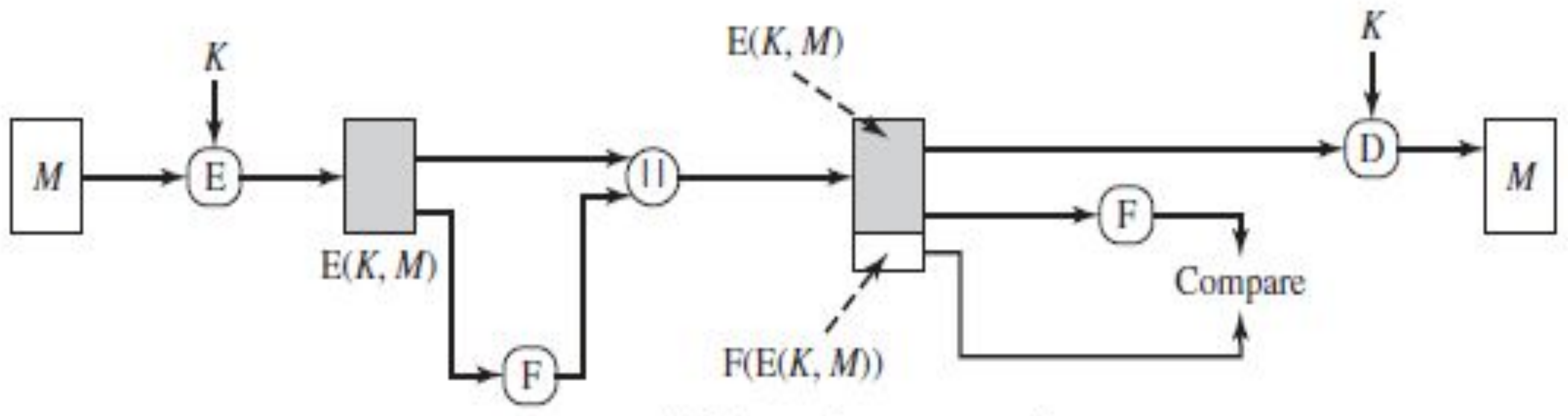


(a) Internal error control

If the calculated FCS is equal to the incoming FCS, then the message is considered authentic.

MESSAGE AUTHENTICATION FUNCTIONS

- Note that the order in which the FCS and encryption functions are performed is critical.
- The sequence in shown as **external error control**



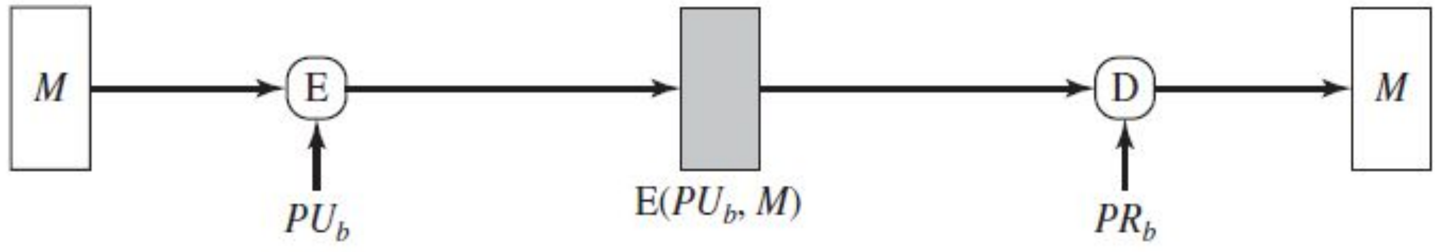
(b) External error control

MESSAGE AUTHENTICATION FUNCTIONS

Message Encryption - PUBLIC-KEY ENCRYPTION

The straightforward use of public-key encryption provides confidentiality but not authentication.

This scheme provides no authentication, because any opponent could also use B's public key to encrypt a message and claim to be A.

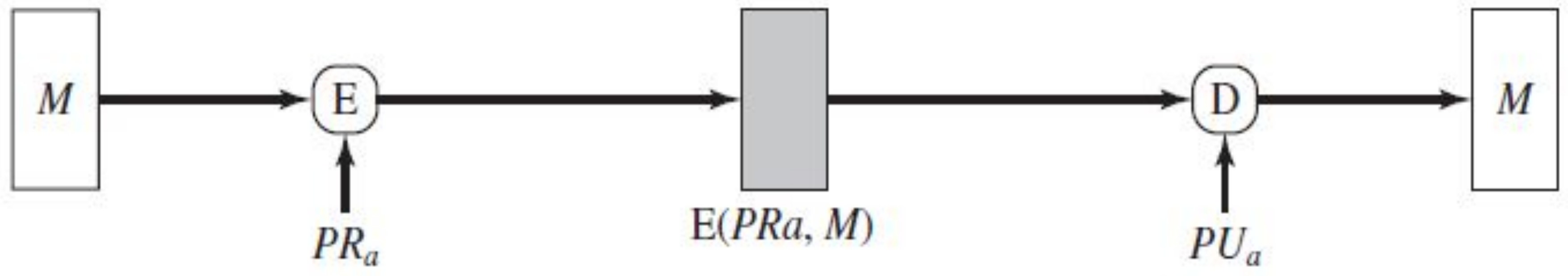


(b) Public-key encryption: confidentiality

MESSAGE AUTHENTICATION FUNCTIONS

Message Encryption - PUBLIC-KEY ENCRYPTION

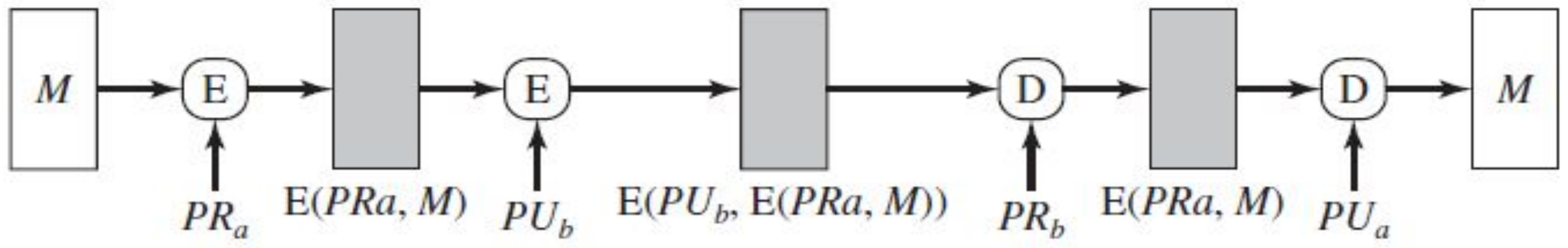
To provide authentication, A uses its private key to encrypt the message, and B uses A's public key to decrypt



(c) Public-key encryption: authentication and signature

MESSAGE AUTHENTICATION FUNCTIONS

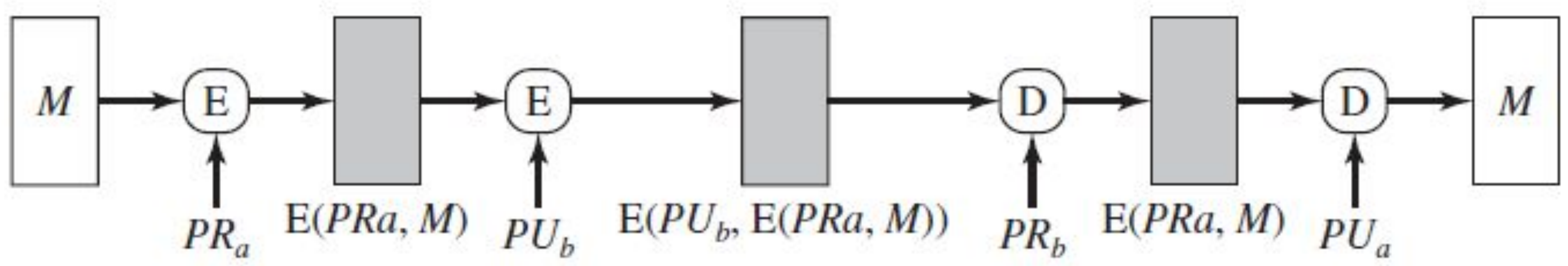
To provide both confidentiality and authentication, A can encrypt first using its private key, which provides the digital signature, and then using B’s public key, which provides confidentiality .



(d) Public-key encryption: confidentiality, authentication, and signature

MESSAGE AUTHENTICATION FUNCTIONS

The disadvantage of this approach is that the public-key algorithm, which is complex, must be exercised four times rather than two in each communication.



(d) Public-key encryption: confidentiality, authentication, and signature

MESSAGE AUTHENTICATION FUNCTIONS

2. Message encryption: The ciphertext of the entire message serves as its authenticator

    1. SYMMETRIC ENCRYPTION
    2. PUBLIC-KEY ENCRYPTION

3. Message authentication code (MAC): A function of the message and a secret key that produces a fixed-length value that serves as the authenticator.

MESSAGE AUTHENTICATION FUNCTIONS

Message authentication code (MAC)

- An authentication technique that involves use of a secret key to generate a small fixed-size block of data, known as a <u>**cryptographic checksum or MAC**</u>, that is appended to the message.
- MAC is a function of the message and the key: MAC = MAC(K, M)
- <u>Basic Uses of Message Authentication code</u> (MAC)

  (a) Message authentication

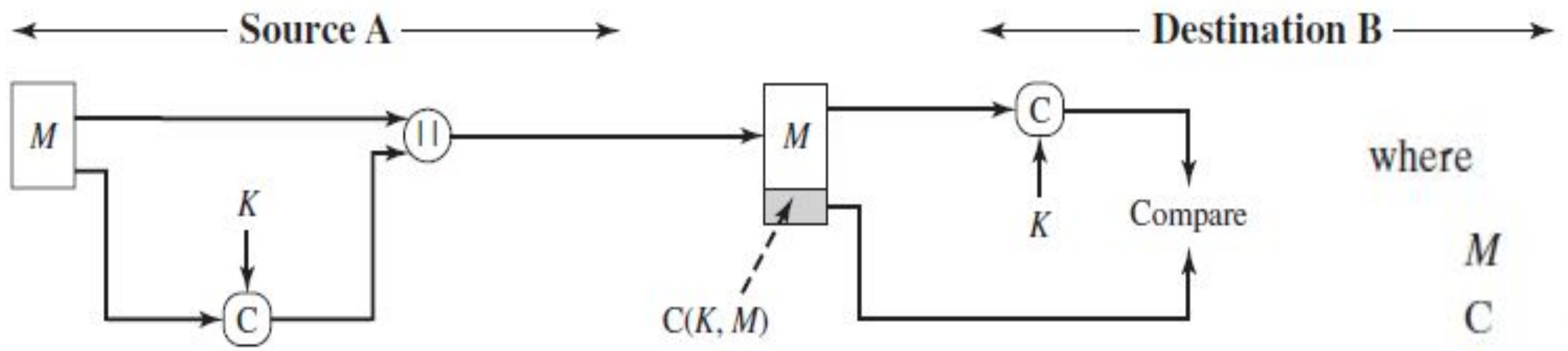
  (b) Message authentication and confidentiality; authentication tied to plaintext

  (c) Message authentication and confidentiality; authentication tied to ciphertext

MESSAGE AUTHENTICATION FUNCTIONS

Message authentication code (MAC)

- Two communicating parties, say A and B, share a common secret key. When A has a message to send to B, it calculates the MAC = C (K, M).
- The process provides authentication but not confidentiality, because the message as a whole is transmitted in the clear.



(a) Message authentication

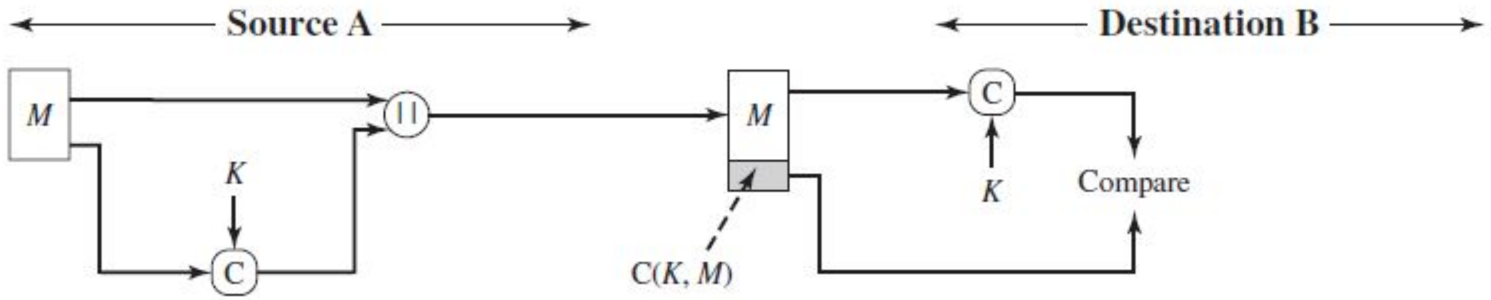
where

M	= input message
C	= MAC function
K	= shared secret key
MAC	= message authentication code

MESSAGE AUTHENTICATION FUNCTIONS

Message authentication code (MAC)

- The message plus MAC are transmitted to the intended recipient.
- The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC



(a) Message authentication

MESSAGE AUTHENTICATION FUNCTIONS

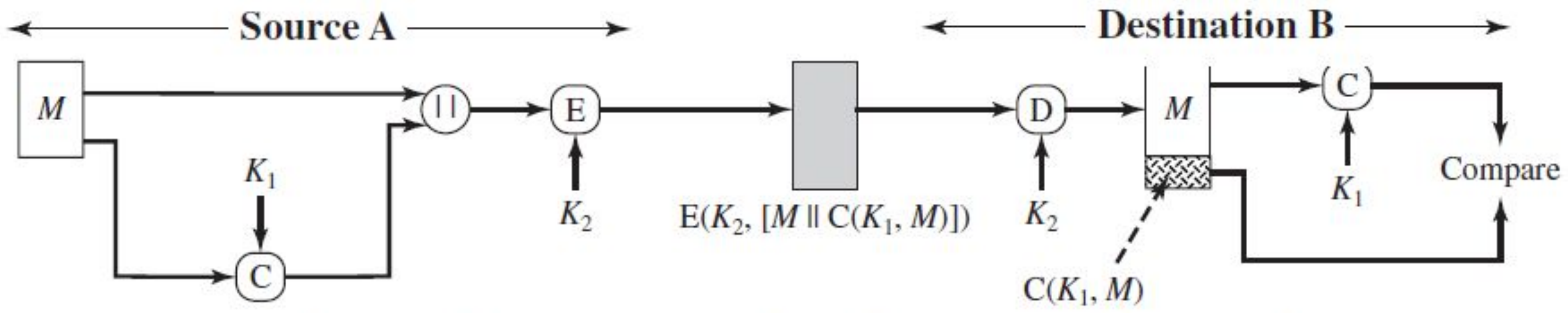
If we assume that only the receiver and the sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then

1. The receiver is assured that the message has not been altered.
2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.
3. If the message includes a sequence number, then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.

MESSAGE AUTHENTICATION FUNCTIONS

Message authentication code (MAC)

Confidentiality can be provided by performing message encryption either after the MAC algorithm.

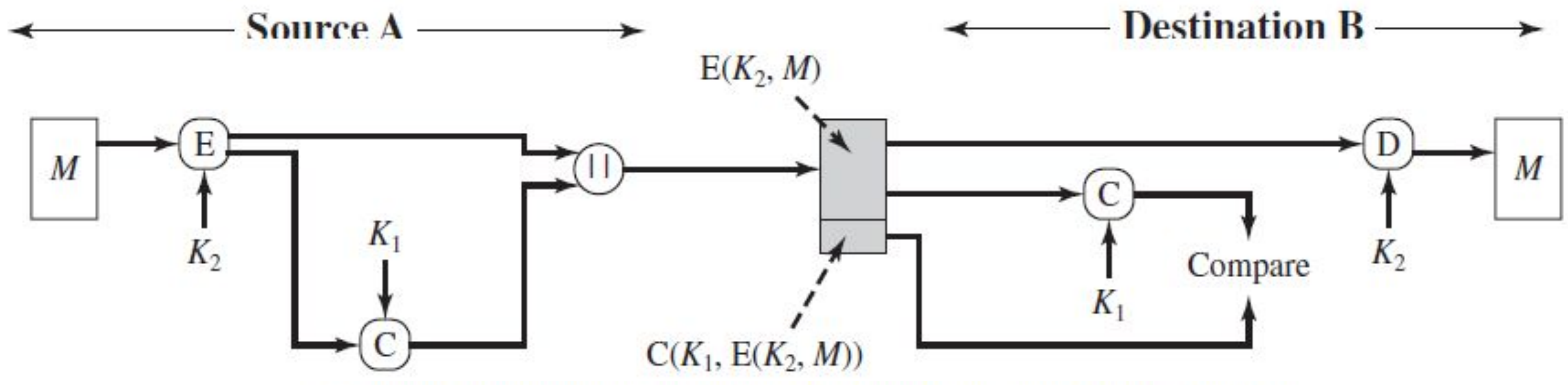


(b) Message authentication and confidentiality; authentication tied to plaintext

MESSAGE AUTHENTICATION FUNCTIONS

Message authentication code (MAC)

Confidentiality can be provided by performing message encryption either before the MAC algorithm.



(c) Message authentication and confidentiality; authentication tied to ciphertext

RAMAIAH Institute of Technology

MESSAGE AUTHENTICATION FUNCTIONS

Situations in which a message authentication code is used

1. There are a number of applications in which the same message is broadcast to a number of destinations. Examples are notification to users that the network is now unavailable or an alarm signal in a military control center.
2. Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages. Authentication is carried out on a selective basis, messages being chosen at random for checking.
3. Authentication of a computer program in plaintext is an attractive service. The computer program can be executed without having to decrypt it every time, which would be wasteful of processor resources. However, if a message authentication code were attached to the program, it could be checked whenever assurance was required of the integrity of the program.

MESSAGE AUTHENTICATION FUNCTIONS

Situations in which a message authentication code is used

4. For some applications, it may not be of concern to keep messages secret, but it is important to authenticate messages.

An example is the Simple Network Management Protocol Version 3 (SNMPv3), which separates the functions of confidentiality and authentication.

For this application, it is usually important for a managed system to authenticate incoming SNMP messages, particularly if the message contains a command to change parameters at the managed system.

MESSAGE AUTHENTICATION FUNCTIONS

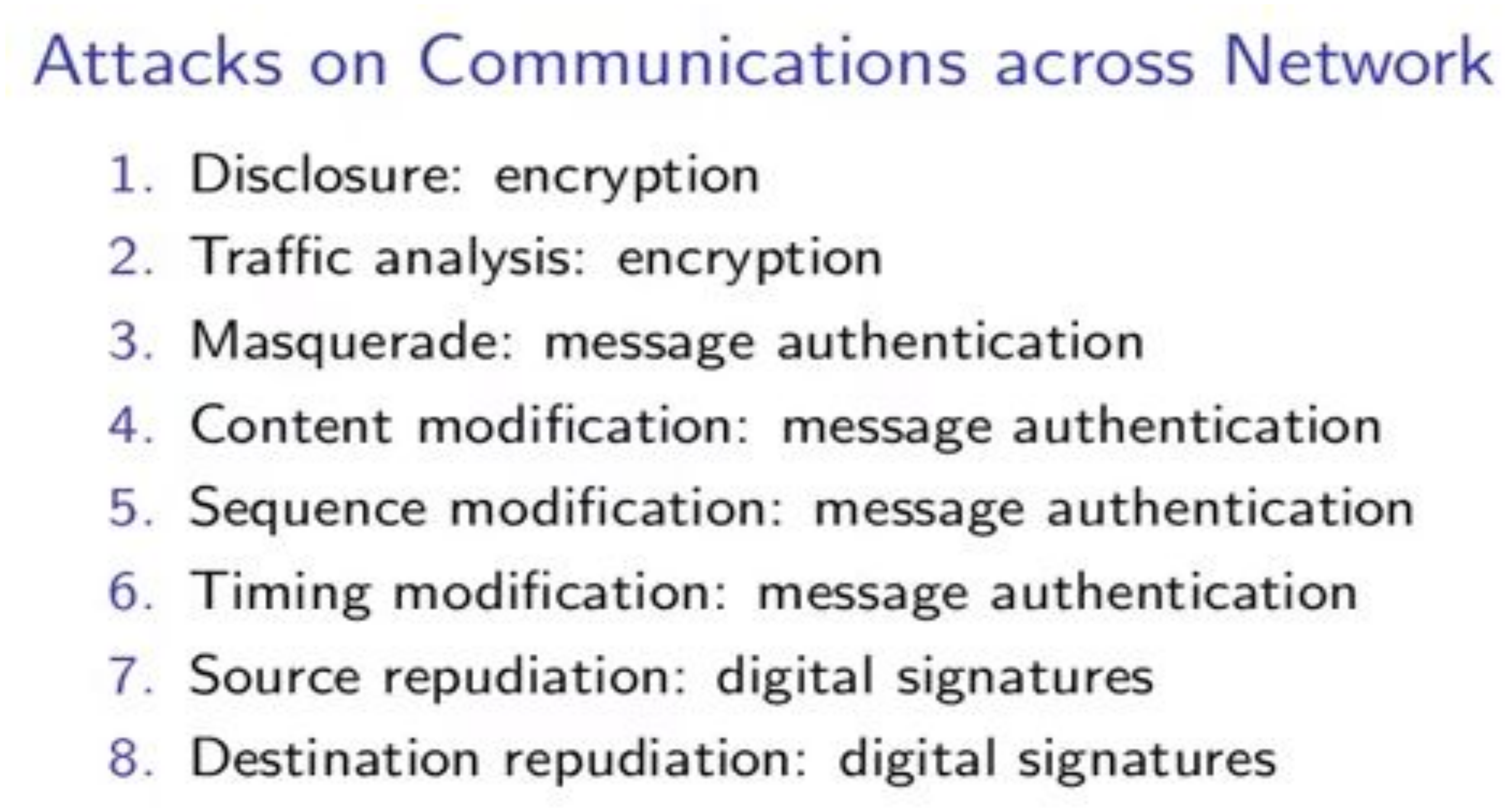
Situations in which a message authentication code is used

5. Separation of authentication and confidentiality functions affords architectural flexibility.

For example, it may be desired to perform authentication at the application level but to provide confidentiality at a lower level, such as the transport layer.

6. A user may wish to prolong the period of protection beyond the time of reception and yet allow processing of message contents. With message encryption, the protection is lost when the message is decrypted, so the message is protected against fraudulent modifications only in transit but not within the target system.

Message Authentication Requirements

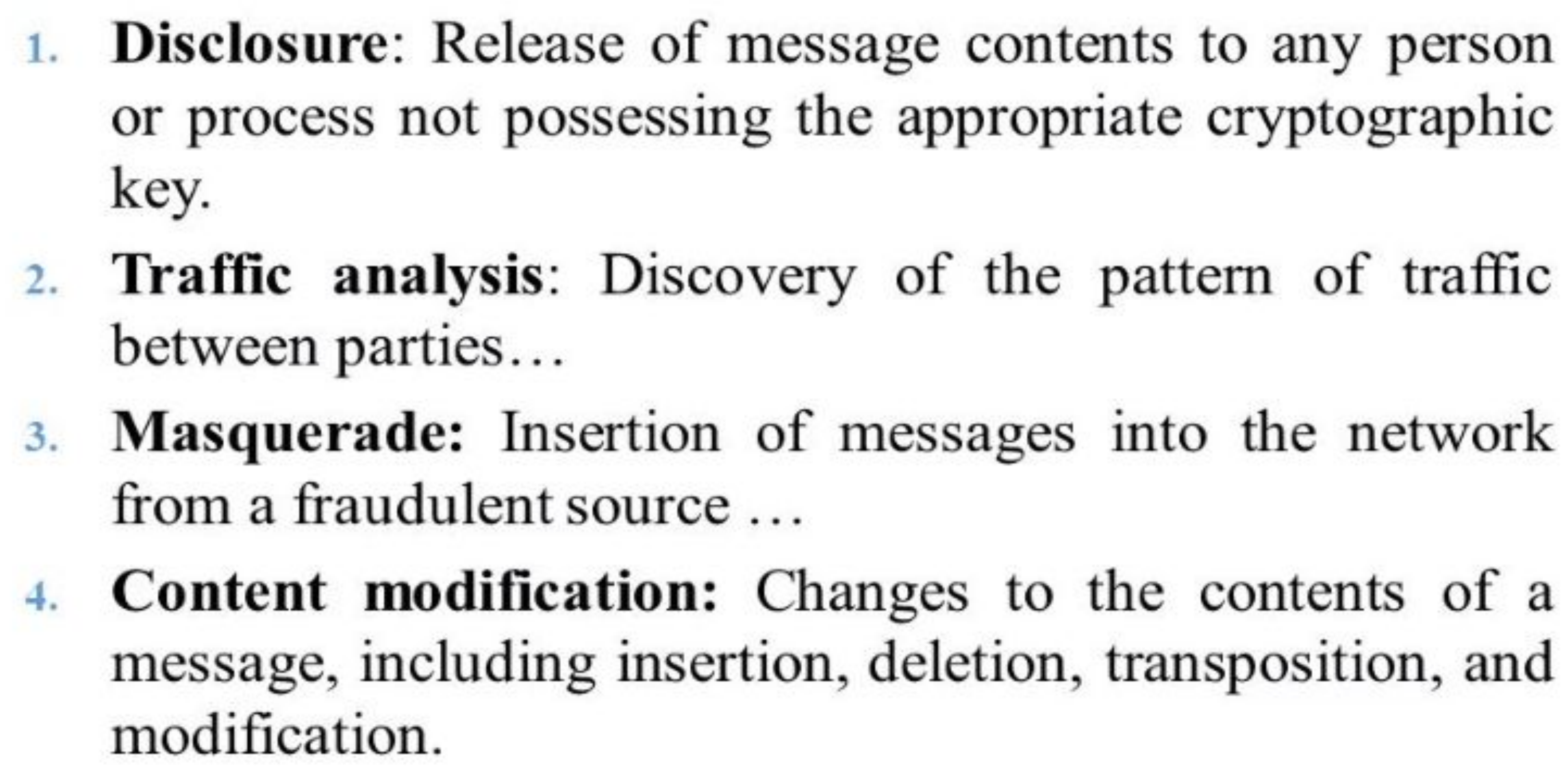


Attacks on Communications across Network

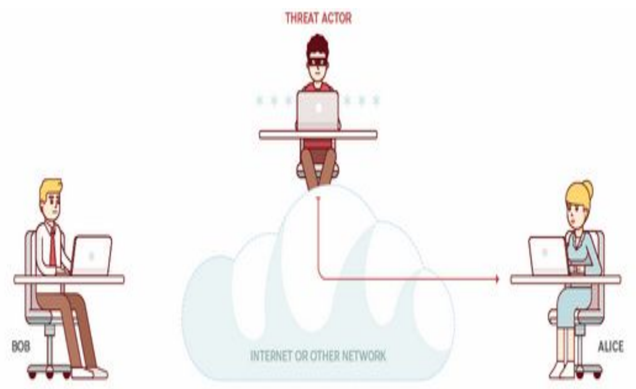
1. Disclosure: encryption
2. Traffic analysis: encryption
3. Masquerade: message authentication
4. Content modification: message authentication
5. Sequence modification: message authentication
6. Timing modification: message authentication
7. Source repudiation: digital signatures
8. Destination repudiation: digital signatures

RAMAIAH
Institute of Technology

Message Authentication Requirements

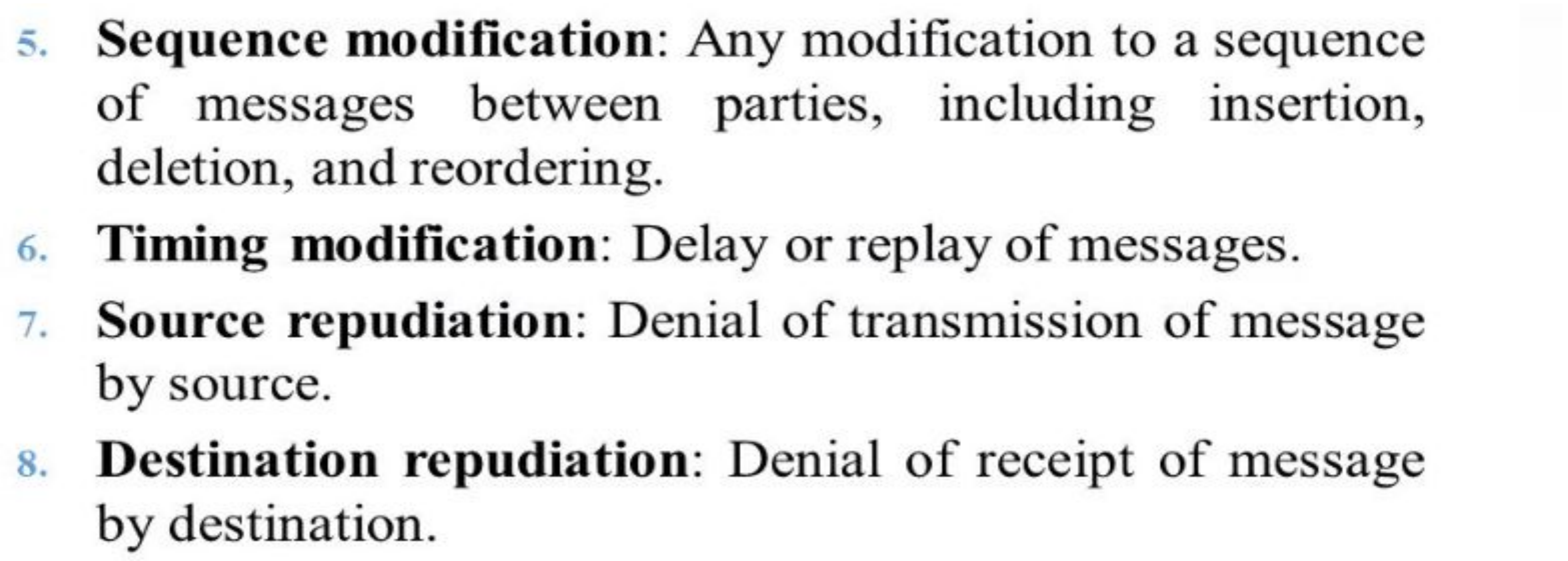


1. **Disclosure**: Release of message contents to any person or process not possessing the appropriate cryptographic key.
2. **Traffic analysis**: Discovery of the pattern of traffic between parties…
3. **Masquerade:** Insertion of messages into the network from a fraudulent source …
4. **Content modification:** Changes to the contents of a message, including insertion, deletion, transposition, and modification.



RAMAIAH
Institute of Technology

Message Authentication Requirements



5. **Sequence modification**: Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.
6. **Timing modification**: Delay or replay of messages.
7. **Source repudiation**: Denial of transmission of message by source.
8. **Destination repudiation**: Denial of receipt of message by destination.

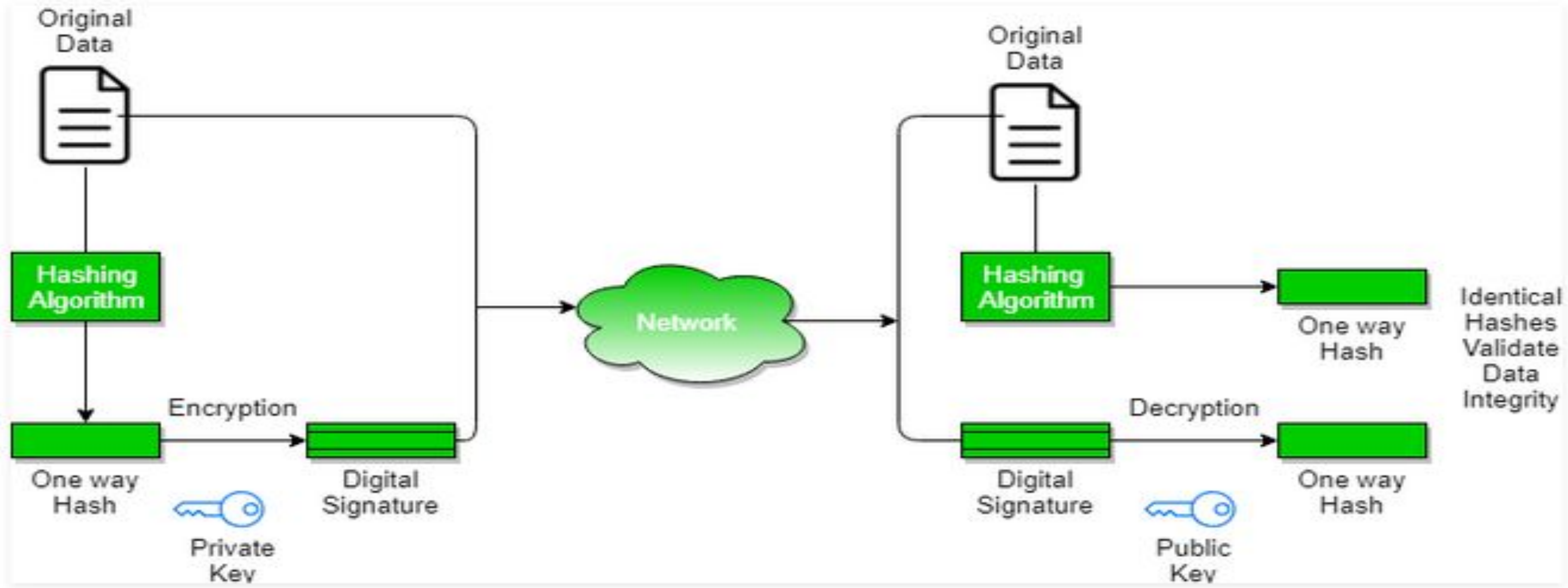
Digital Signatures

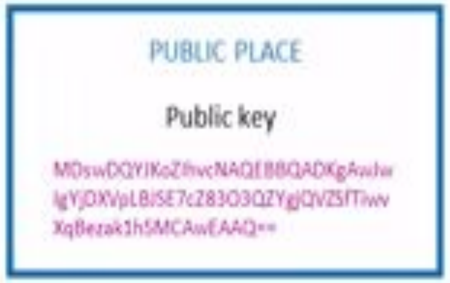
- A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software or digital document
- It's the digital equivalent of a handwritten signature or stamped seal.
- A digital signature is intended to solve the problem of tampering and impersonation in digital communications.
- Digital signatures are based on asymmetric cryptography.

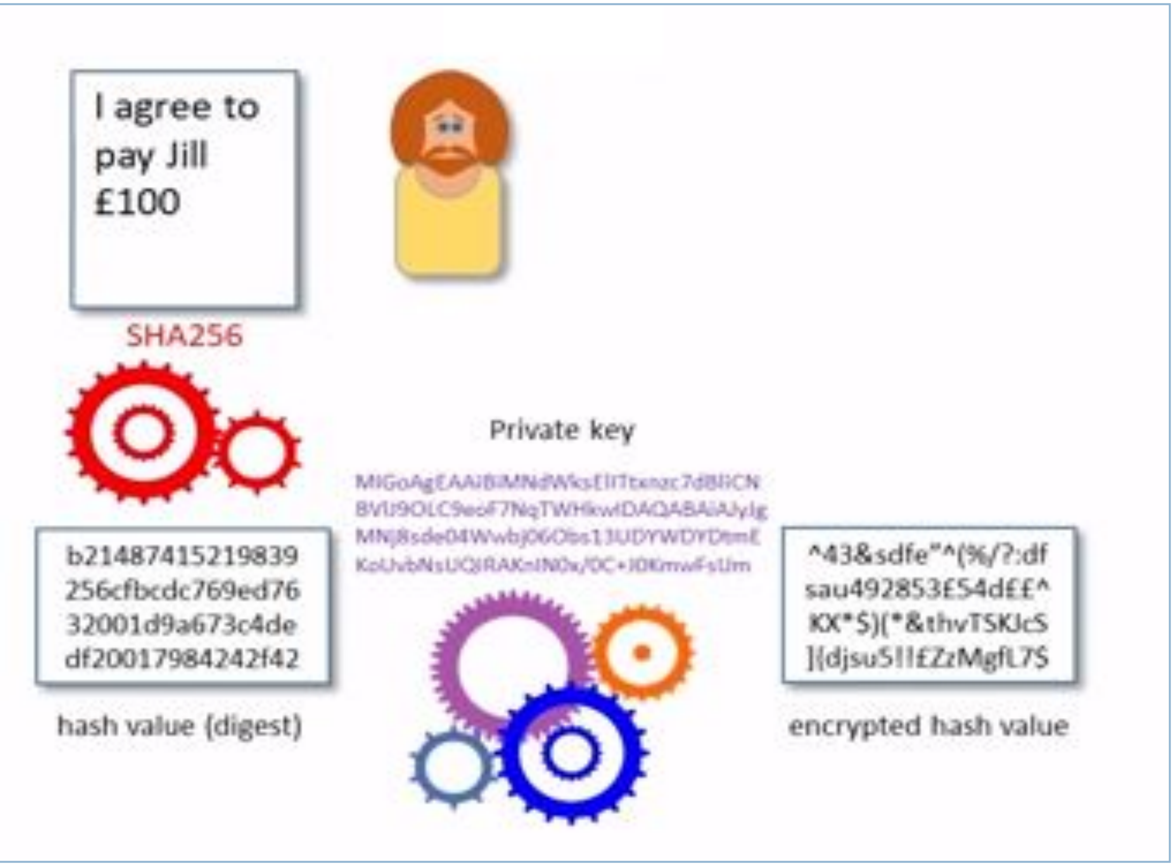
RAMAIAH
Institute of Technology

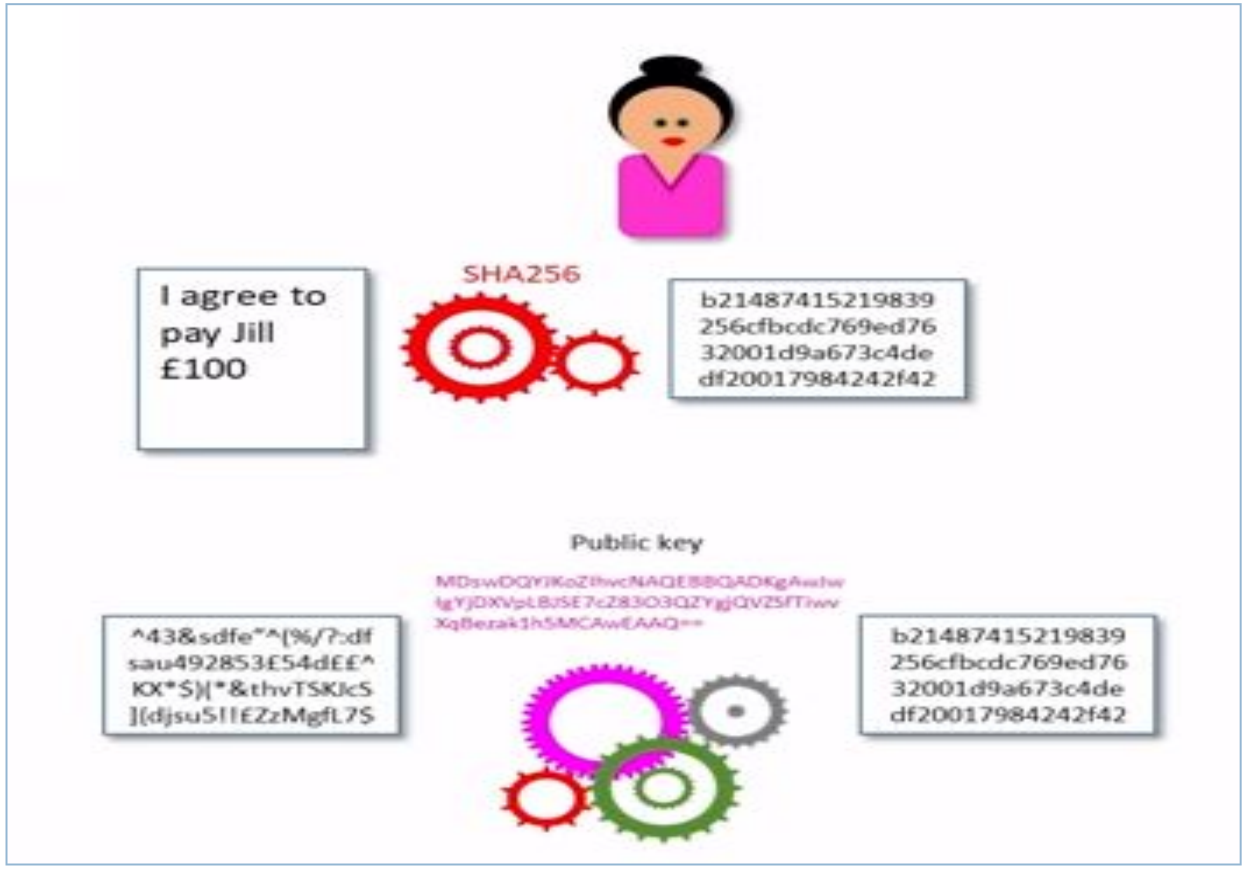
Digital Signatures

- A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature.
- Typically the signature is formed by taking the hash of the message and encrypting the message with the creator’s private key.
- The signature guarantees the source and integrity of the message.



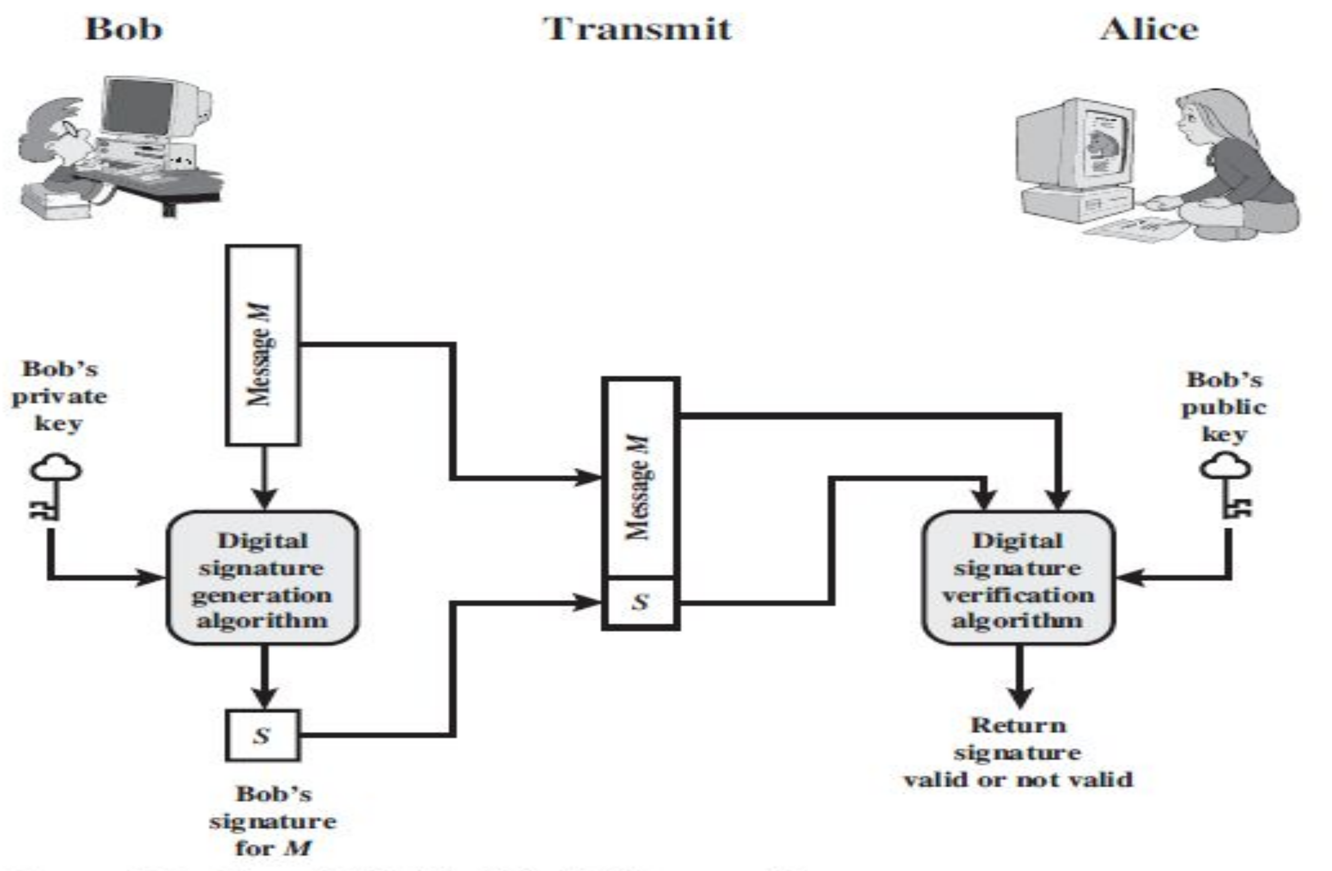

PUBLIC PLACE
Public key
MDswDQYJKoZIhvcNAQEBBQADKgAwJw
IgYjDXVpLBJSE7cZ83O3QZYgjQVZ5fTiwv
XqBezak1h5MCAwEAAQ==


I agree to pay Jill £100
SHA256
b21487415219839
256cfbcdc769ed76
32001d9a673c4de
df20017984242f42
hash value (digest)
Private key
MIGoAgEAAiBiMNdWksEIlTtxnzc7dBliCN
BVU9OLC9eoF7NqTWHkwIDAQABAiAJyIg
MNj8sde04WwbjO6Obs13UDYWDYDtmE
KoUvbNsUQjRAKnIN0x/0C+J0KmwFsUm
^43&sdfe"^(%/?:df
sau492853£54d££^
KX*$)(*&thvTSKJcS
](djsu5!!£ZzMgfL7$
encrypted hash value


I agree to pay Jill £100
SHA256
b21487415219839
256cfbcdc769ed76
32001d9a673c4de
df20017984242f42
Public key
MDswDQYJKoZIhvcNAQEBBQADKgAwJw
IgYjDXVpLBJSE7cZ83O3QZYgjQVZ5fTiwv
XqBezak1h5MCAwEAAQ==
^43&sdfe"^(%/?:df
sau492853£54d££^
KX*$)(*&thvTSKJcS
](djsu5!!£ZzMgfL7$
b21487415219839
256cfbcdc769ed76
32001d9a673c4de
df20017984242f42

Digital Signatures

- Generic model of the process of making and using digital signatures.

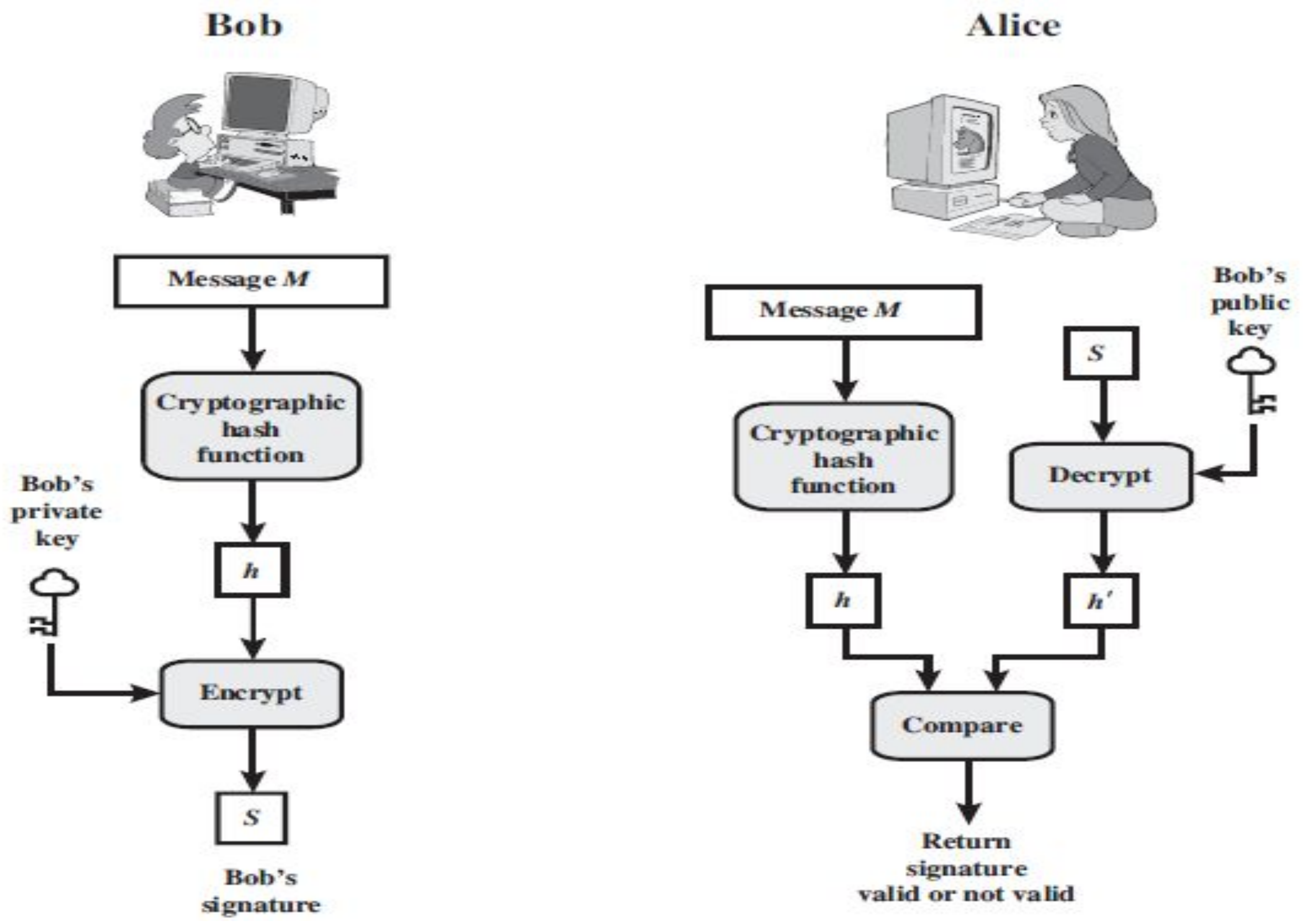


Bob
Transmit
Alice
Bob's private key
Message M
Digital signature generation algorithm
S
Bob's signature for M
Message M
S
Bob's public key
Digital signature verification algorithm
Return signature valid or not valid

Figure 13.1 Generic Model of Digital Signature Process

Digital Signatures

The essence of the digital signature mechanism is shown in Figure.



Digital Signature Requirements

- The signature must be a bit pattern that depends on the message being signed.
- The signature must use some information unique to the sender to prevent both forgery and denial.
- It must be relatively easy to produce the digital signature.
- It must be relatively easy to recognize and verify the digital signature.
- It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
- It must be practical to retain a copy of the digital signature in storage.

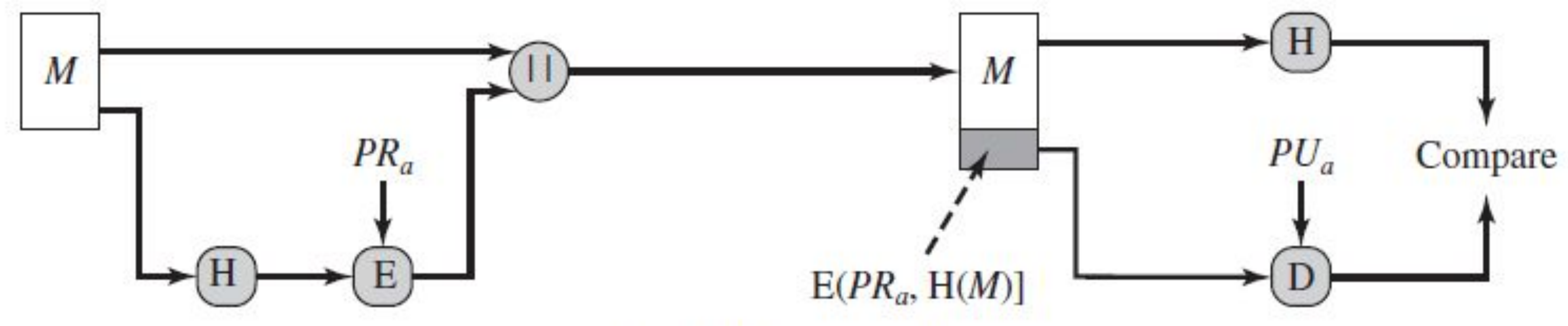
RAMAIAH
Institute of Technology

Digital Signature Algorithm

- The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as the Digital Signature Standard (DSS).
- The DSS makes use of the Secure Hash Algorithm (SHA) and presents a new digital signature technique, the Digital Signature Algorithm (DSA) in 1991.
- The latest version incorporates digital signature algorithms based on RSA and on elliptic curve cryptography.

Digital Signature Algorithm

Figure shows the DSS approach for generating digital signatures used with RSA.

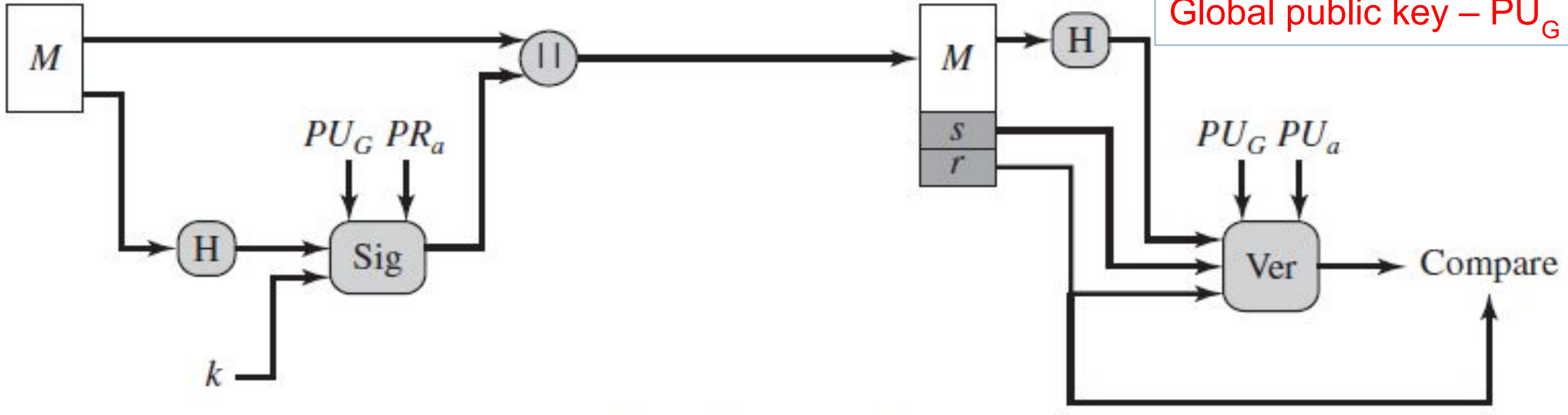


(a) RSA approach

Digital Signature Algorithm

Figure shows the DSS approach for generating digital signatures used with RSA.

Sender's private key – PR_a
Sender's public key – Pu_a
Random number - k
Sig – Signature function
Signature = {s,r}
Ver –Verification function
Global public key – PU_G



(b) DSS approach

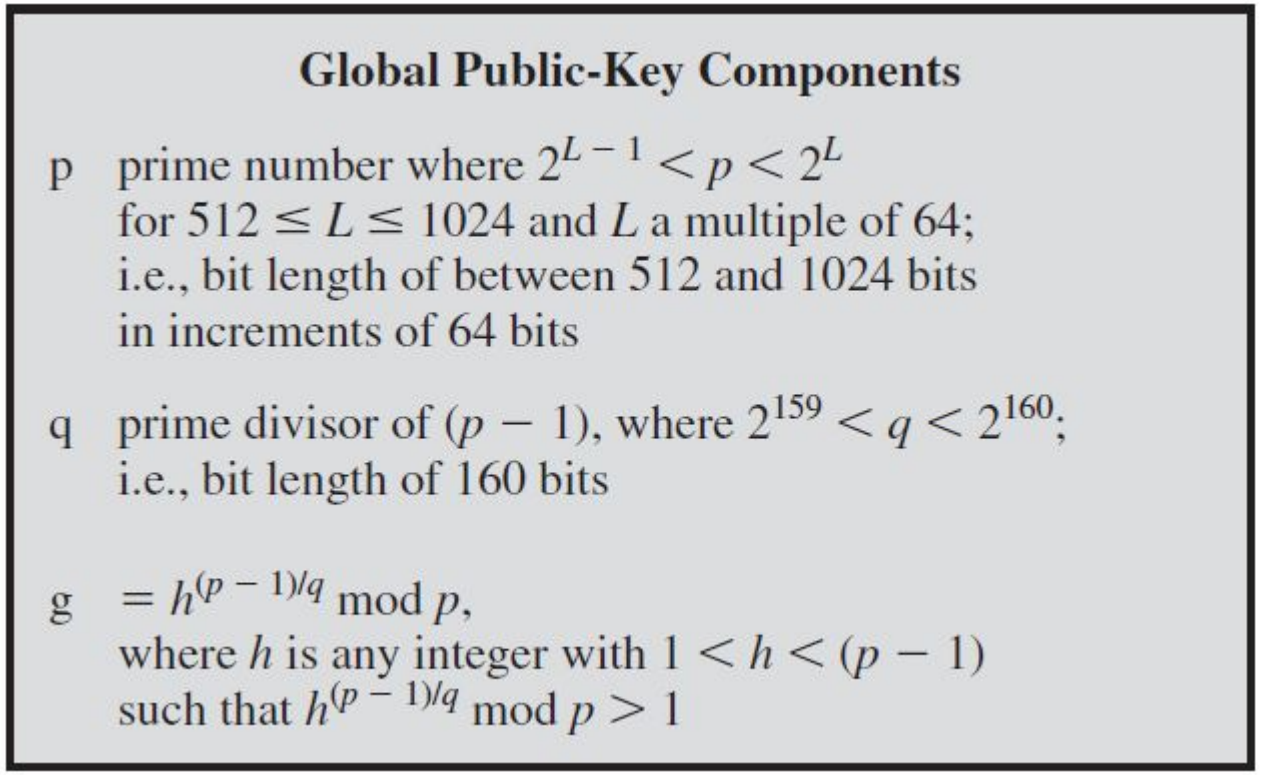
RAMAIAH
Institute of Technology

Digital Signature Algorithm

There are 3 parameters that are public and can be common to a group of users

1. Prime number p is selected with a length between 512 and 1024 bits.
2. A 160-bit prime number q is chosen
3. Finally, g is chosen

P = 11, q = (11-1)□ 10 = 5



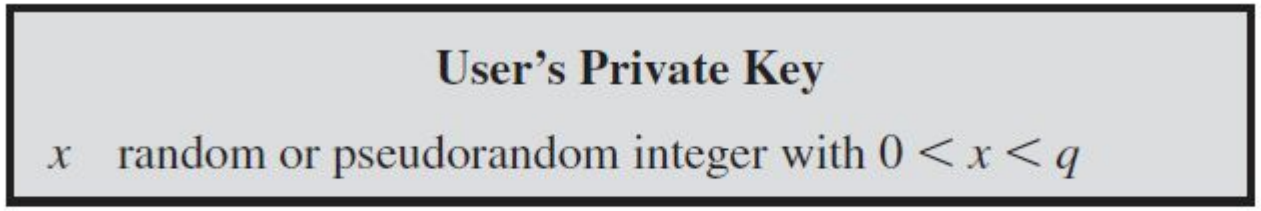
Global Public-Key Components

p	prime number where $2^{L-1} < p < 2^L$ for $512 \le L \le 1024$ and L a multiple of 64; i.e., bit length of between 512 and 1024 bits in increments of 64 bits
q	prime divisor of $(p - 1)$, where $2^{159} < q < 2^{160}$; i.e., bit length of 160 bits
g	$= h^{(p-1)/q} \bmod p$, where h is any integer with $1 < h < (p - 1)$ such that $h^{(p-1)/q} \bmod p > 1$

Digital Signature Algorithm

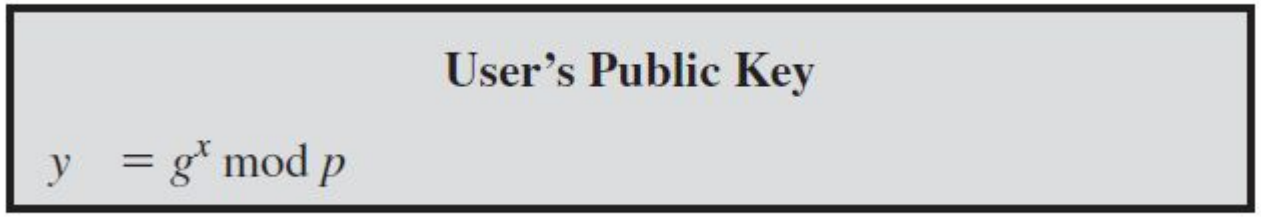
Each user selects a private key and generates a public key.

- The private key x must be a number from 0 to q and should be chosen randomly or pseudo randomly.
- The public key y is calculated from the private key.
- Additional integer k is generated randomly or pseudorandomly and be unique for each signing.



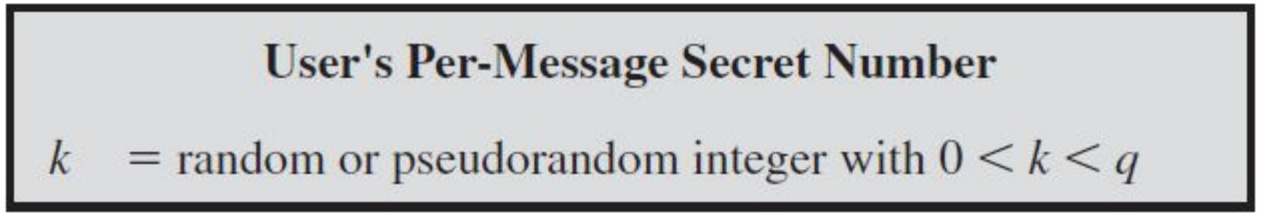
User's Private Key

x random or pseudorandom integer with $0 < x < q$



User's Public Key

$y = g^x \bmod p$



User's Per-Message Secret Number

k = random or pseudorandom integer with $0 < k < q$

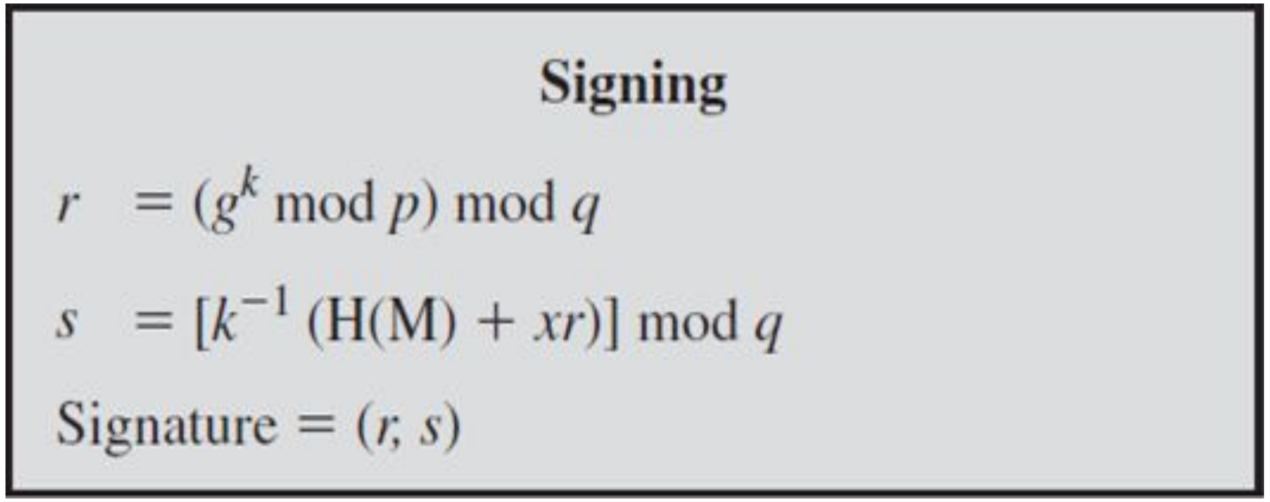
RAMAIAH
Institute of Technology

Digital Signature Algorithm

To create a signature

User calculates two quantities r and s, that are functions of the

- public key components(p,q,g)
- the user's private key x
- the hash code of the message H(M)
- additional integer k



Signing

$r = (g^k \bmod p) \bmod q$

$s = [k^{-1}(H(M) + xr)] \bmod q$

Signature = (r, s)

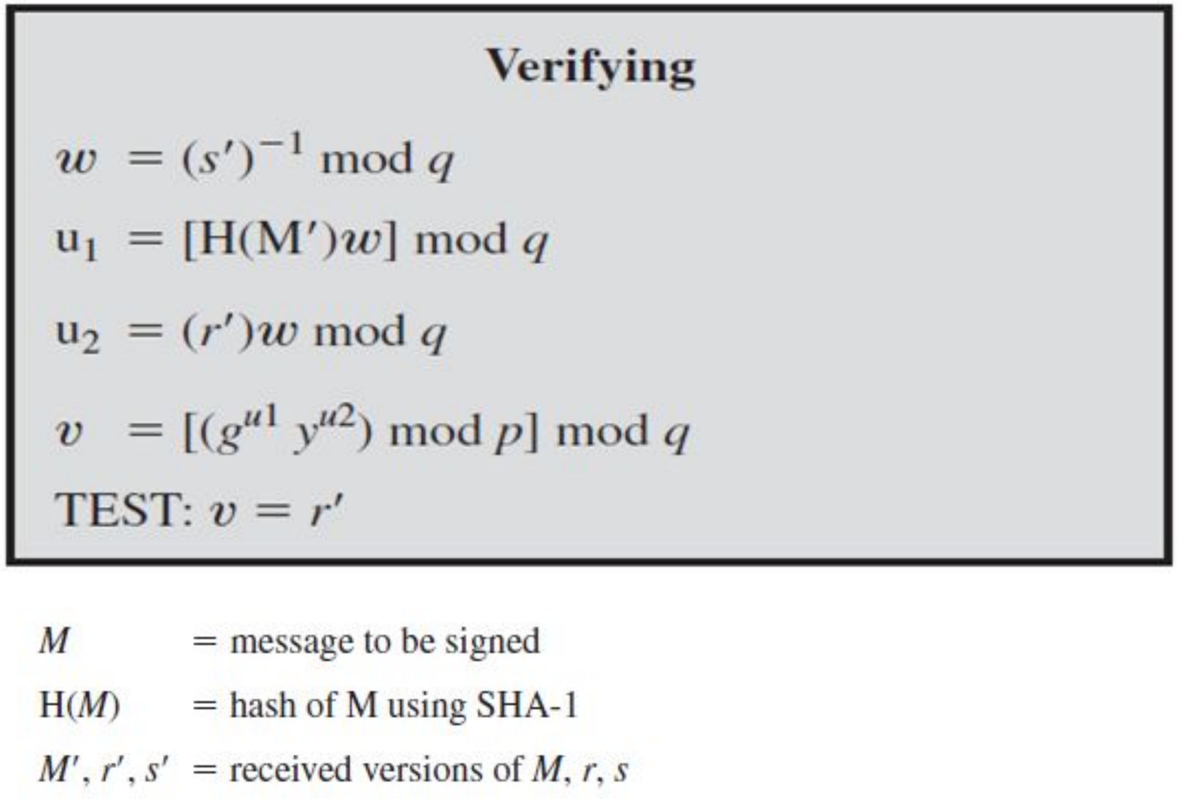
Digital Signature Algorithm

At the receiving end, verification is performed.

The receiver generates a quantity v that is a function of

- the public key components(p,q,g)
- the sender’s public key
- the hash code of the incoming message.

- If this quantity matches the r component of the signature, then the signature is validated.



Verifying

$$w = (s')^{-1} \bmod q$$

$$u_1 = [H(M')w] \bmod q$$

$$u_2 = (r')w \bmod q$$

$$v = [(g^{u1}\, y^{u2}) \bmod p] \bmod q$$

TEST: $v = r'$

M	= message to be signed
$H(M)$	= hash of M using SHA-1
M', r', s'	= received versions of M, r, s

Distribution of public keys

Several techniques have been proposed for the distribution of public keys.

- Public announcement
- Publicly available directory
- Public-key authority
- Public-key certificates

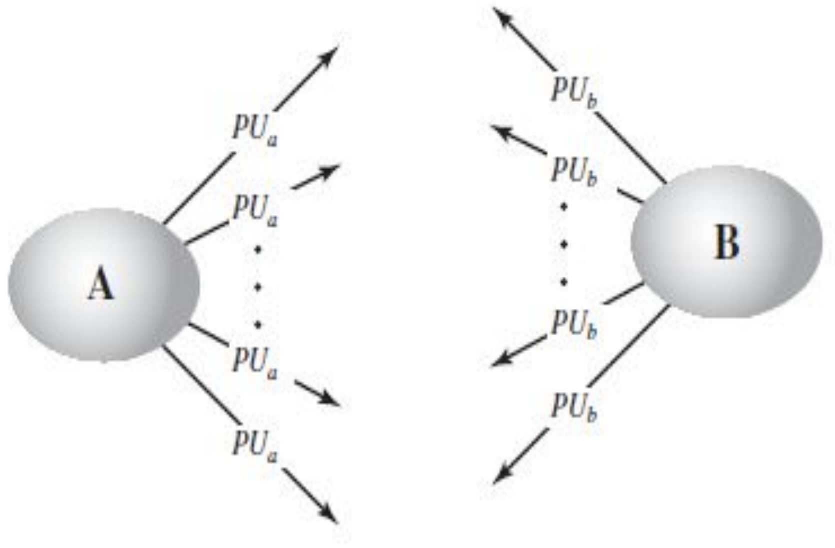
Distribution of public keys

Public announcement

Users distribute public keys to recipients or broadcast to community at large

Major weakness is forgery

- anyone can create a key claiming to be someone else and broadcast it
- until forgery is discovered can masquerade as claimed user



Uncontrolled Public-Key Distribution

Distribution of public keys

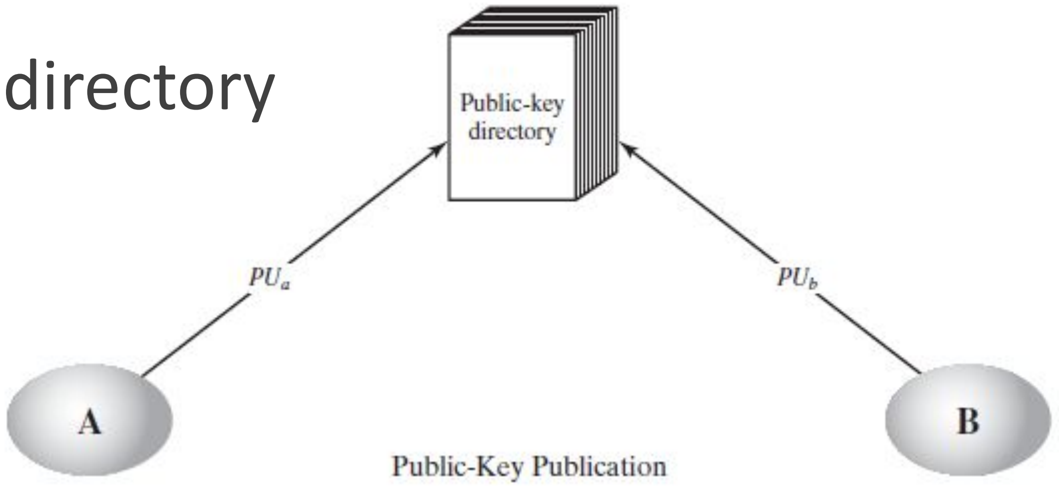
Publicly available directory

Can obtain greater security by registering keys with a public directory

directory must be trusted with properties:

- contains {name, public-key} entries
- Each participants register securely with directory
- participants can replace key at any time
- directory is periodically published
- directory can be accessed electronically

still vulnerable to tampering or forgery

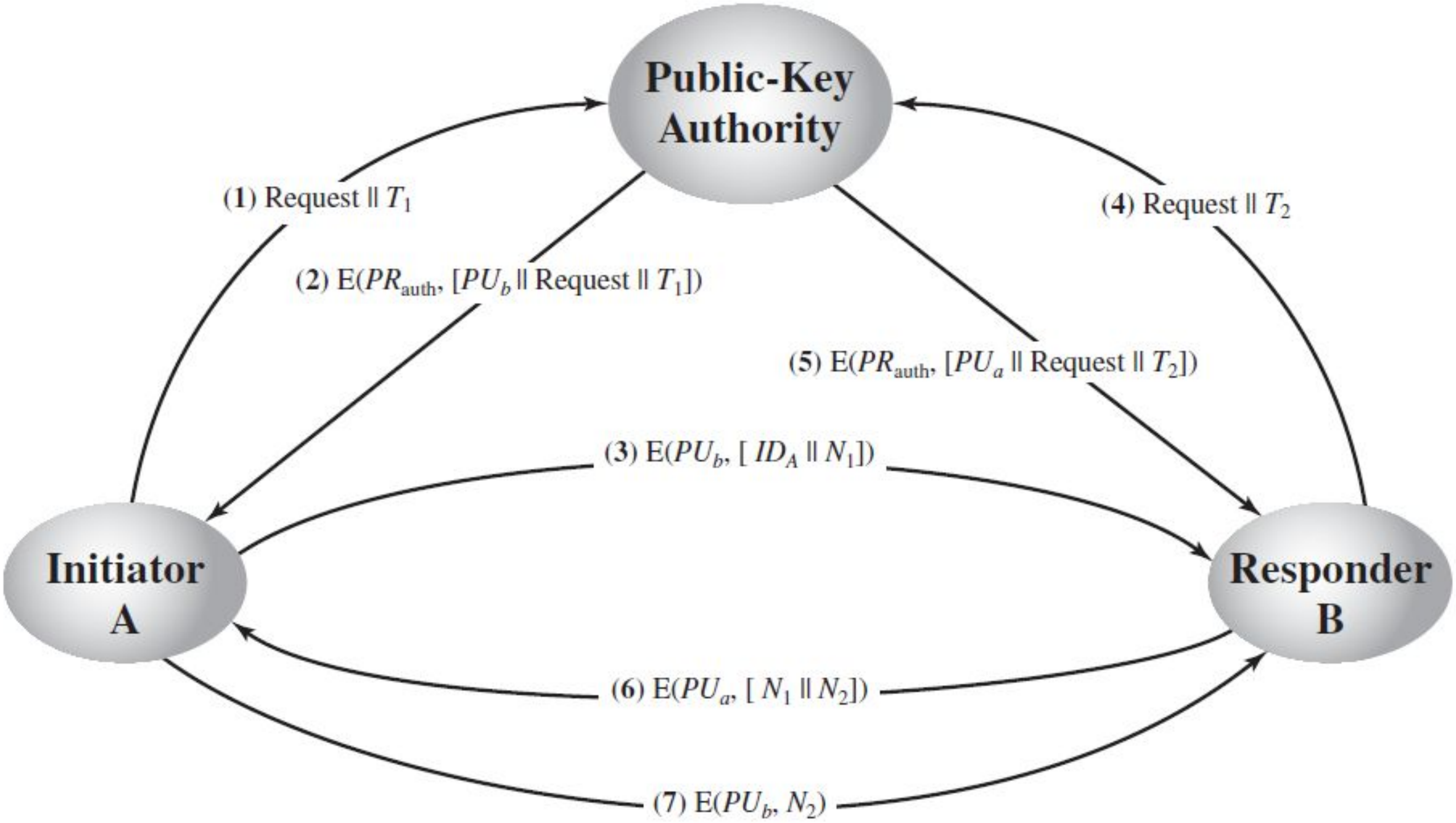


Public-Key Publication

Distribution of public keys

Public-key authority

- Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory.
- A central authority maintains a dynamic directory of public keys of all participants.
- In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key.
- A total of seven messages are required.
- However, the initial four messages need be used only infrequently because both A and B can save the other’s public key for future use—a technique known as caching.
- Periodically, a user should request fresh copies of the public keys of its correspondents to ensure currency.


Public-Key Authority
Initiator A
Responder B
(1) Request || T_1
(2) $E(PR_{auth}, [PU_b \| Request \| T_1])$
(3) $E(PU_b, [ID_A \| N_1])$
(4) Request || T_2
(5) $E(PR_{auth}, [PU_a \| Request \| T_2])$
(6) $E(PU_a, [N_1 \| N_2])$
(7) $E(PU_b, N_2)$

Public-Key Distribution Scenario

Distribution of public keys

Drawback with Public-key authority

The public-key authority could be somewhat of a bottleneck in the system, for a user must appeal to the authority for a public key for every other user that it wishes to contact.

As before, the directory of names and public keys maintained by the authority is vulnerable to tampering.

Distribution of public keys

Public-key certificates

- Use certificates to exchange keys without contacting a public-key authority.
- Typically, the third party is a certificate authority, such as a government agency or a financial institution, that is trusted by the user community.
- A user can present his or her public key to the authority in a secure manner and obtain a certificate.
- The user can then publish the certificate.

Distribution of public keys

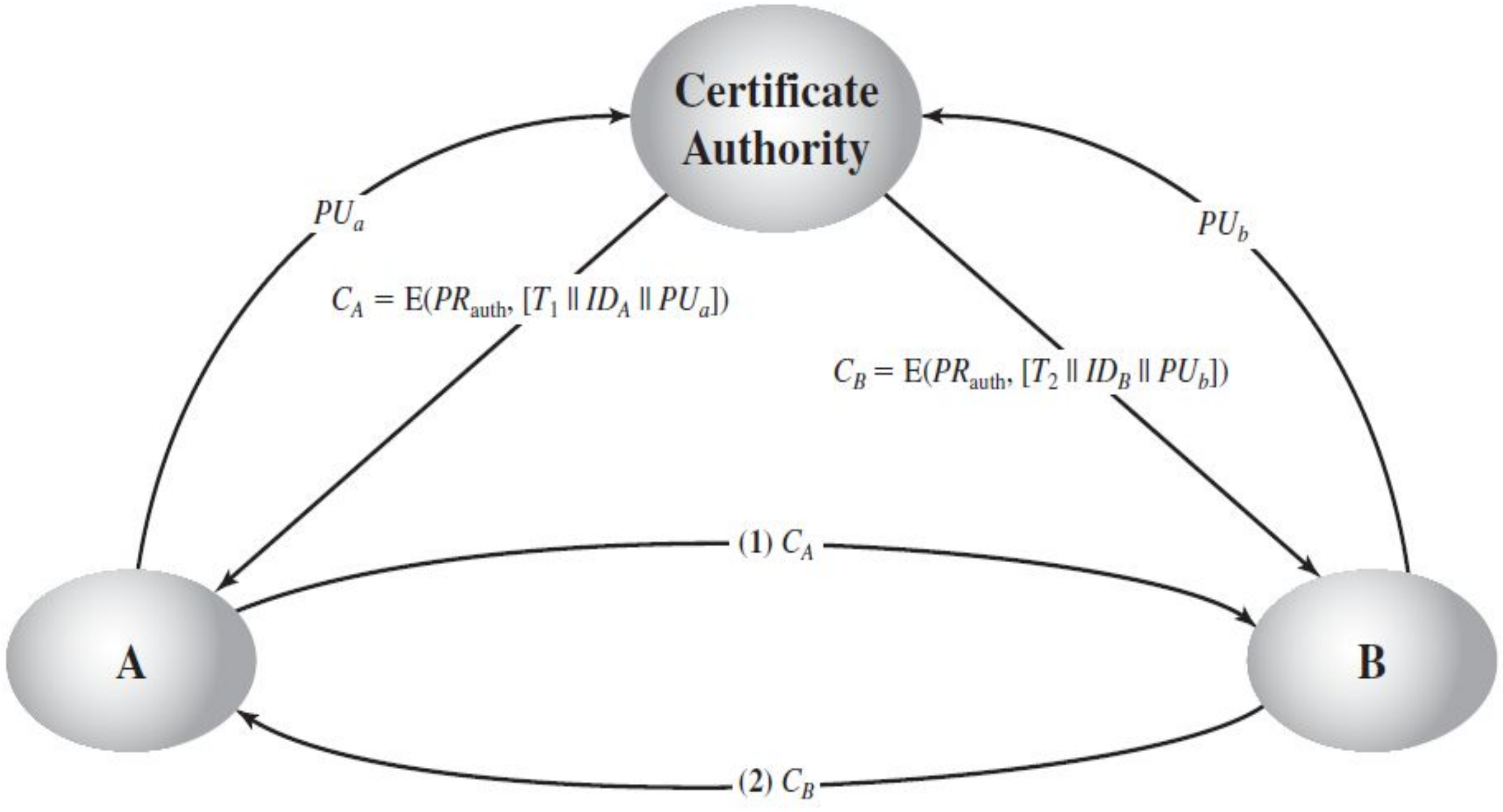
Public-key certificates

- Use certificates to exchange keys without contacting a public-key authority.
- Typically, the third party is a certificate authority, such as a government agency or a financial institution, that is trusted by the user community.
- A user can present his or her public key to the authority in a secure manner and obtain a certificate.
- The user can then publish the certificate.

Distribution of public keys

The CA signs the certificate with its private key.

If the corresponding public key is known to a user, then that user can verify that a certificate signed by the CA is valid.


Certificate Authority
PU_a
PU_b
$C_A = E(PR_{auth}, [T_1 \| ID_A \| PU_a])$
$C_B = E(PR_{auth}, [T_2 \| ID_B \| PU_b])$
(1) C_A
(2) C_B
A
B

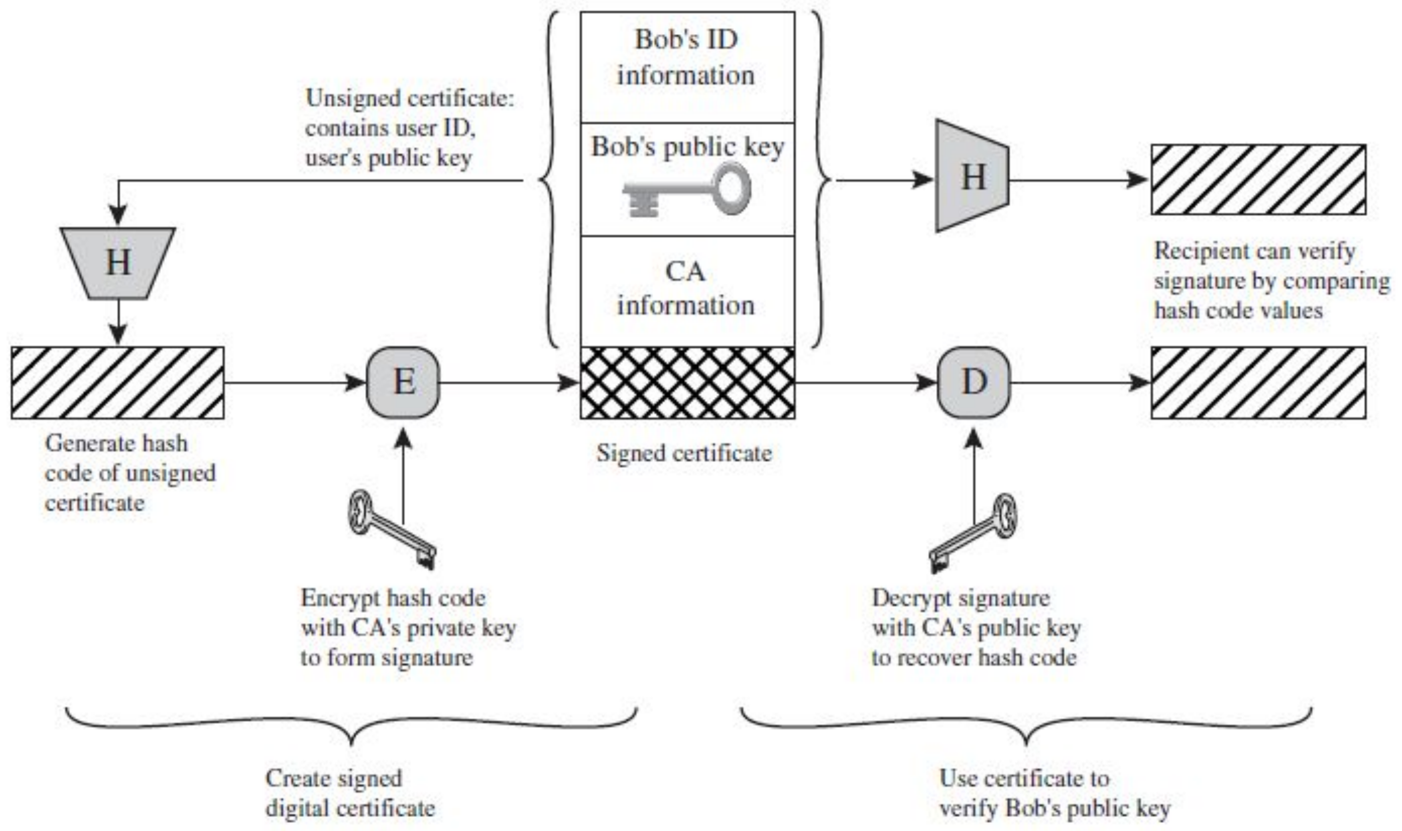
Exchange of Public-Key Certificates

X.509 certificates

- ITU-T recommendation X.509 is part of the X.500 series of recommendations that define a directory service.
- The heart of the X.509 scheme is the public-key certificate associated with each user.
- These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user.

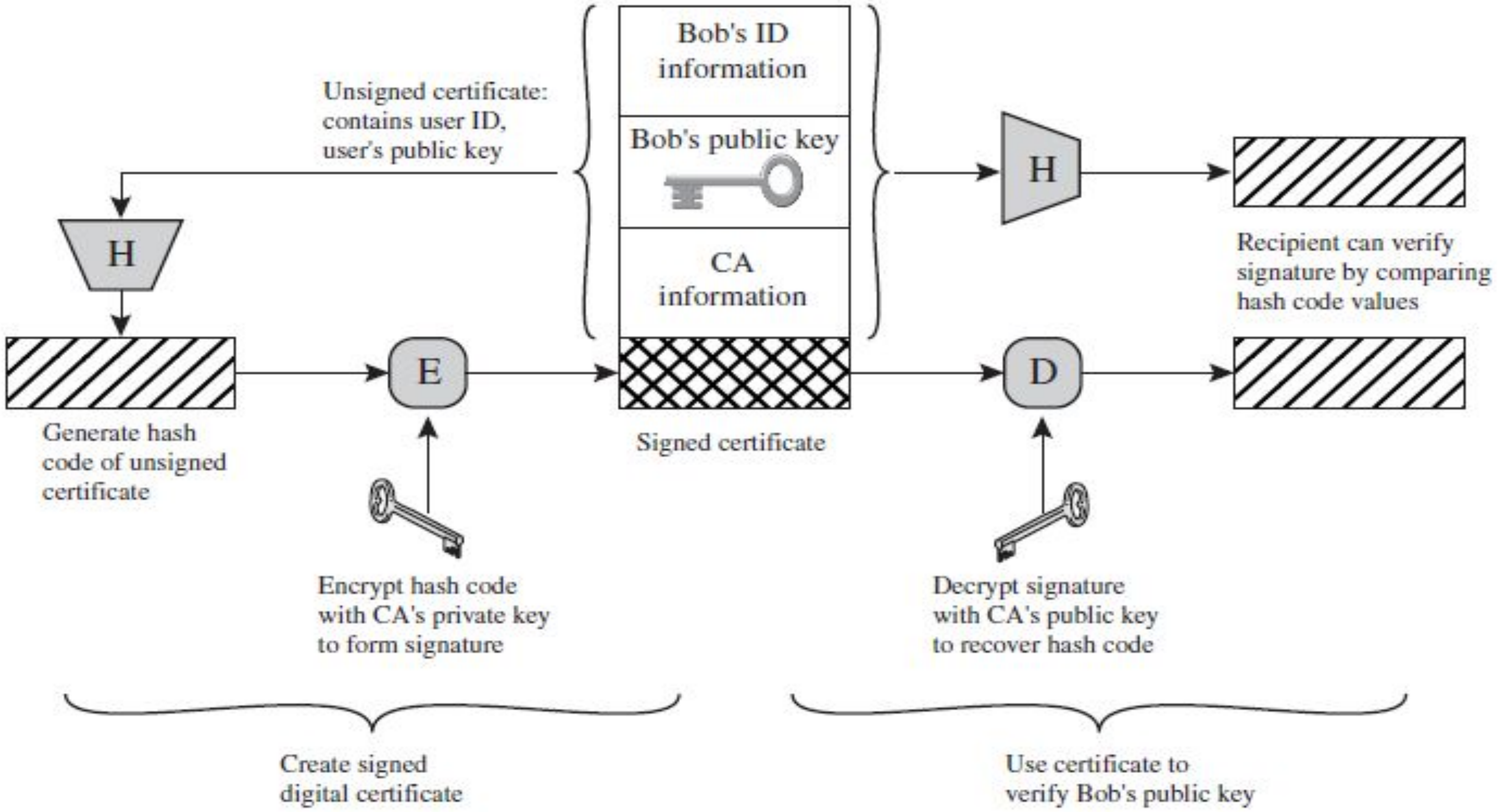
X.509 certificates

Figure illustrates the generation of a public-key certificate.



Bob's ID information
Bob's public key
CA information
Unsigned certificate: contains user ID, user's public key
H
H
E
D
Signed certificate
Generate hash code of unsigned certificate
Encrypt hash code with CA's private key to form signature
Decrypt signature with CA's public key to recover hash code
Recipient can verify signature by comparing hash code values
Create signed digital certificate
Use certificate to verify Bob's public key

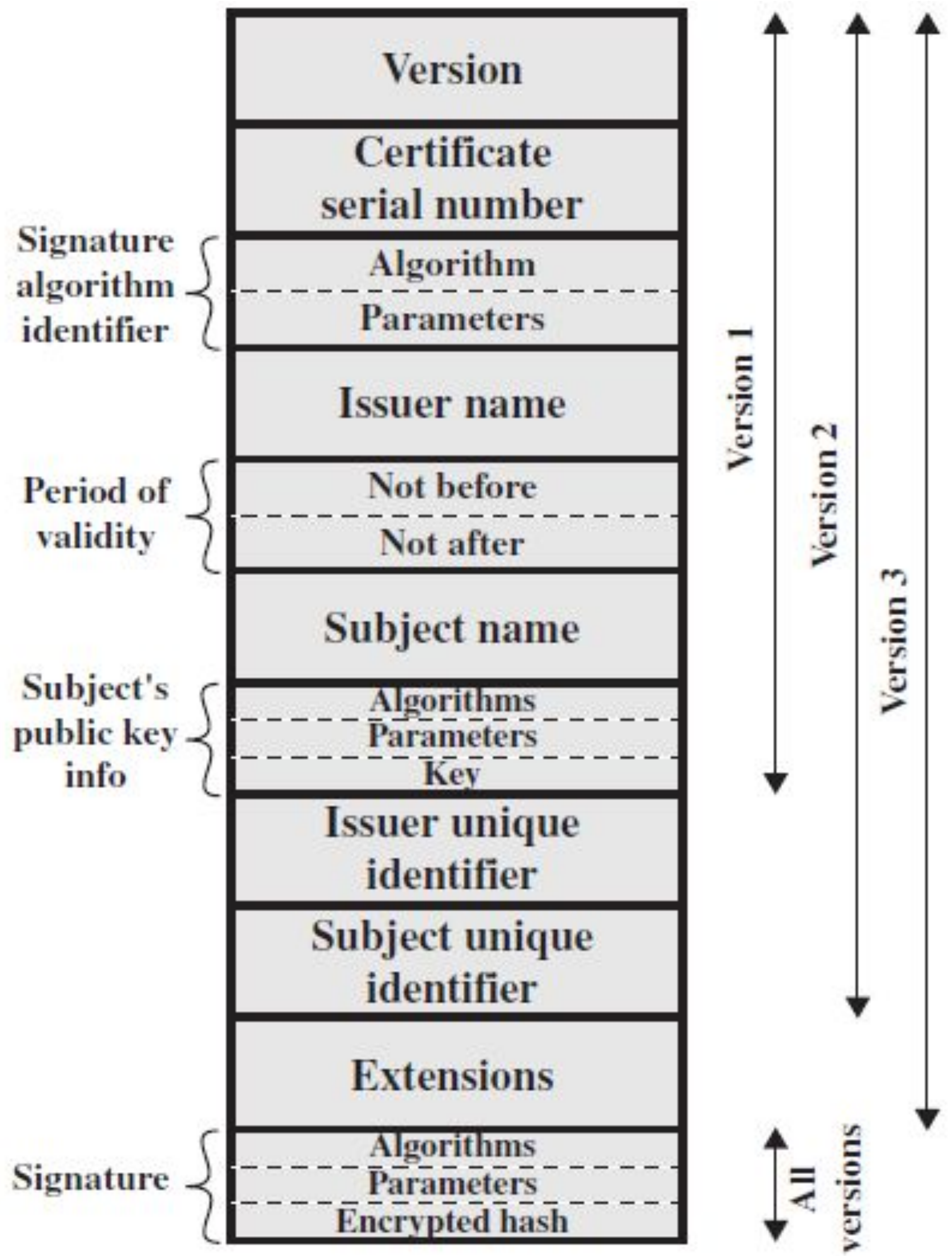
Public-Key Certificate Use


Unsigned certificate: contains user ID, user's public key
Bob's ID information
Bob's public key
CA information
H
H
Recipient can verify signature by comparing hash code values
E
D
Generate hash code of unsigned certificate
Signed certificate
Encrypt hash code with CA's private key to form signature
Decrypt signature with CA's public key to recover hash code
Create signed digital certificate
Use certificate to verify Bob's public key

Public-Key Certificate Use

X.509 certificates

Figure shows the general format of a certificate, which includes the following elements.



(a) X.509 certificate

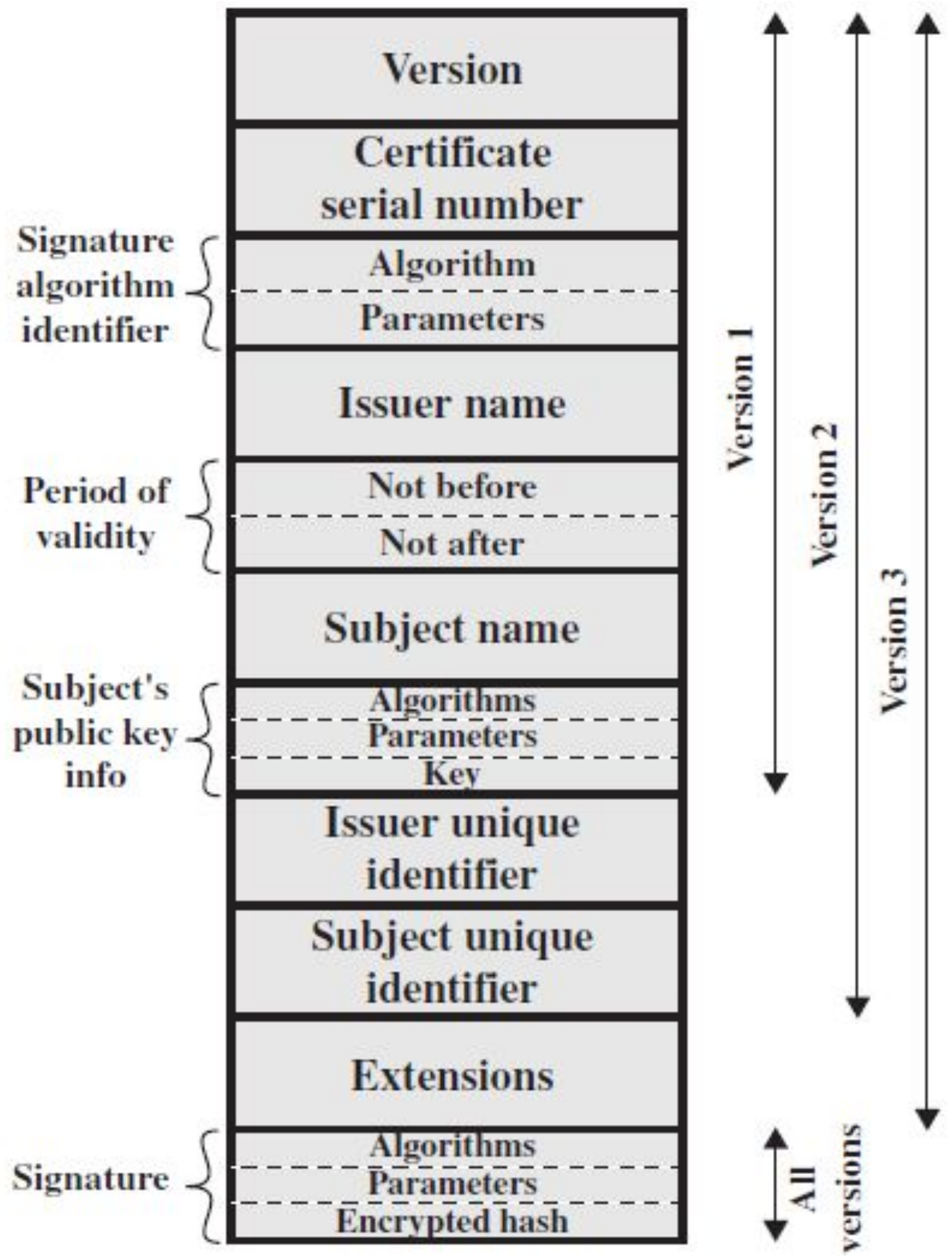
Version: Differentiates among successive versions

Serial number: An integer value unique within the issuing CA that is unambiguously associated with this certificate.

Signature algorithm identifier: The algorithm used to sign the certificate together with any associated parameters.

Issuer name: X.500 is the name of the CA that created and signed this certificate.

Period of validity: Consists of two dates: the first and last on which the certificate is valid.

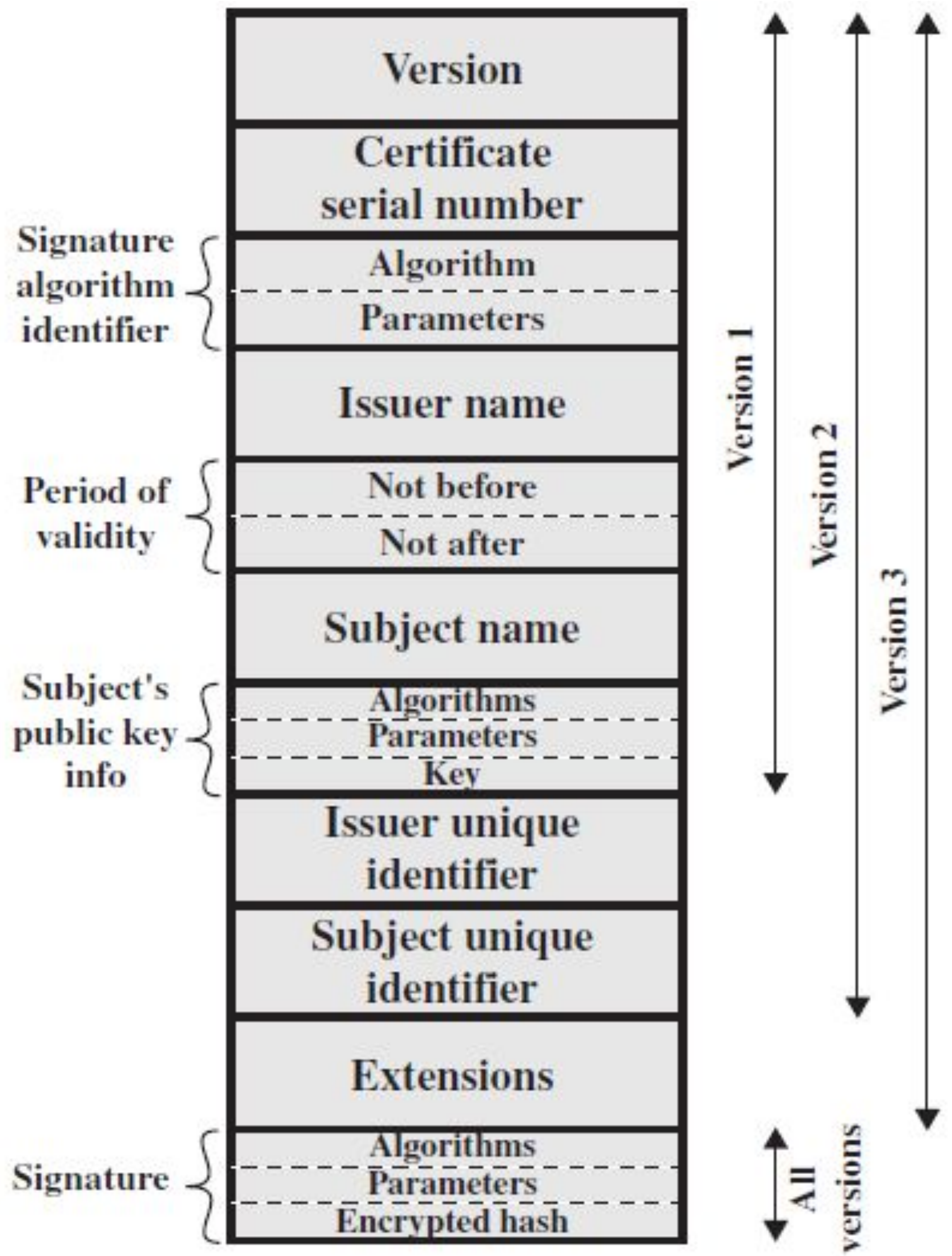


(a) X.509 certificate

Subject name: The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.

Subject's public-key information: The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.

Issuer unique identifier: An optional-bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.

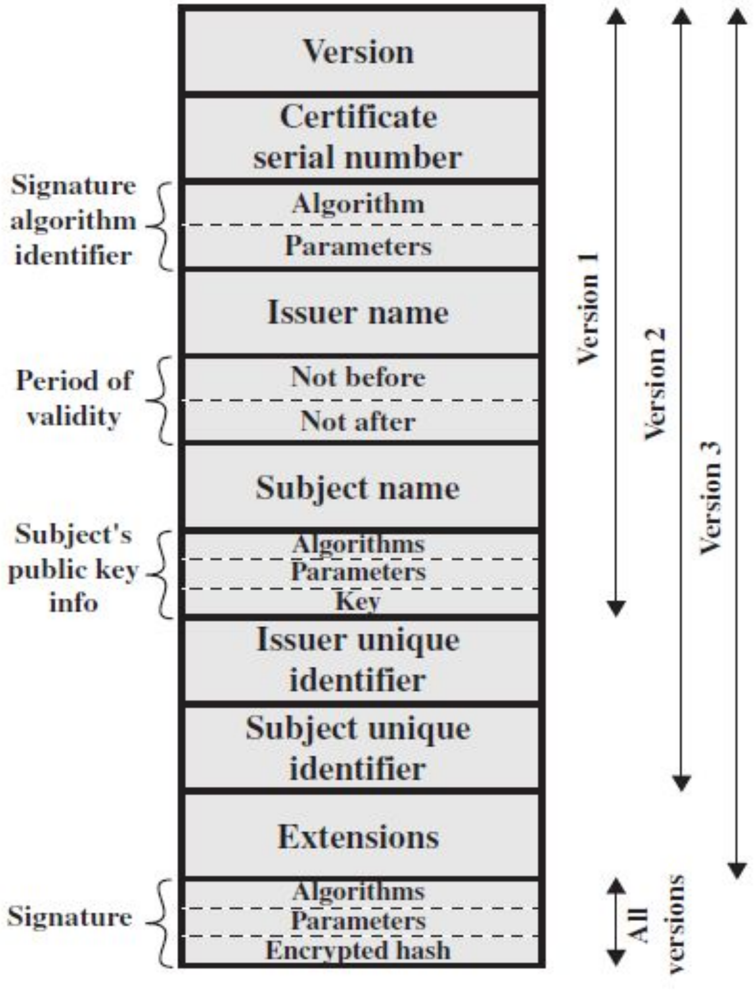


(a) X.509 certificate

Subject unique identifier: An optional-bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.

Extensions: A set of one or more extension fields. Extensions were added in version 3

Signature: Covers all of the other fields of the certificate; it contains the hash code of the other fields encrypted with the CA's private key. This field includes the signature algorithm identifier.



(a) X.509 certificate

The standard uses the following notation to define a certificate:

$$CA \ll A \gg = CA\,\{V, SN, AI, CA, UCA, A, UA, Ap, T^A\}$$

where

$Y \ll X \gg$	=	the certificate of user X issued by certification authority Y
$Y\,\{I\}$	=	the signing of I by Y. It consists of I with an encrypted hash code appended
V	=	version of the certificate
SN	=	serial number of the certificate
AI	=	identifier of the algorithm used to sign the certificate
CA	=	name of certificate authority
UCA	=	optional unique identifier of the CA
A	=	name of user A
UA	=	optional unique identifier of the user A
Ap	=	public key of user A
T^A	=	period of validity of the certificate

REVOCATION OF CERTIFICATES

- Each certificate includes a period of validity, much like a credit card.
- Typically, a new certificate is issued just before the expiration of the old one.
- In addition, it may be desirable on occasion to revoke a certificate before it expires, for one of the following reasons.
    1. The user's private key is assumed to be compromised.
    2. The user is no longer certified by this CA. Reasons for this include that the subject's name has changed, the certificate is superseded, or the certificate was not issued in conformance with the CA's policies.
    3. The CA's certificate is assumed to be compromised.

REVOCATION OF CERTIFICATES

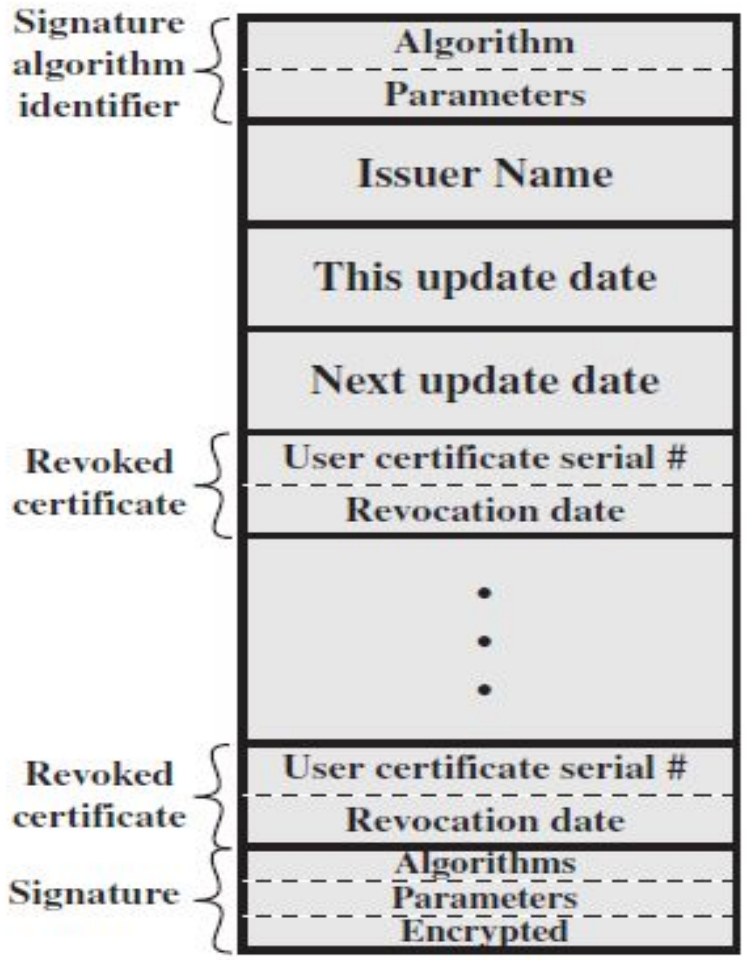
Each CA must maintain a list consisting of all revoked but not expired certificates issued by that CA, including both those issued to users and to other CAs.

These lists should also be posted on the directory.

REVOCATION OF CERTIFICATES

Each certificate revocation list (CRL) posted to the directory is signed by the issuer and includes

- the issuer’s name,
- the date the list was created,
- the date the next CRL is scheduled to be issued,
- an entry for each revoked certificate.



(b) Certificate revocation list

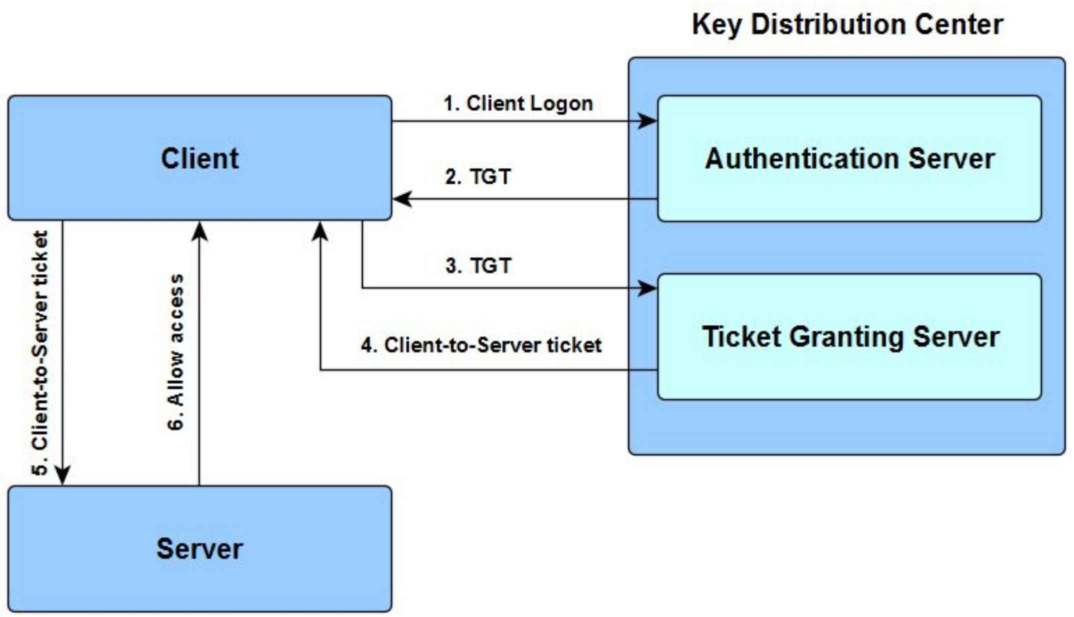
Kerberos

- Motivation
- Kerberos Version 4
- Kerberos Version 5

RAMAIAH
Institute of Technology

Kerberos

- Kerberos is an authentication service designed for use in a distributed environment.
- Kerberos provides a trusted third-party authentication service that enables clients and servers to establish authenticated communication.

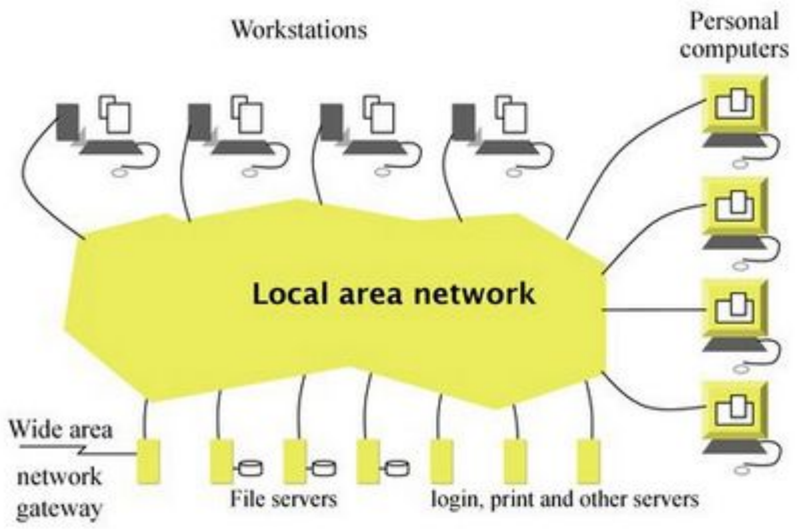


RAMAIAH Institute of Technology

Kerberos

The problem that Kerberos addresses is this:

- Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network.
- We would like for servers to be able to restrict access to authorized users and to be able to authenticate requests for service.
- In this environment, a workstation cannot be trusted to identify its users correctly to network services.



Kerberos

Following three threats exist:

1. A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
2. A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
3. A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

In any of these cases, an unauthorized user may be able to gain access to services and data that he or she is not authorized to access.

RAMAIAH
Institute of Technology

Kerberos

- Rather than building in elaborate authentication protocols at each server, Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users.
- Kerberos relies exclusively on symmetric encryption.
- Two versions of Kerberos are in common use.
    - Version 4 implementations still exist.
    - Version 5 corrects some of the security deficiencies of version 4 and has been issued as a proposed Internet Standard (RFC 4120).

Kerberos

Motivation for the Kerberos approach

- If a set of users is provided with dedicated personal computers that have no network connections, then a user’s resources and files can be protected by physically securing each personal computer.
- When these users instead are served by a centralized timesharing system, the time-sharing operating system must provide the security.
- The operating system can enforce access-control policies based on user identity and use the logon procedure to identify users.
- Today, neither of these scenarios is typical.

Kerberos

Motivation for the Kerberos approach

More common is a distributed architecture consisting of dedicated user workstations (clients) and distributed or centralized servers.

3 approaches to security can be envisioned.

1. Rely on each individual client workstation to assure the identity of its user or users and rely on each server to enforce a security policy based on user identification (ID).
2. Require that client systems authenticate themselves to servers, but trust the client system concerning the identity of its user.
3. Require the user to prove his or her identity for each service invoked. Also require that servers prove their identity to clients.

RAMAIAH
Institute of Technology

Kerberos

Motivation for the Kerberos approach

Kerberos supports this third approach.

Kerberos assumes a distributed client/server architecture and employs one or more Kerberos servers to provide an authentication service.

Kerberos

Motivation for the Kerberos approach

Kerberos listed the following requirements.

1. **Secure**: Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
2. **Reliable:** Kerberos should be highly reliable and should employ a distributed server architecture with one system able to back up another.
3. **Transparent**: Ideally, the user should not be aware that authentication is taking place beyond the requirement to enter a password.
4. **Scalable:** The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.

Kerberos

Motivation for the Kerberos approach

- To support these requirements, the overall scheme of Kerberos is that of a trusted third-party authentication service that uses a protocol based Needham and Schroeder.
- It is trusted in the sense that clients and servers trust Kerberos to mediate their mutual authentication.

Kerberos

Kerberos version 4

- Version 4 of Kerberos makes use of DES to provide the authentication service.
- Therefore, we adopt a strategy used by Bill Bryant of Project Athena and build up to the full protocol by looking first at several hypothetical dialogues.
- Each successive dialogue adds additional complexity to counter security vulnerabilities revealed in the preceding dialogue.

Kerberos

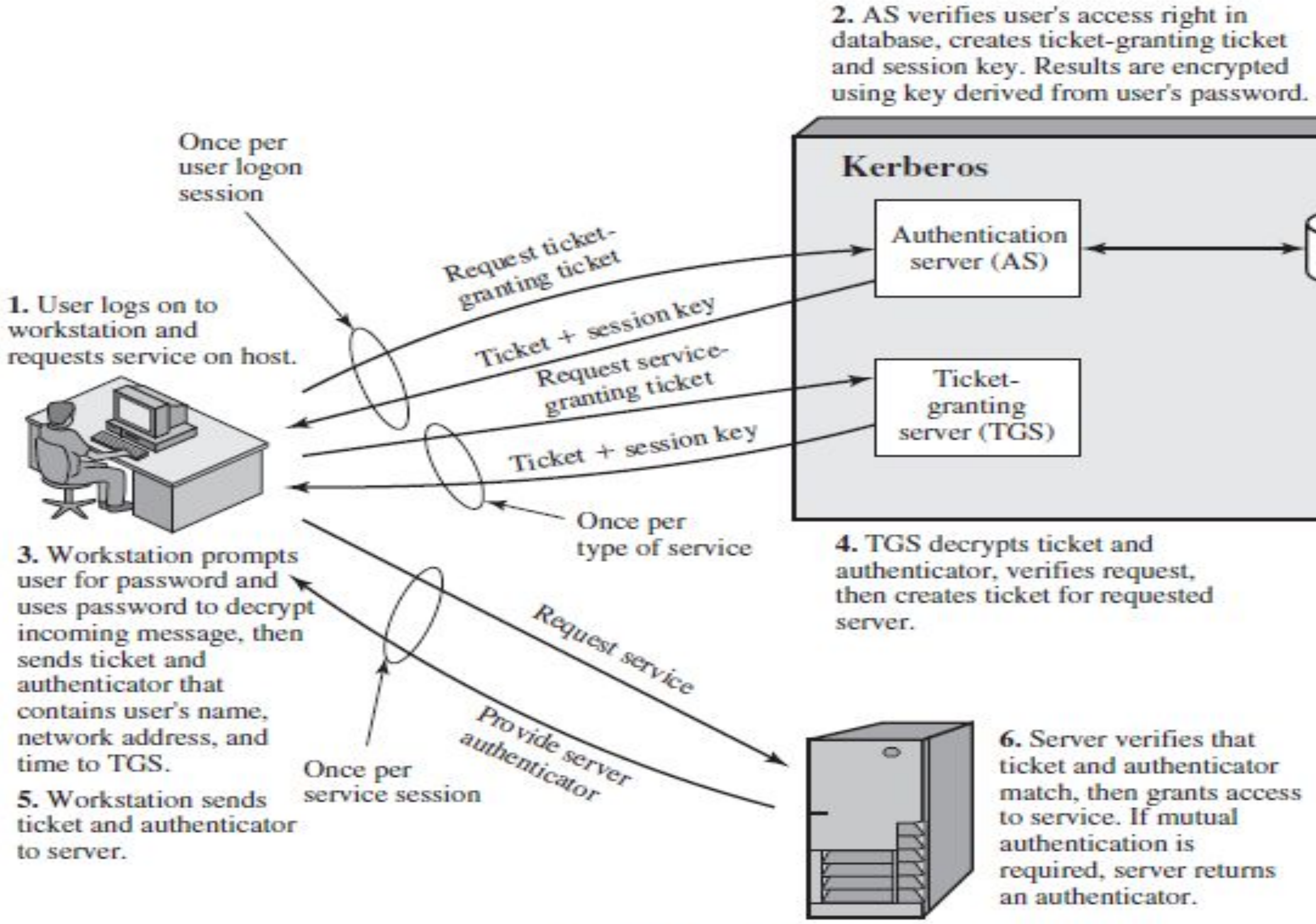
Kerberos version 4 - A SIMPLE AUTHENTICATION DIALOGUE

- In an unprotected network environment, any client can apply to any server for service.
- The obvious security risk is that of impersonation. An opponent can pretend to be another client and obtain unauthorized privileges on server machines.
- To counter this threat, servers must be able to confirm the identities of clients who request service.
- Each server can be required to undertake this task for each client/server interaction, but in an open environment, this places a substantial burden on each server.

Kerberos

Kerberos version 4 - A SIMPLE AUTHENTICATION DIALOGUE

- An alternative is to use an authentication server (AS) that knows the passwords of all users and stores these in a centralized database.
- In addition, the AS shares a unique secret key with each server.
- These keys have been distributed physically or in some other secure manner.


1. User logs on to workstation and requests service on host.

2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.

3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.

4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.

5. Workstation sends ticket and authenticator to server.

6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Kerberos
Authentication server (AS)
Ticket-granting server (TGS)
Once per user logon session
Request ticket-granting ticket
Ticket + session key
Request service-granting ticket
Ticket + session key
Once per type of service
Request service
Provide server authenticator
Once per service session

Overview of Kerberos

RAMAIAH
Institute of Technology

Kerberos

Kerberos version 4 - A SIMPLE AUTHENTICATION DIALOGUE

Consider the following hypothetical dialogue:

$$\begin{aligned} &\textbf{(1)}\ \ \mathrm{C} \rightarrow \mathrm{AS}: && ID_C \| P_C \| ID_V \\ &\textbf{(2)}\ \ \mathrm{AS} \rightarrow \mathrm{C}: && Ticket \\ &\textbf{(3)}\ \ \mathrm{C} \rightarrow \mathrm{V}: && ID_C \| Ticket \end{aligned}$$

$$Ticket = E(K_v, [ID_C \| AD_C \| ID_V])$$

where

C	= client	ID_V	= identifier of V
AS	= authentication server	P_C	= password of user on C
V	= server	AD_C	= network address of C
ID_C	= identifier of user on C	K_v	= secret encryption key shared by AS and V

RAMAIAH Institute of Technology

Kerberos

Kerberos version 4 - A SIMPLE AUTHENTICATION DIALOGUE

User logs on to a workstation and requests access to server V.

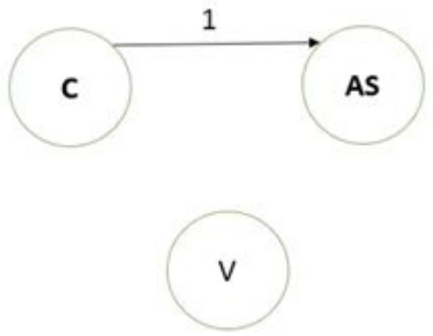
1. The client module C in the user's workstation requests the user's password and then sends a message to the AS that includes the user's ID, the server's ID, and the user's password.

$$\textbf{(1)}\ \mathrm{C} \rightarrow \mathrm{AS}: \quad ID_C \| P_C \| ID_V$$

$$\textbf{(2)}\ \mathrm{AS} \rightarrow \mathrm{C}: \quad Ticket$$

$$\textbf{(3)}\ \mathrm{C} \rightarrow \mathrm{V}: \quad ID_C \| Ticket$$

$$Ticket = E(K_v, [ID_C \| AD_C \| ID_V])$$



Kerberos

Kerberos version 4 - A SIMPLE AUTHENTICATION DIALOGUE

2. The AS checks its database to see if the user has supplied the proper password for this user ID and whether this user is permitted access to server V.

If both tests are passed, the AS accepts the user as authentic and must now convince the server that this user is authentic.

To do so, the AS creates **a ticket** that contains the user's ID and network address and the server's ID. This ticket is encrypted using the secret key shared by the AS and this server. This ticket is then sent back to C.

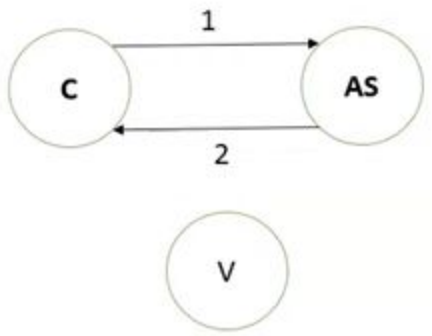
Because the ticket is encrypted, it cannot be altered by C or by an opponent.

(1) $C \rightarrow AS: \quad ID_C \| P_C \| ID_V$

(2) $AS \rightarrow C: \quad Ticket$

(3) $C \rightarrow V: \quad ID_C \| Ticket$

$$Ticket = E(K_v, [ID_C \| AD_C \| ID_V])$$



Kerberos

Kerberos version 4 - A SIMPLE AUTHENTICATION DIALOGUE

3. With this ticket, C can now apply to V for service.

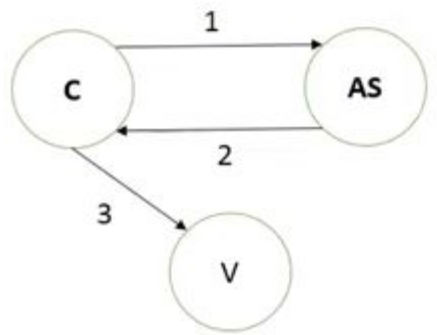
C sends a message to V containing C’s ID and the ticket.

V decrypts the ticket and verifies that the user ID in the ticket is the same as the unencrypted user ID in the message.

If these two match, the server considers the user authenticated and grants the requested service.

$$
\begin{aligned}
&\textbf{(1)}\ \ \mathrm{C} \rightarrow \mathrm{AS}: && ID_C \| P_C \| ID_V \\
&\textbf{(2)}\ \ \mathrm{AS} \rightarrow \mathrm{C}: && Ticket \\
&\textbf{(3)}\ \ \mathrm{C} \rightarrow \mathrm{V}: && ID_C \| Ticket
\end{aligned}
$$

$$Ticket = E(K_v, [ID_C \| AD_C \| ID_V])$$



Kerberos

Kerberos version 4 - A SIMPLE AUTHENTICATION DIALOGUE

Problem 1:

- To minimize the number of times that a user has to enter a password.
- Suppose each ticket can be used only once.
- If user C logs on to a workstation in the morning and wishes to check his or her mail at a mail server, C must supply a password to get a ticket for the mail server. If C wishes to check the mail several times during the day, each attempt requires reentering the password.

RAMAIAH Institute of Technology

Kerberos

Kerberos version 4 - A SIMPLE AUTHENTICATION DIALOGUE

Problem 2:

- The second problem is plaintext transmission of the password
- An eavesdropper could capture the password and use any service accessible to the victim.

$$\begin{array}{lll} \textbf{(1)} & \mathrm{C} \rightarrow \mathrm{AS}: & ID_C \| P_C \| ID_V \\ \textbf{(2)} & \mathrm{AS} \rightarrow \mathrm{C}: & Ticket \\ \textbf{(3)} & \mathrm{C} \rightarrow \mathrm{V}: & ID_C \| Ticket \end{array}$$

$$Ticket = E(K_v, [ID_C \| AD_C \| ID_V])$$

Kerberos

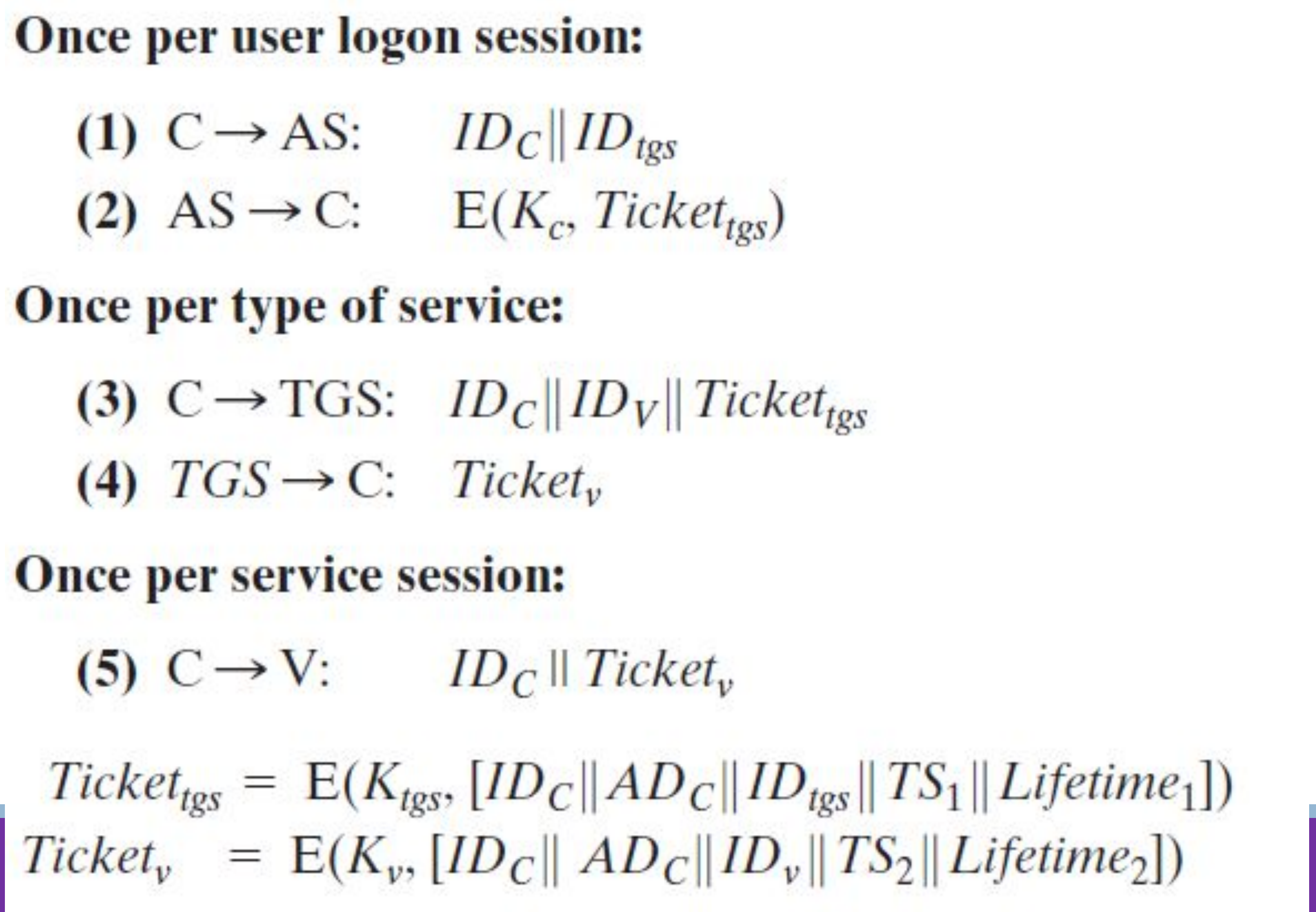
Kerberos version 4 - A MORE SIMPLE AUTHENTICATION DIALOGUE

- We introduce a scheme for avoiding plaintext passwords and a new server, known as the ticket-granting server (TGS).
- TGS issues tickets to users who have been authenticated to AS.
- The new (but still hypothetical) scenario is as follows.

RAMAIAH
Institute of Technology

Kerberos

Kerberos version 4 - A MORE SIMPLE AUTHENTICATION DIALOGUE



Once per user logon session:

(1) $C \rightarrow AS$: $ID_C \| ID_{tgs}$

(2) $AS \rightarrow C$: $E(K_c, Ticket_{tgs})$

Once per type of service:

(3) $C \rightarrow TGS$: $ID_C \| ID_V \| Ticket_{tgs}$

(4) $TGS \rightarrow C$: $Ticket_v$

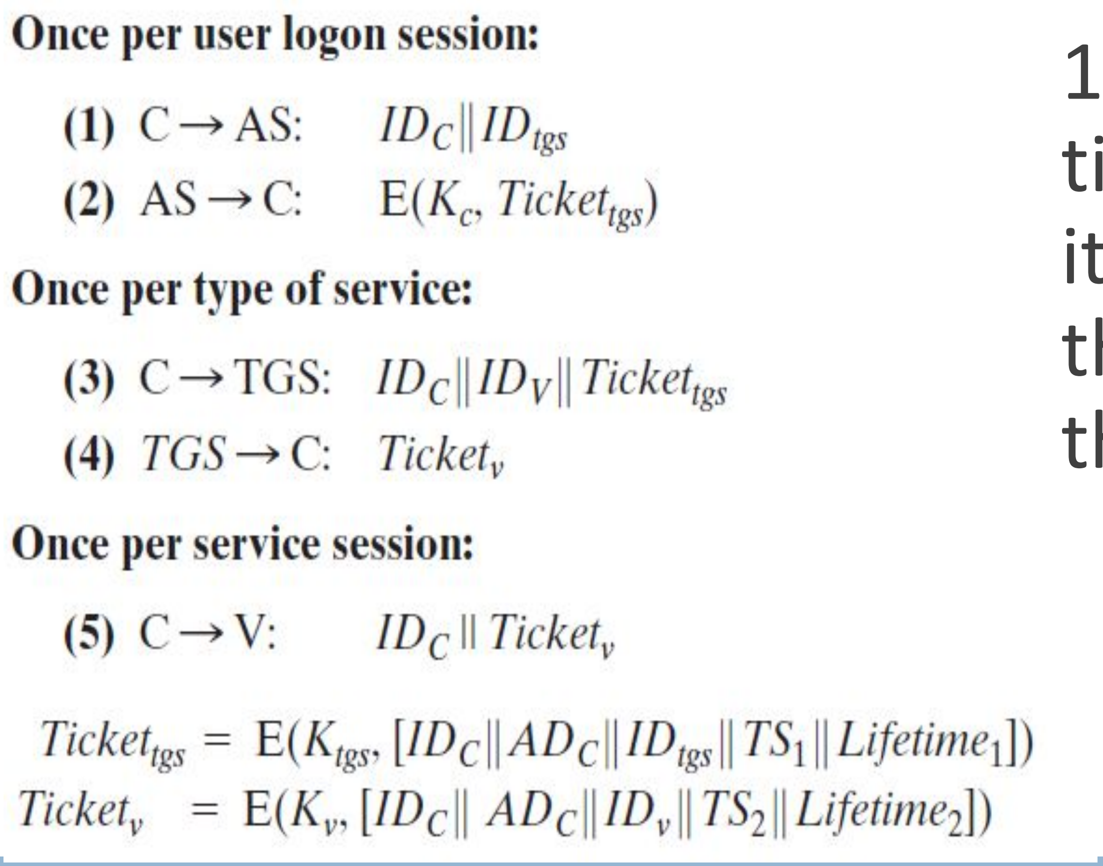
Once per service session:

(5) $C \rightarrow V$: $ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$
$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

Kerberos

Kerberos version 4 - A MORE SIMPLE AUTHENTICATION DIALOGUE



Once per user logon session:

(1) $C \rightarrow AS$: $ID_C \| ID_{tgs}$

(2) $AS \rightarrow C$: $E(K_c, Ticket_{tgs})$

Once per type of service:

(3) $C \rightarrow TGS$: $ID_C \| ID_V \| Ticket_{tgs}$

(4) $TGS \rightarrow C$: $Ticket_v$

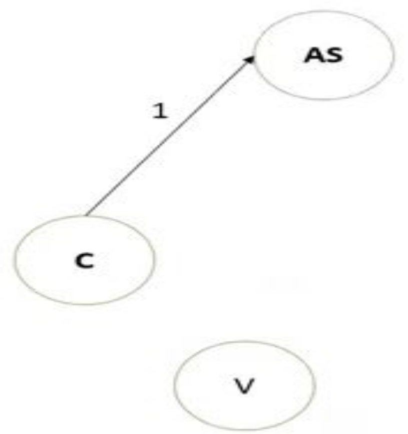
Once per service session:

(5) $C \rightarrow V$: $ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$

$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

1. The client requests a ticket-granting ticket on behalf of the user by sending its user’s ID to the AS, together with the TGS ID, indicating a request to use the TGS service.



Kerberos

Kerberos version 4 - A MORE SIMPLE AUTHENTICATION DIALOGUE

Once per user logon session:

(1) $C \rightarrow AS$: $ID_C \| ID_{tgs}$

(2) $AS \rightarrow C$: $E(K_c, Ticket_{tgs})$

Once per type of service:

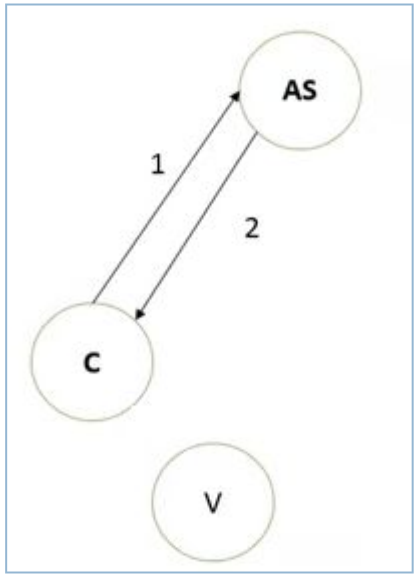
(3) $C \rightarrow TGS$: $ID_C \| ID_V \| Ticket_{tgs}$

(4) $TGS \rightarrow C$: $Ticket_v$

Once per service session:

(5) $C \rightarrow V$: $ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$
$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$



2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password (K_c), which is already stored at the AS.

When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message.

If the correct password is supplied, the ticket is successfully recovered.

Kerberos

Kerberos version 4 - A MORE SIMPLE AUTHENTICATION DIALOGUE

Once per user logon session:

(1) $C \rightarrow AS$: $ID_C \| ID_{tgs}$

(2) $AS \rightarrow C$: $E(K_c, Ticket_{tgs})$

Once per type of service:

(3) $C \rightarrow TGS$: $ID_C \| ID_V \| Ticket_{tgs}$

(4) $TGS \rightarrow C$: $Ticket_v$

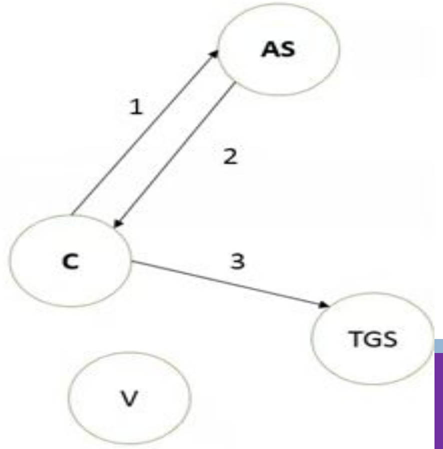
Once per service session:

(5) $C \rightarrow V$: $ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$
$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

3. The client requests a service-granting ticket on behalf of the user.

For this purpose, the client transmits a message to the TGS containing the user’s ID, the ID of the desired service, and the ticket-granting ticket.



RAMAIAH Institute of Technology

Kerberos

Kerberos version 4 - A MORE SIMPLE AUTHENTICATION DIALOGUE

Once per user logon session:

(1) $C \rightarrow AS$: $ID_C \| ID_{tgs}$

(2) $AS \rightarrow C$: $E(K_c, Ticket_{tgs})$

Once per type of service:

(3) $C \rightarrow TGS$: $ID_C \| ID_V \| Ticket_{tg}$

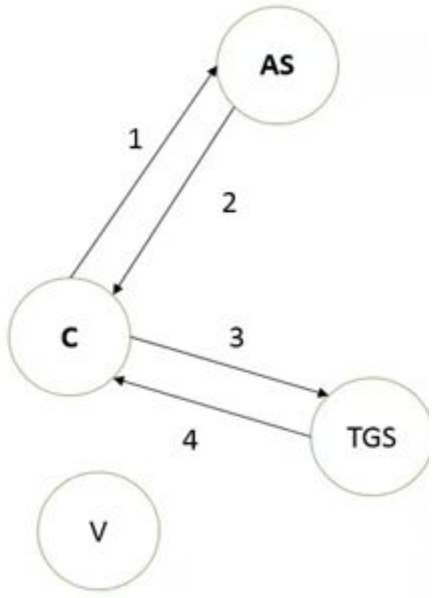
(4) $TGS \rightarrow C$: $Ticket_v$

Once per service session:

(5) $C \rightarrow V$: $ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$

$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$



4. The TGS decrypts the incoming ticket using a key shared only by the AS and the TGS (K_{tgs}) and verifies the success of the decryption by the presence of its ID.

It checks to make sure that the lifetime has not expired.

Then it compares the user ID and network address with the incoming information to authenticate the user.

If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.

Kerberos

Kerberos version 4 - A MORE SIMPLE AUTHENTICATION DIALOGUE

Once per user logon session:

(1) $C \rightarrow AS$: $ID_C \| ID_{tgs}$

(2) $AS \rightarrow C$: $E(K_c, Ticket_{tgs})$

Once per type of service:

(3) $C \rightarrow TGS$: $ID_C \| ID_V \| Ticket_{tgs}$

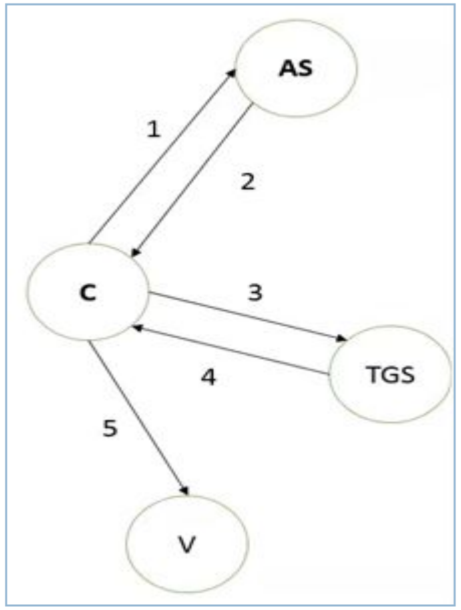
(4) $TGS \rightarrow C$: $Ticket_v$

Once per service session:

(5) $C \rightarrow V$: $ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$

$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$



5. The client requests access to a service on behalf of the user.

For this purpose, the client transmits a message to the server containing the user’s ID and the service-granting ticket.

The server authenticates by using the contents of the ticket.

Kerberos

Kerberos version 4 - A MORE SIMPLE AUTHENTICATION DIALOGUE

Problem 1:

Lifetime associated with the ticket-granting ticket.

If this lifetime is very short (e.g., minutes), then the user will be repeatedly asked for a password.

If the lifetime is long (e.g., hours), then an opponent has a greater opportunity for replay.

Kerberos

Kerberos version 4 - A MORE SIMPLE AUTHENTICATION DIALOGUE

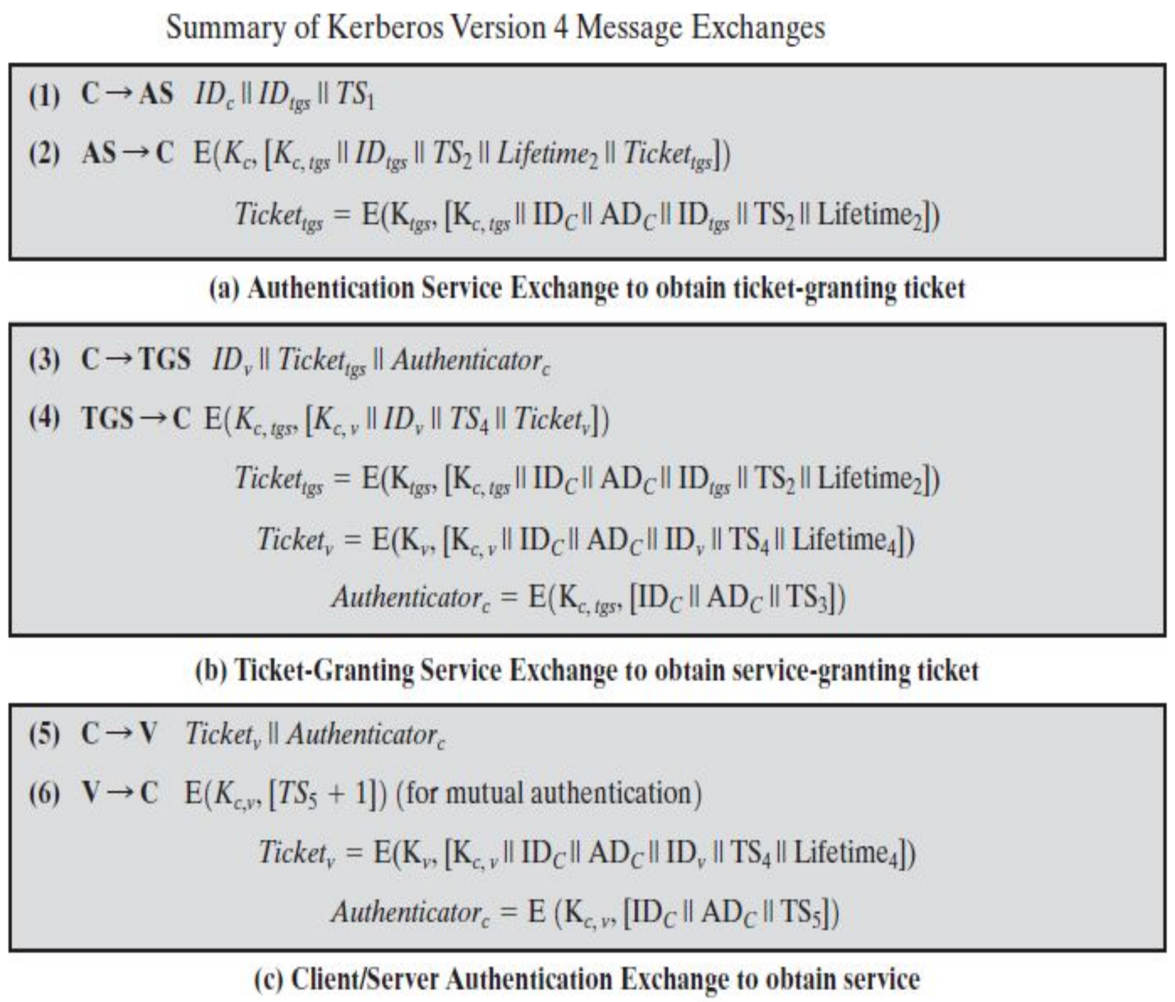
Problem 2:

The second problem is that there may be a requirement for servers to authenticate themselves to users.

Without such authentication, an opponent could sabotage the configuration so that messages to a server were directed to another location.

The false server would then be in a position to act as a real server and capture any information from the user and deny the true service to the user.

Kerberos - THE VERSION 4 AUTHENTICATION DIALOGUE



Summary of Kerberos Version 4 Message Exchanges

(1) **C → AS** $ID_c \| ID_{tgs} \| TS_1$

(2) **AS → C** $E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$

$$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) **C → TGS** $ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) **TGS → C** $E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$

$$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$$

$$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$$

$$Authenticator_c = E(K_{c,tgs}, [ID_C \| AD_C \| TS_3])$$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

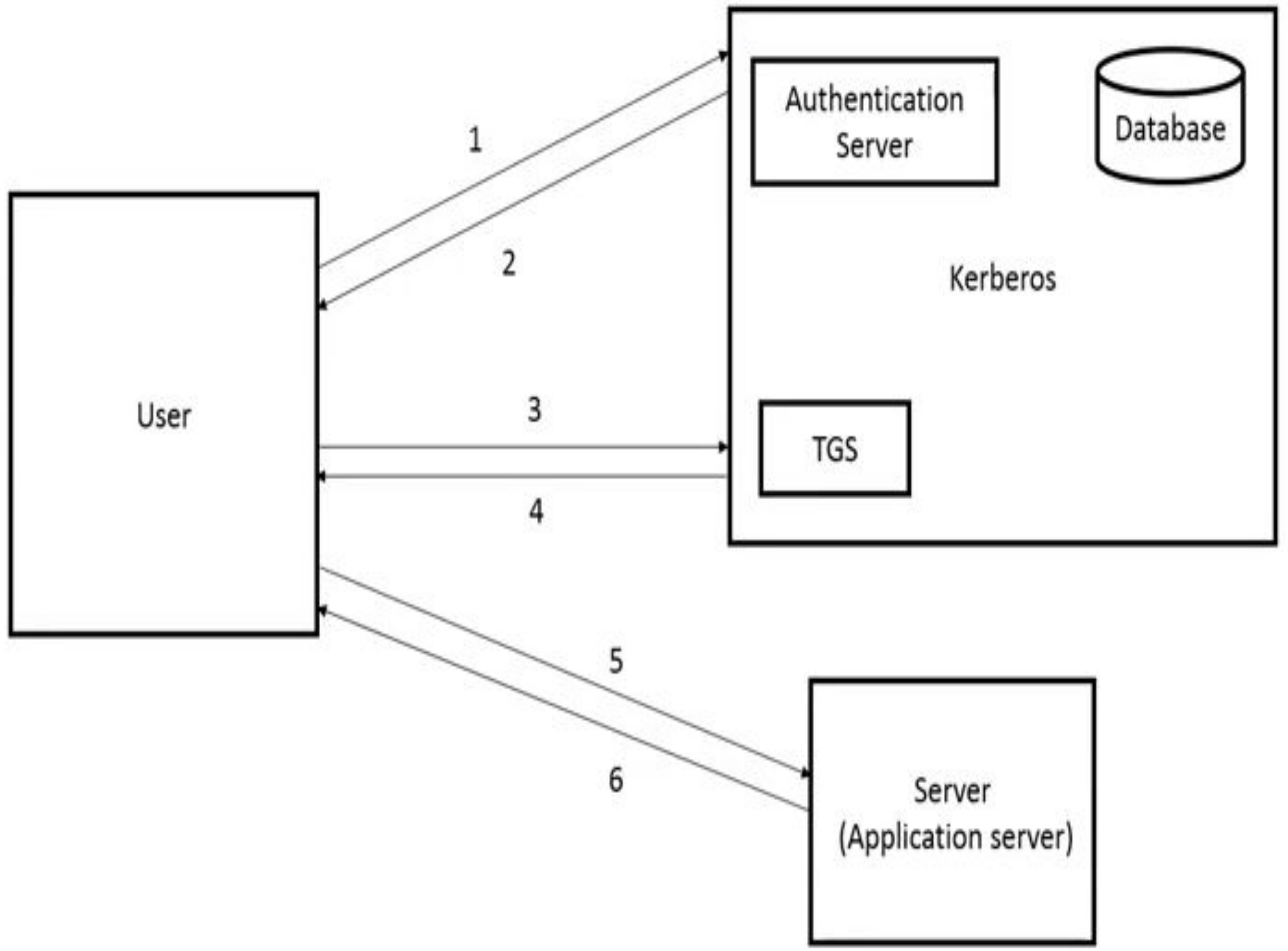
(5) **C → V** $Ticket_v \| Authenticator_c$

(6) **V → C** $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

$$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$$

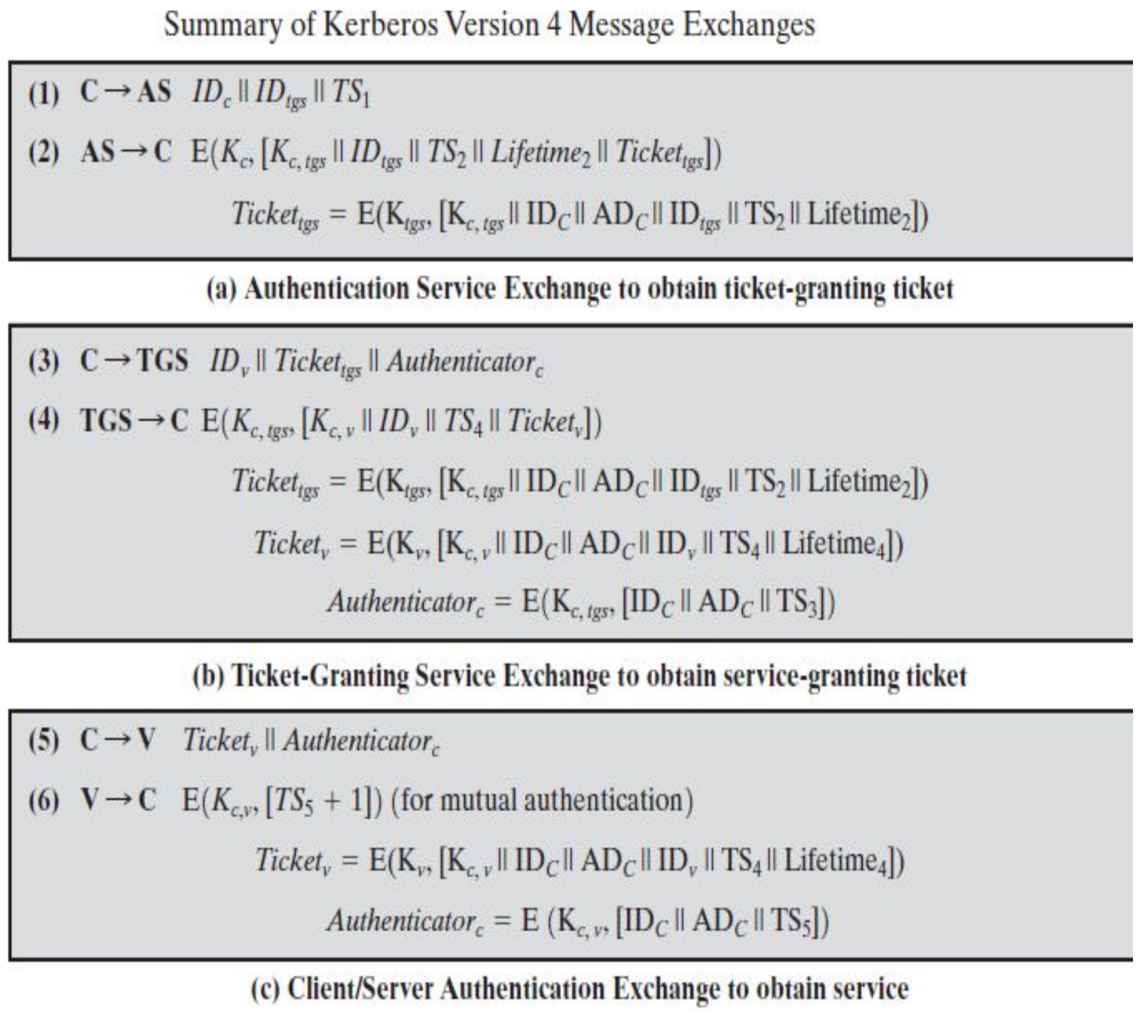
$$Authenticator_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$$

(c) Client/Server Authentication Exchange to obtain service



RAMAIAH Institute of Technology

Kerberos - THE VERSION 4 AUTHENTICATION DIALOGUE



Summary of Kerberos Version 4 Message Exchanges

(1) C → AS $ID_c \| ID_{tgs} \| TS_1$

(2) AS → C $E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}])$

$$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C → TGS $ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C $E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$

$$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$$

$$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$$

$$Authenticator_c = E(K_{c,tgs}, [ID_C \| AD_C \| TS_3])$$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

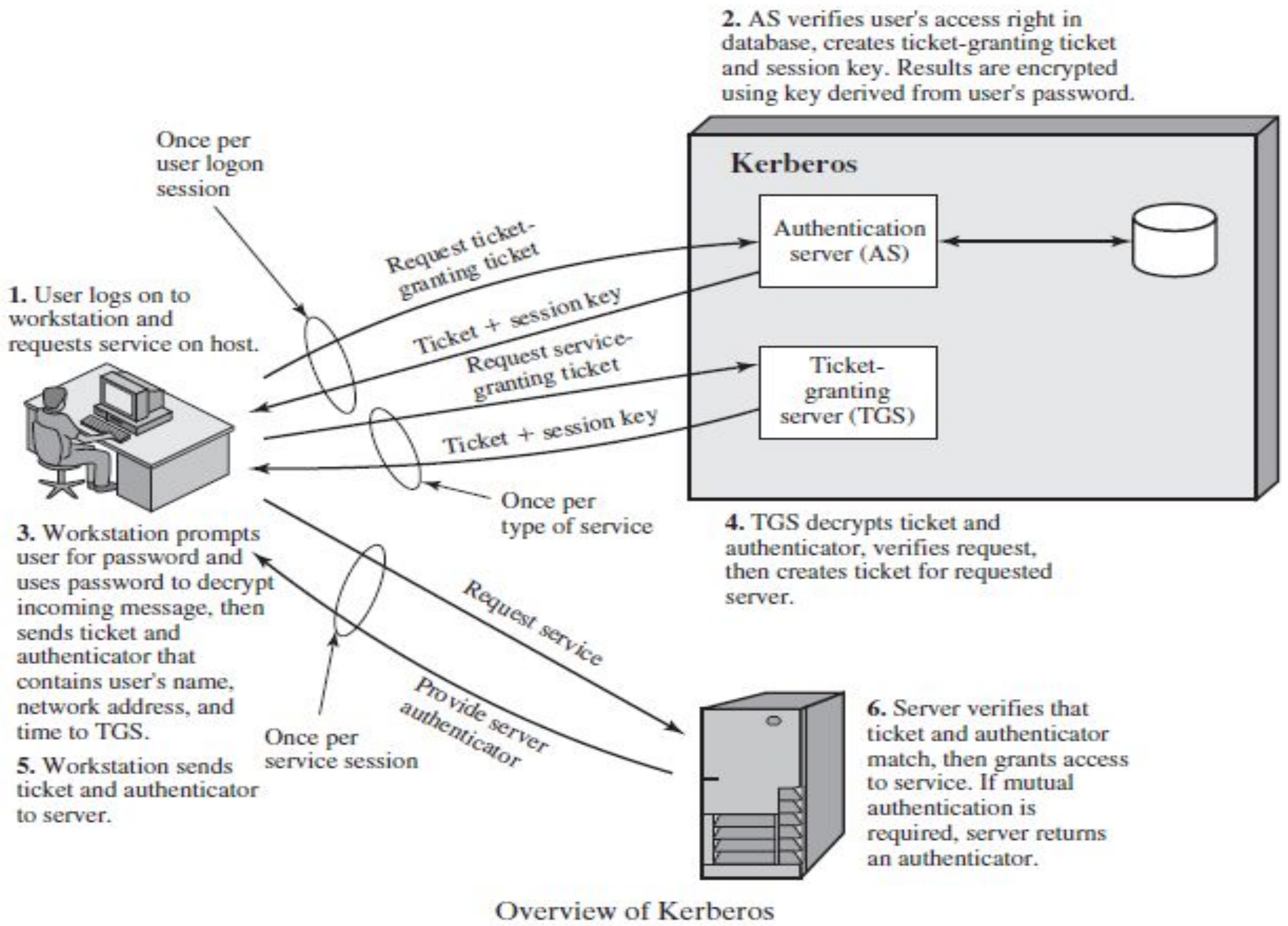
(5) C → V $Ticket_v \| Authenticator_c$

(6) V → C $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

$$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$$

$$Authenticator_c = E(K_{c,v}, [ID_C \| AD_C \| TS_5])$$

(c) Client/Server Authentication Exchange to obtain service


2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
Once per user logon session
Kerberos
Authentication server (AS)
Request ticket-granting ticket
Ticket + session key
1. User logs on to workstation and requests service on host.
Request service-granting ticket
Ticket-granting server (TGS)
Ticket + session key
Once per type of service
3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.
4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.
Request service
Provide server authenticator
Once per service session
5. Workstation sends ticket and authenticator to server.
6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.
Overview of Kerberos

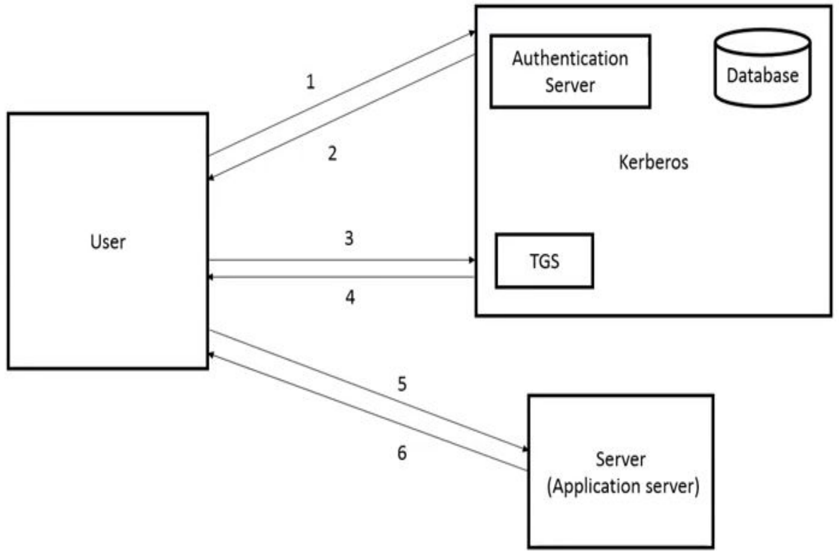
(1) $\mathbf{C \rightarrow AS}$ $ID_c \parallel ID_{tgs} \parallel TS_1$

(2) $\mathbf{AS \rightarrow C}$ $E(K_c, [K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}])$

$$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$$

Message (1)	Client requests ticket-granting ticket.
ID_C	Tells AS identity of user from this client.
ID_{tgs}	Tells AS that user requests access to TGS.
TS_1	Allows AS to verify that client's clock is synchronized with that of AS.
Message (2)	AS returns ticket-granting ticket.
K_c	Encryption is based on user's password, enabling AS and client to verify password, and protecting contents of message (2).
$K_{c,tgs}$	Copy of session key accessible to client created by AS to permit secure exchange between client and TGS without requiring them to share a permanent key.
ID_{tgs}	Confirms that this ticket is for the TGS.
TS_2	Informs client of time this ticket was issued.
$Lifetime_2$	Informs client of the lifetime of this ticket.
$Ticket_{tgs}$	Ticket to be used by client to access TGS.

(a) Authentication Service Exchange



K_c = key that is derived from user password
$K_{c,tgs}$ = session key for C and TGS
K_{tgs} = key shared only by the AS and the TGS

RAMAIAH Institute of Technology

$$\textbf{(3)}\quad \mathbf{C \rightarrow TGS}\quad ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$$

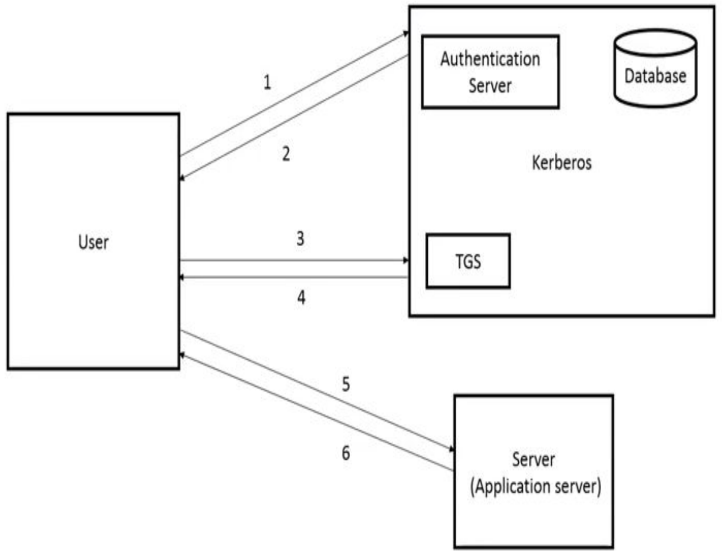
$$\textbf{(4)}\quad \mathbf{TGS \rightarrow C}\quad \mathrm{E}(K_{c,\,tgs}, [K_{c,\,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v])$$

$$Ticket_{tgs} = \mathrm{E}(\mathrm{K}_{tgs}, [\mathrm{K}_{c,\,tgs} \parallel \mathrm{ID}_C \parallel \mathrm{AD}_C \parallel \mathrm{ID}_{tgs} \parallel \mathrm{TS}_2 \parallel \mathrm{Lifetime}_2])$$

$$Ticket_v = \mathrm{E}(\mathrm{K}_v, [\mathrm{K}_{c,\,v} \parallel \mathrm{ID}_C \parallel \mathrm{AD}_C \parallel \mathrm{ID}_v \parallel \mathrm{TS}_4 \parallel \mathrm{Lifetime}_4])$$

$$Authenticator_c = \mathrm{E}(\mathrm{K}_{c,\,tgs}, [\mathrm{ID}_C \parallel \mathrm{AD}_C \parallel \mathrm{TS}_3])$$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket



Message (3)	Client requests service-granting ticket.
ID_V	Tells TGS that user requests access to server V.
$Ticket_{tgs}$	Assures TGS that this user has been authenticated by AS.
$Authenticator_c$	Generated by client to validate ticket.

K_c = key that is derived from user password
$K_{c,tgs}$ = session key for C and TGS
K_{tgs} = key shared only by the AS and the TGS

(3) **C → TGS** $ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) **TGS → C** $E(K_{c,tgs}, [K_{c,v} \| ID_v \| TS_4 \| Ticket_v])$

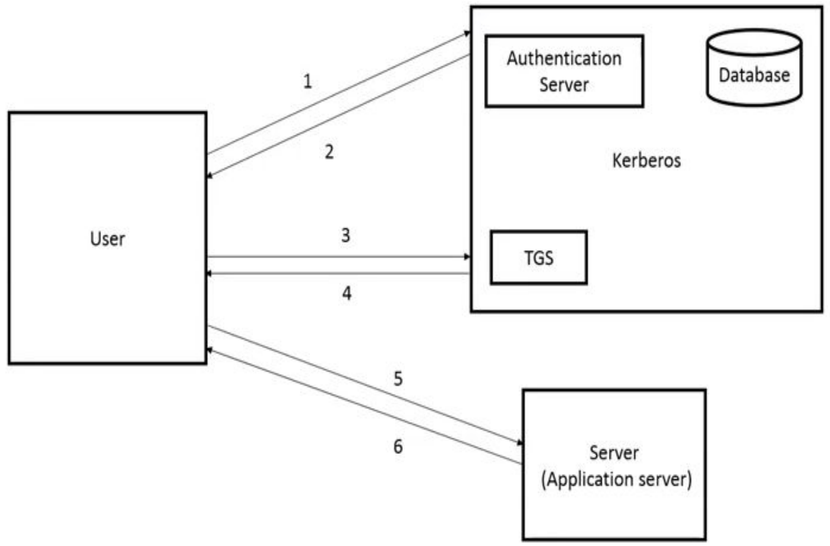
$$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2])$$

$$Ticket_v = E(K_v, [K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4])$$

$$Authenticator_c = E(K_{c,tgs}, [ID_C \| AD_C \| TS_3])$$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

Message (4)	TGS returns service-granting ticket.
$K_{c,tgs}$	Key shared only by C and TGS protects contents of message (4).
$K_{c,v}$	Copy of session key accessible to client created by TGS to permit secure exchange between client and server without requiring them to share a permanent key.
ID_V	Confirms that this ticket is for server V.
TS_4	Informs client of time this ticket was issued.
K_v	Ticket is encrypted with key known only to TGS and server, to prevent tampering.



RAMAIAH
Institute of Technology

Kerberos - THE VERSION 4 AUTHENTICATION DIALOGUE

Summary of Kerberos Version 4 Message Exchanges

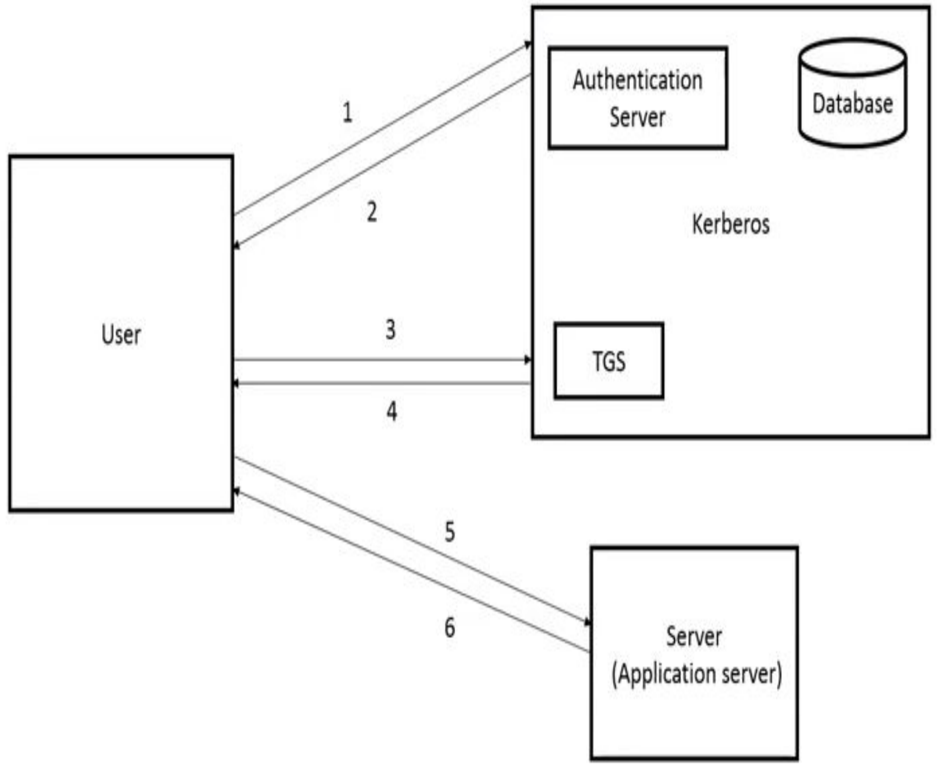
(5) $\mathbf{C \rightarrow V}$ $\quad Ticket_v \parallel Authenticator_c$

(6) $\mathbf{V \rightarrow C}$ $\quad E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

$$Ticket_v = E(K_v, [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$$

$$Authenticator_c = E(K_{c,v}, [ID_C \parallel AD_C \parallel TS_5])$$

(c) Client/Server Authentication Exchange to obtain service



Summary of Kerberos Version 4 Message Exchanges

(1) $C \rightarrow AS \quad ID_c \parallel ID_{tgs} \parallel TS_1$

(2) $AS \rightarrow C \quad E(K_c, [K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}])$

$$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) $C \rightarrow TGS \quad ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$

(4) $TGS \rightarrow C \quad E(K_{c,tgs}, [K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v])$

$$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$$

$$Ticket_v = E(K_v, [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$$

$$Authenticator_c = E(K_{c,tgs}, [ID_C \parallel AD_C \parallel TS_3])$$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

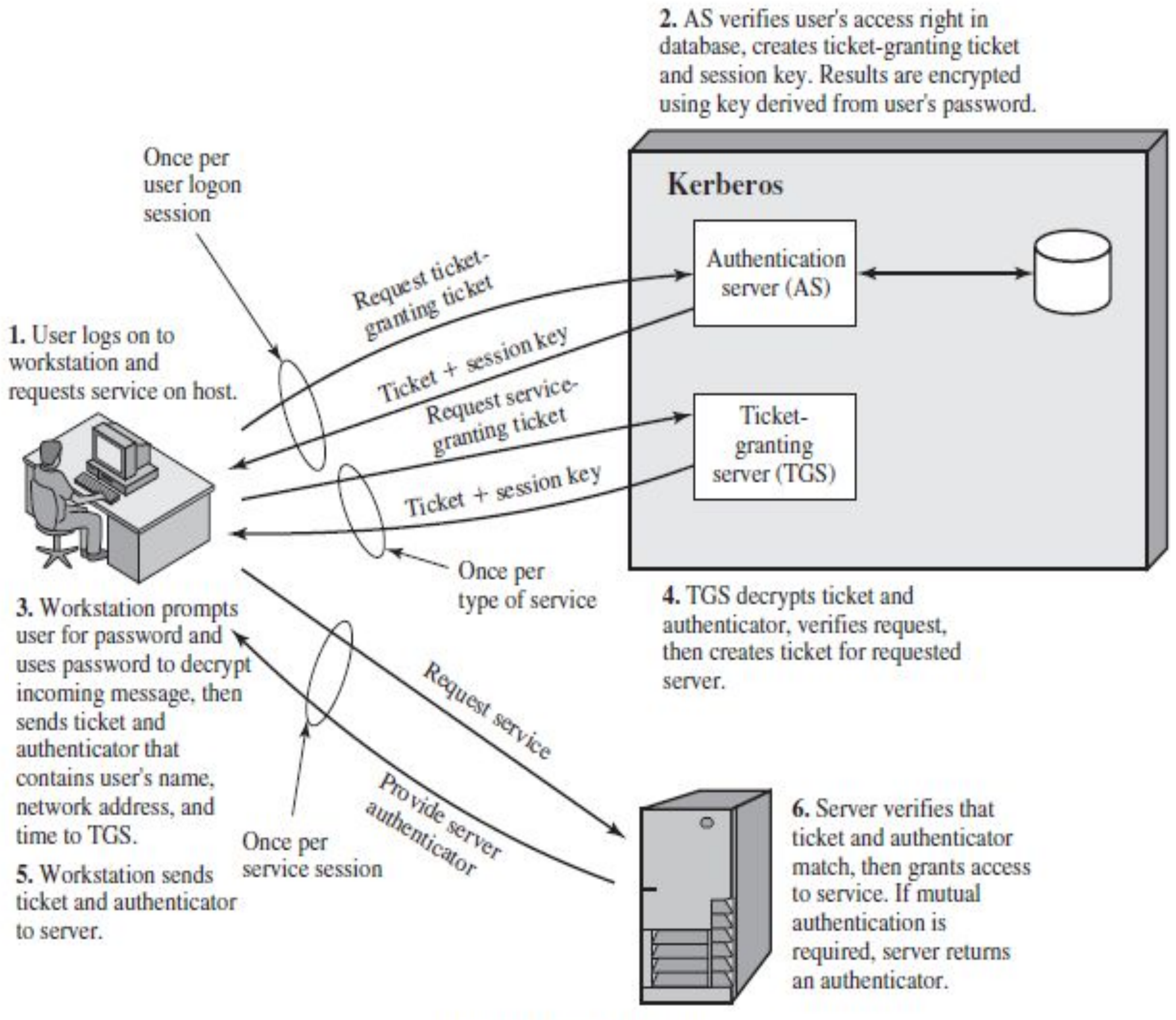
(5) $C \rightarrow V \quad Ticket_v \parallel Authenticator_c$

(6) $V \rightarrow C \quad E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

$$Ticket_v = E(K_v, [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$$

$$Authenticator_c = E(K_{c,v}, [ID_C \parallel AD_C \parallel TS_5])$$

(c) Client/Server Authentication Exchange to obtain service


Once per user logon session
Kerberos
Authentication server (AS)
Request ticket-granting ticket
Ticket + session key
Request service-granting ticket
Ticket + session key
Ticket-granting server (TGS)
Once per type of service
Request service
Provide server authenticator
Once per service session

1. User logs on to workstation and requests service on host.

2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.

3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.

4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.

5. Workstation sends ticket and authenticator to server.

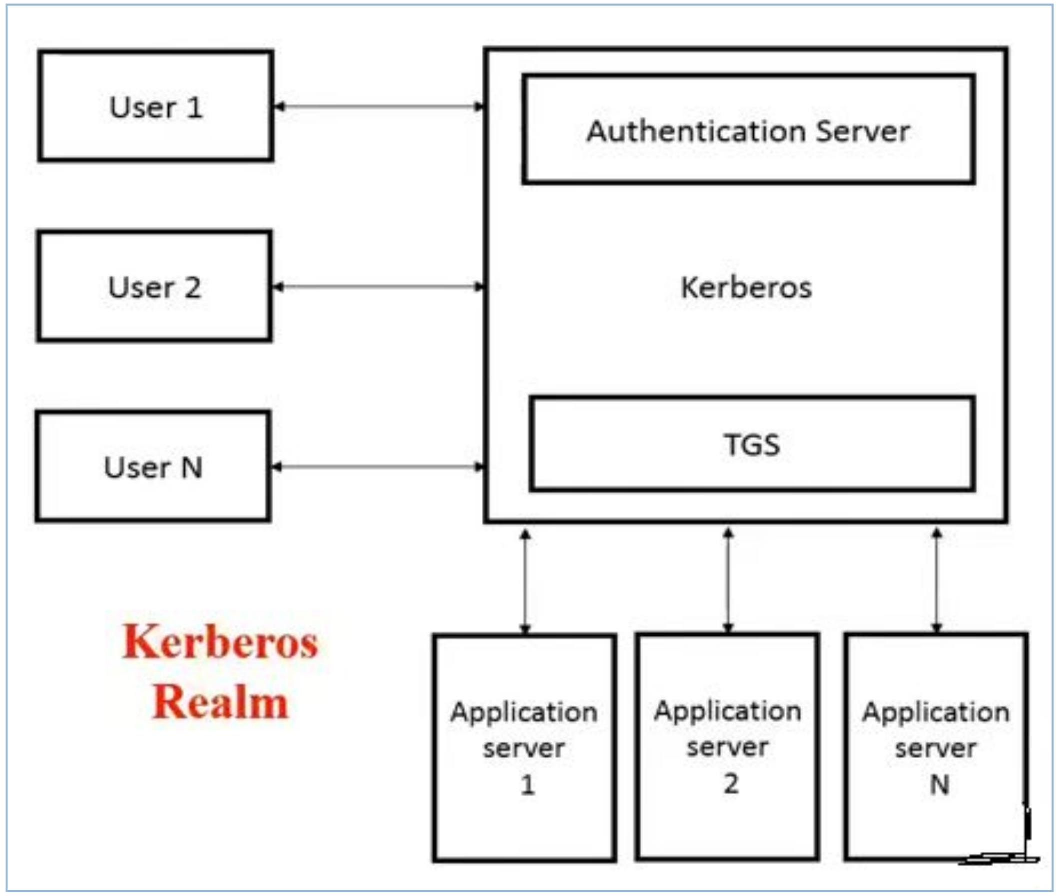
6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Overview of Kerberos

RAMAIAH
Institute of Technology

KERBEROS REALM

- A full-service Kerberos environment consists of
  - a Kerberos server
  - a number of clients, all are registered with Kerberos server
  - a number of application servers, all are sharing keys with Kerberos server
- Such an environment is referred to as a **Kerberos realm.**



RAMAIAH
Institute of Technology

Inter Realm Authentication

(1) $C \rightarrow AS$: $ID_c \| \boxed{ID_{tgs}} \| TS_1$

(2) $AS \rightarrow C$: $E(K_c, [K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| \boxed{Ticket_{tgs}}])$

(3) $C \rightarrow TGS$: $\boxed{ID_{tgsrem}} \| Ticket_{tgs} \| Authenticator_c$

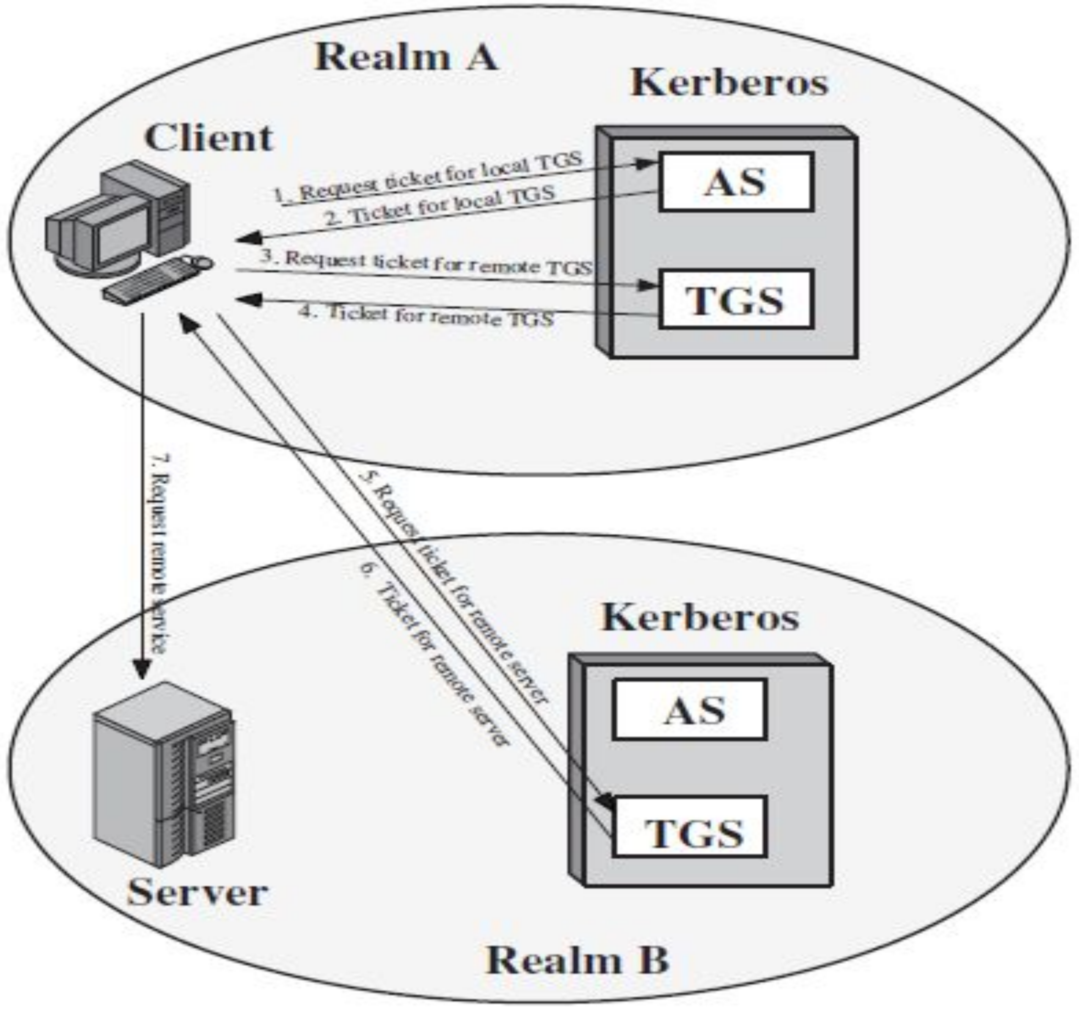
(4) $TGS \rightarrow C$: $E(K_{c,tgs}, [K_{c,tgsrem} \| ID_{tgsrem} \| TS_4 \| \boxed{Ticket_{tgsrem}}])$

(5) $C \rightarrow TGS_{rem}$: $\boxed{ID_{vrem}} \| Ticket_{tgsrem} \| Authenticator_c$

(6) $TGS_{rem} \rightarrow C$: $E(K_{c,tgsrem}, [K_{c,vrem} \| ID_{vrem} \| TS_6 \| \boxed{Ticket_{vrem}}])$

(7) $C \rightarrow V_{rem}$: $Ticket_{vrem} \| Authenticator_c$

1. C → AS : Request ticket for local TGS
2. AS → C : Ticket for local TGS
3. C → TGS : Request ticket for remote TGS
4. TGS → C : Ticket for remote TGS
5. C → TGS_{rem} : Request ticket for remote Server
6. TGS_{rem} → C : Ticket for remote Server
7. C → V_{rem} : Request for remote service


Realm A
Kerberos
Client
AS
TGS
1. Request ticket for local TGS
2. Ticket for local TGS
3. Request ticket for remote TGS
4. Ticket for remote TGS
7. Request remote service
5. Request ticket for remote server
6. Ticket for remote server
Kerberos
AS
TGS
Server
Realm B
Request for Service in Another Realm

Kerberos Version 5

Kerberos version 5 is specified in RFC 4120 and provides a number of improvements over version 4.

- Differences between versions 4 and 5
- Version 5 protocol

Kerberos Version 5

Differences between versions 4 and 5

Version 5 is intended to address the limitations of version 4 in two areas:

1. Environmental shortcomings
2. Technical deficiencies

Kerberos Version 5

Environmental shortcomings

1. Encryption system dependence
2. Internet protocol dependence
3. Message byte ordering
4. Ticket lifetime
5. Authentication forwarding
6. Interrealm authentication

Kerberos Version 5

Environmental shortcomings

1.Encryption system dependence

- Version 4 ***use only DES (Data encryption Standards).***
- In version 5, ***any encryption techniques can be used.***

2.Internet protocol dependence

- Version 4 supports ***Internet Protocol (IP) addresses***, but cannot support ***ISO network address.***
- Version 5 supports ***any types of network addresses*** with variable length.

Kerberos Version 5

Environmental shortcomings

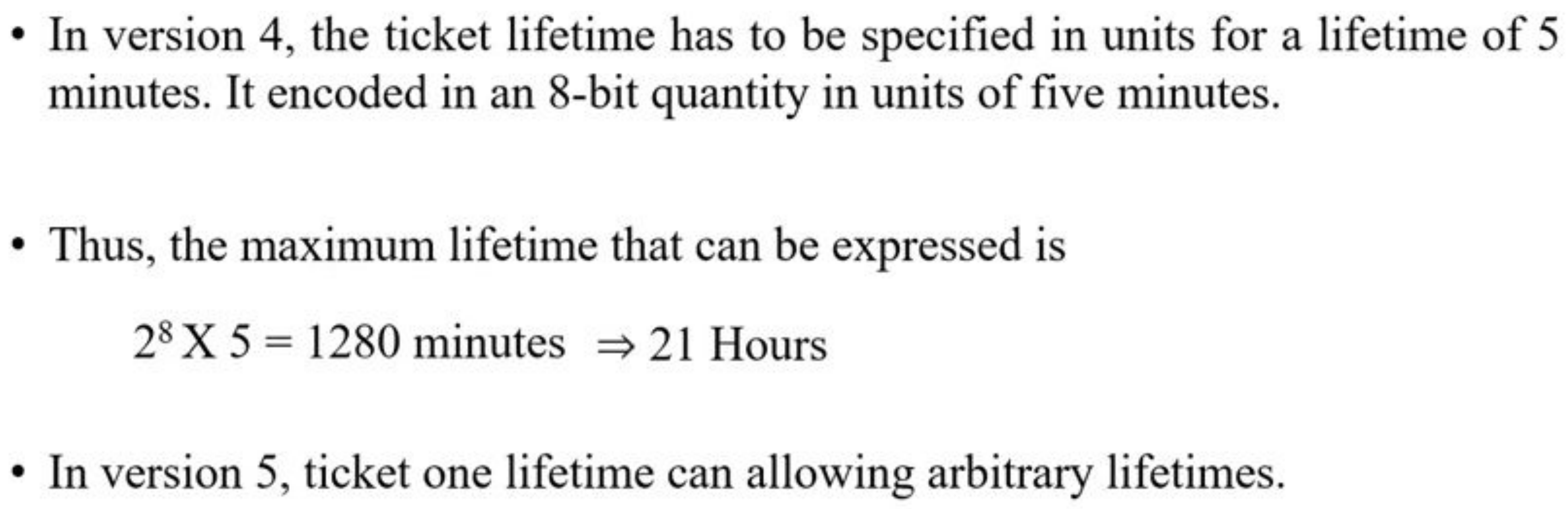
3. Message byte ordering

- In version 4, the sender of a message employs a byte ordering of its own choosing and tags the message to indicate least significant byte in lowest address or most significant byte in lowest address.
- In version 5, all message structures are defined using Abstract Syntax Notation One (ASN.1) and Basic Encoding Rules (BER), which provide an unambiguous byte ordering.

Kerberos Version 5

Environmental shortcomings

4. Ticket lifetime



- In version 4, the ticket lifetime has to be specified in units for a lifetime of 5 minutes. It encoded in an 8-bit quantity in units of five minutes.

- Thus, the maximum lifetime that can be expressed is

  $2^8 X 5 = 1280$ minutes $\Rightarrow 21$ Hours

- In version 5, ticket one lifetime can allowing arbitrary lifetimes.

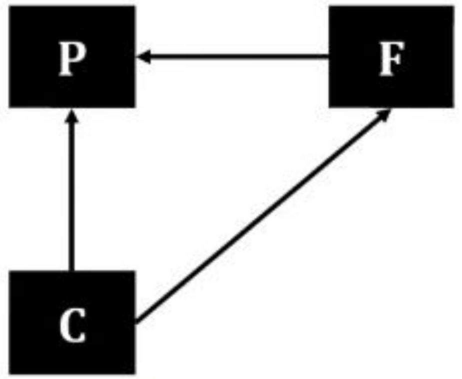
RAMAIAH Institute of Technology

Kerberos Version 5

Environmental shortcomings

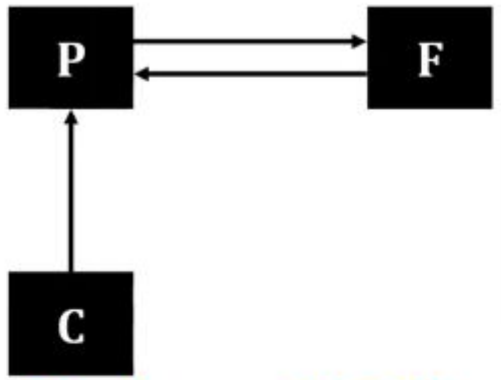
5.Authentication forwarding

- Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client.
- Version 5 provides authentication forwarding, it means client to access a server and have that server access another server on behalf of the client.



Kerberos V4

C = Client
P = Print Server
F = File Server



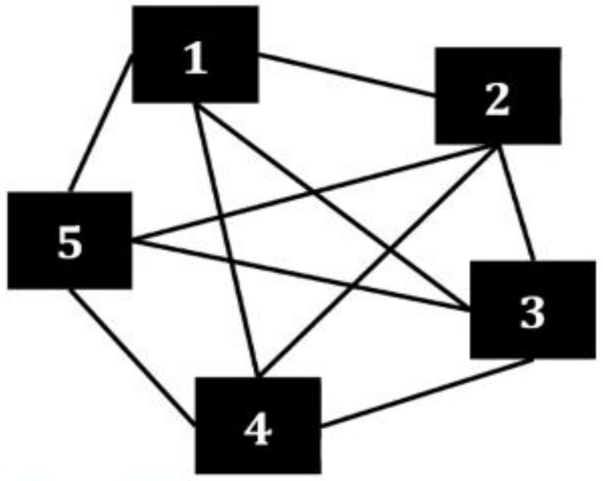
Kerberos V 5.0

Kerberos Version 5

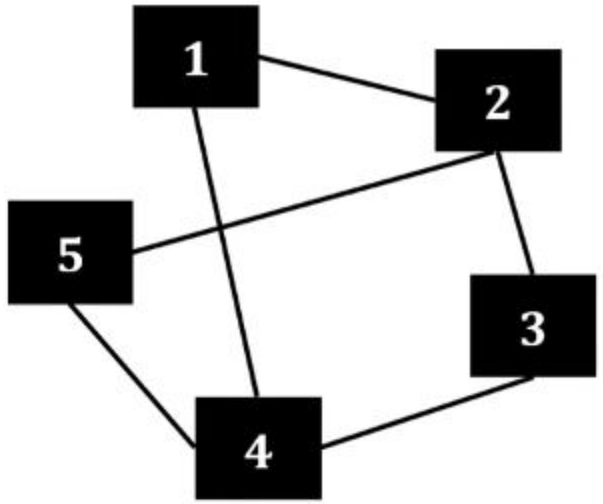
Environmental shortcomings

6.Interrealm authentication

- In version 4, interoperability among realms requires ***many Kerberos - to - Kerberos relationships***.
- In version 5 supports a method that requires ***fewer relationships***.



Kerberos V4



Kerberos Version 5

Technical deficiencies

1. Double encryption
2. PCBC encryption
3. Session keys
4. Password attacks

Kerberos Version 5

1. Double encryption

In Kerberos Version4, tickets provided to clients are encrypted twice—once with the secret key of the target server and then again with a secret key known to the client.

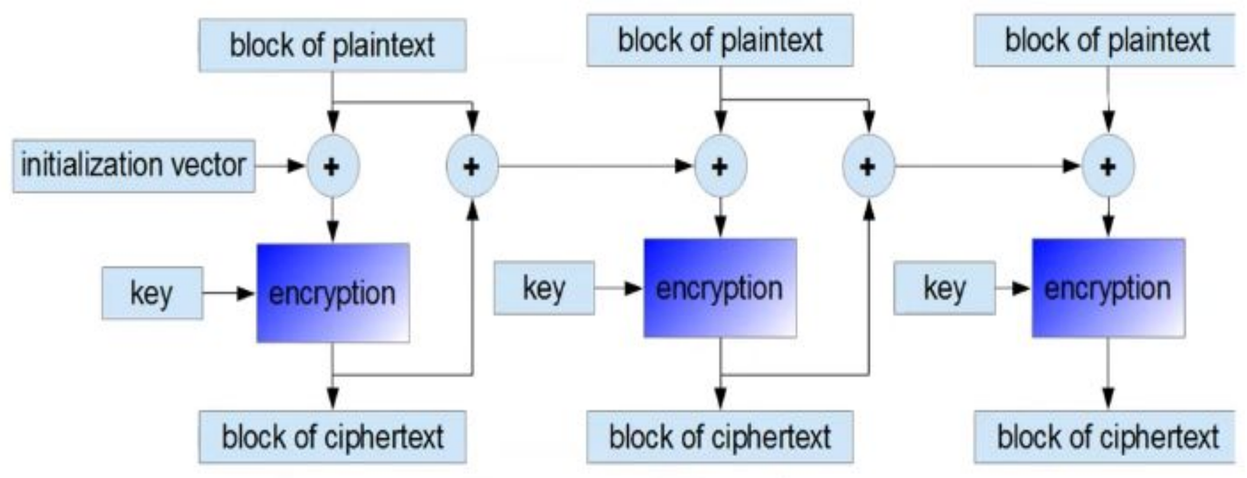
The second encryption is not necessary and is computationally wasteful.

RAMAIAH Institute of Technology

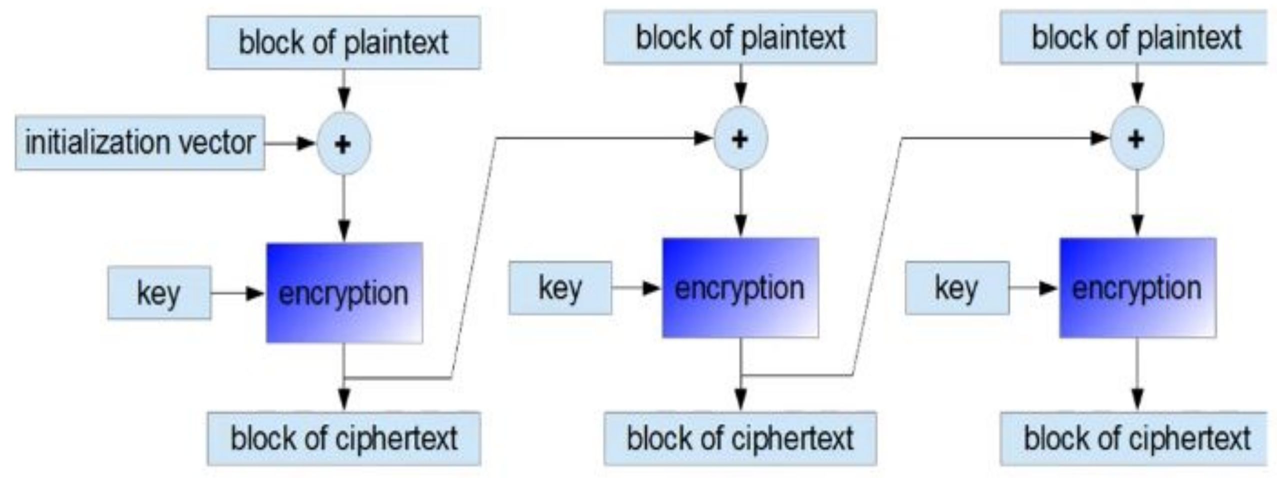
Kerberos Version 5

PCBC encryption

- Encryption in version 4 makes use of a non standard mode of DES known as ***propagating cipher block chaining (PCBC).*** This mode is vulnerable to an attack involving the ***interchange of ciphertext blocks.***
- Version 5 allows the ***standard CBC mode*** to be used for encryption.



PCBC Encryption

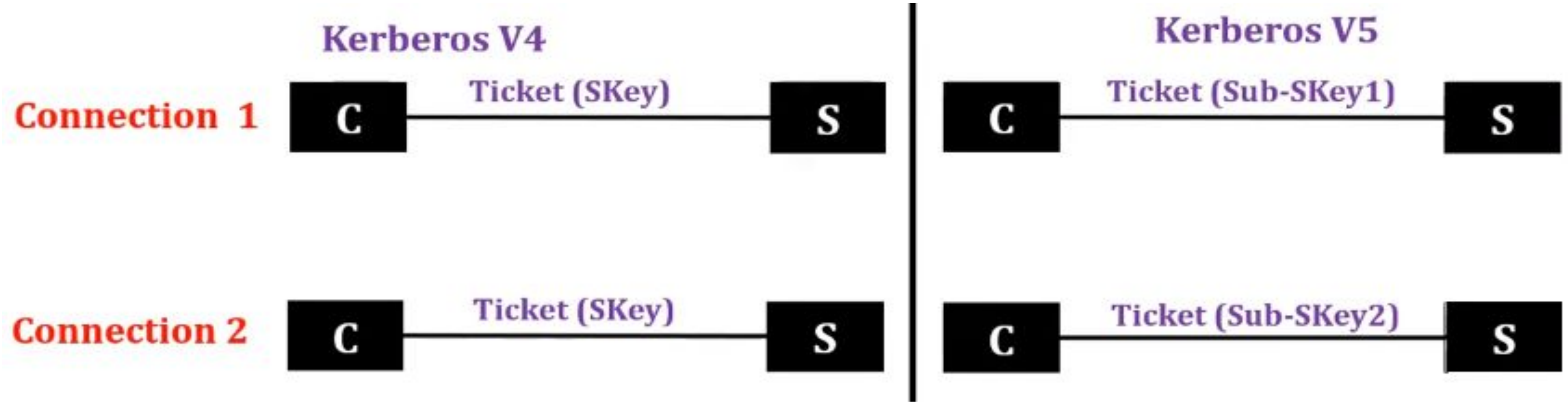


CBC Encryption

Kerberos Version 5

Session keys

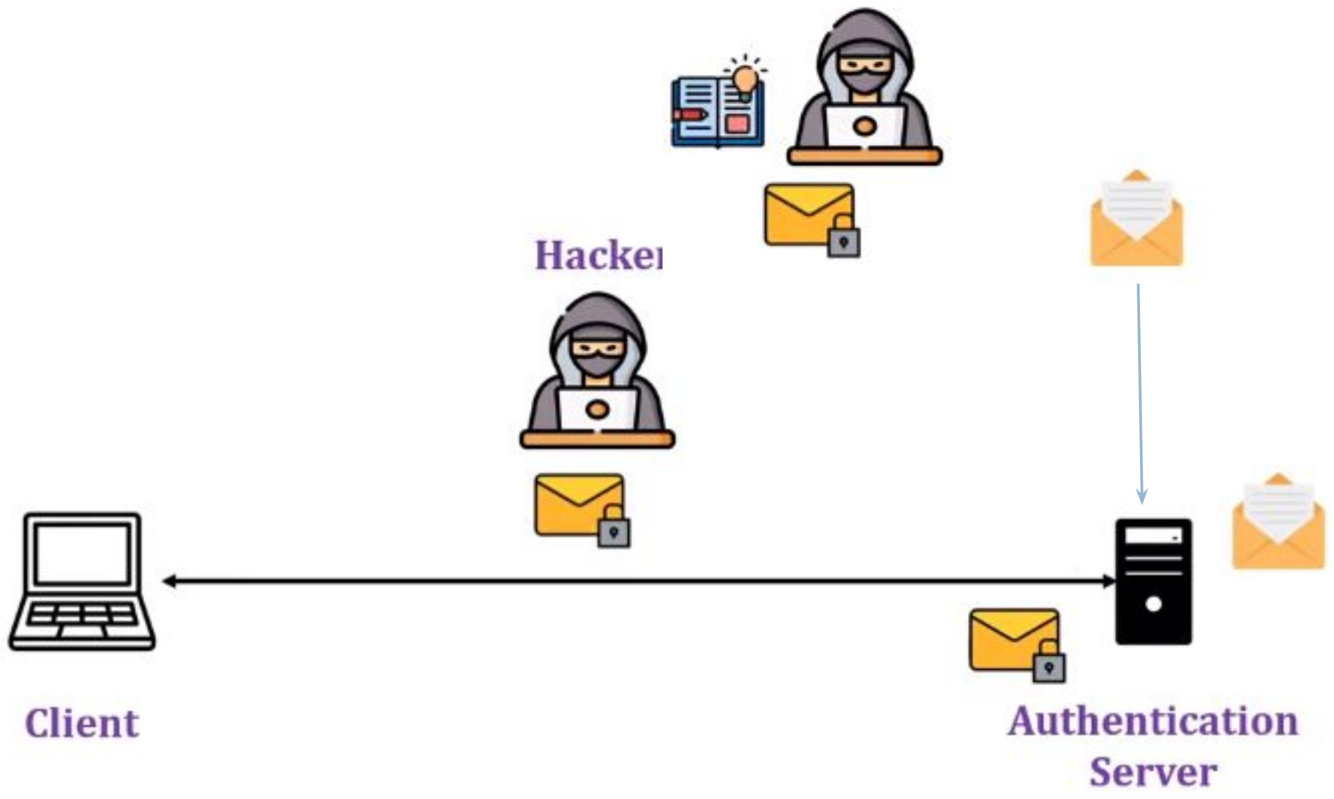
- Each ticket includes a ***session key used for encrypting messages.***
- However, because the same ticket may be used repeatedly, ***replay attack is possible.***
- In version 5, it is possible for a client and server to negotiate a ***sub-session key***, which is to be used only for that one connection.



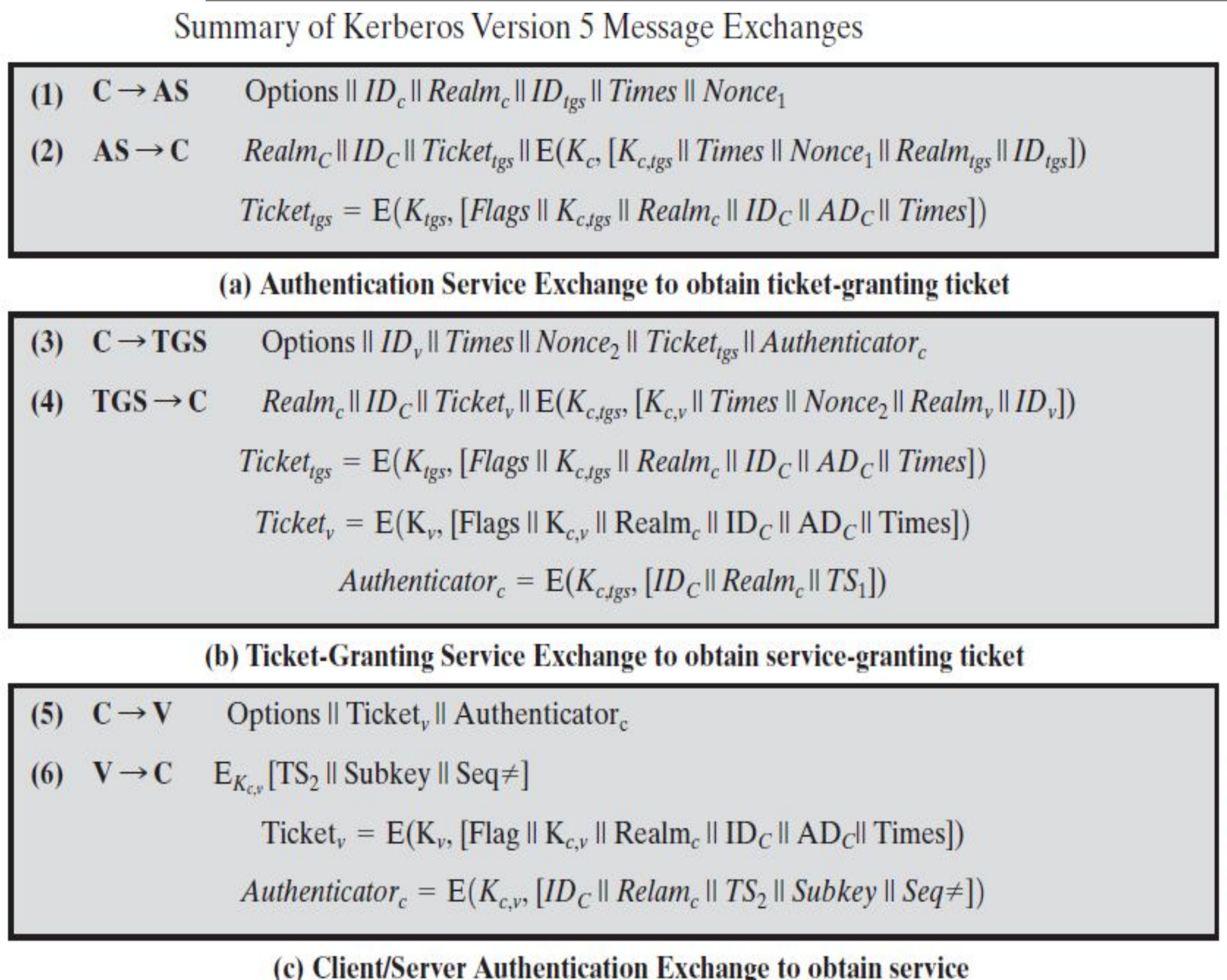
Kerberos Version 5

Password attacks

Both versions are vulnerable to a password attack



Kerberos Version 5



Summary of Kerberos Version 5 Message Exchanges

(1) C → AS $\quad$ Options ‖ ID_c ‖ $Realm_c$ ‖ ID_{tgs} ‖ $Times$ ‖ $Nonce_1$

(2) AS → C $\quad$ $Realm_C$ ‖ ID_C ‖ $Ticket_{tgs}$ ‖ $E(K_c, [K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C → TGS $\quad$ Options ‖ ID_v ‖ $Times$ ‖ $Nonce_2$ ‖ $Ticket_{tgs}$ ‖ $Authenticator_c$

(4) TGS → C $\quad$ $Realm_c$ ‖ ID_C ‖ $Ticket_v$ ‖ $E(K_{c,tgs}, [K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_v])$

$Ticket_{tgs} = E(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$

$Ticket_v = E(K_v, [Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times])$

$Authenticator_c = E(K_{c,tgs}, [ID_C \| Realm_c \| TS_1])$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

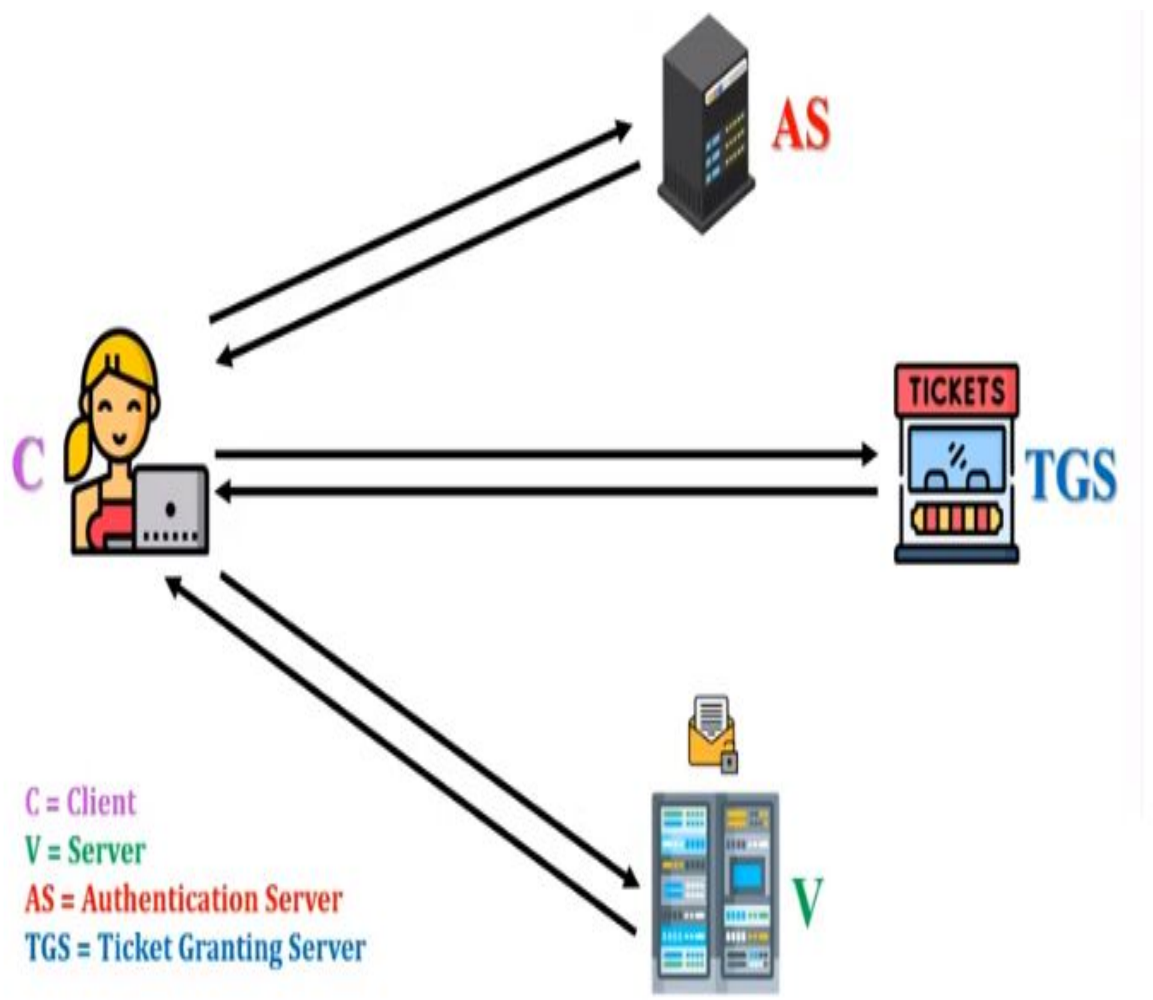
(5) C → V $\quad$ Options ‖ $Ticket_v$ ‖ $Authenticator_c$

(6) V → C $\quad$ $E_{K_{c,v}}[TS_2 \| Subkey \| Seq\neq]$

$Ticket_v = E(K_v, [Flag \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times])$

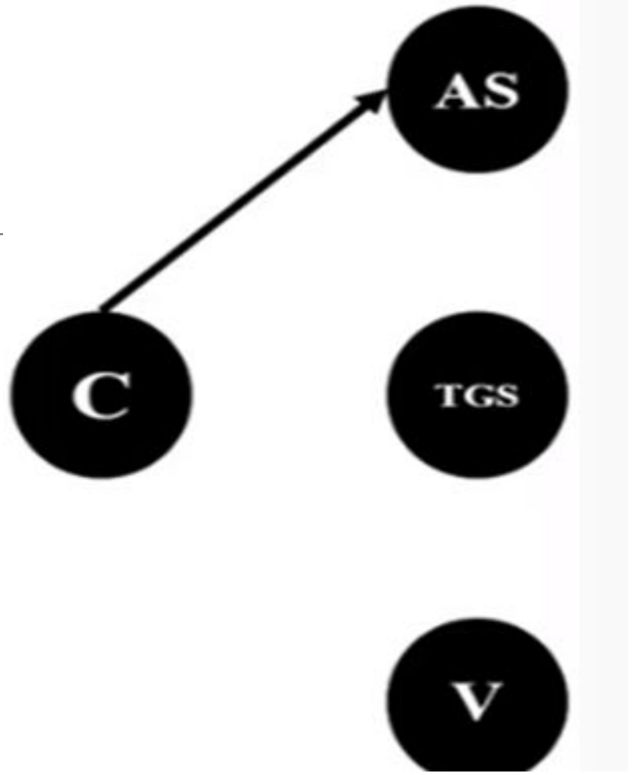
$Authenticator_c = E(K_{c,v}, [ID_C \| Relam_c \| TS_2 \| Subkey \| Seq\neq])$

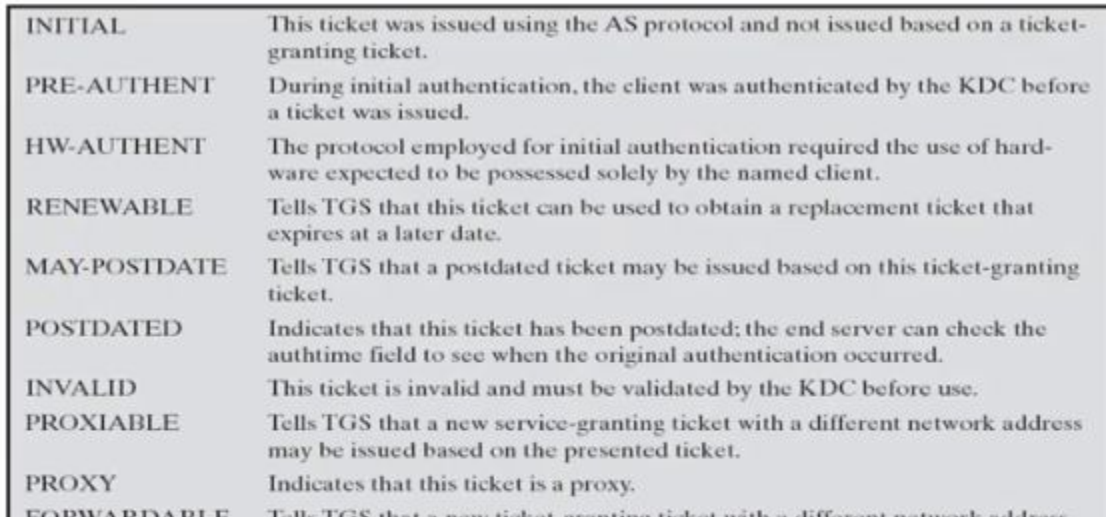
(c) Client/Server Authentication Exchange to obtain service



1) C ➔ AS: Options || ID_c || $Realm_c$ || ID_{tgs} || Times || $Nonce_1$

- **Options:** Used to request that certain flags be set in the returned ticket
- **Realm:** Indicates realm of user
- **Times:** Used by the client to request the following time settings in the ticket:
    - **from:** start time for the requested ticket
    - **till:** expiration time for the requested ticket
    - **rtime:** requested renew-till time
- **Nonce:** A random value to avoid replay attack





INITIAL	This ticket was issued using the AS protocol and not issued based on a ticket-granting ticket.
PRE-AUTHENT	During initial authentication, the client was authenticated by the KDC before a ticket was issued.
HW-AUTHENT	The protocol employed for initial authentication required the use of hardware expected to be possessed solely by the named client.
RENEWABLE	Tells TGS that this ticket can be used to obtain a replacement ticket that expires at a later date.
MAY-POSTDATE	Tells TGS that a postdated ticket may be issued based on this ticket-granting ticket.
POSTDATED	Indicates that this ticket has been postdated; the end server can check the authtime field to see when the original authentication occurred.
INVALID	This ticket is invalid and must be validated by the KDC before use.
PROXIABLE	Tells TGS that a new service-granting ticket with a different network address may be issued based on the presented ticket.
PROXY	Indicates that this ticket is a proxy.
FORWARDABLE	Tells TGS that a new ticket-granting ticket with a different network address

Kerberos Version 5

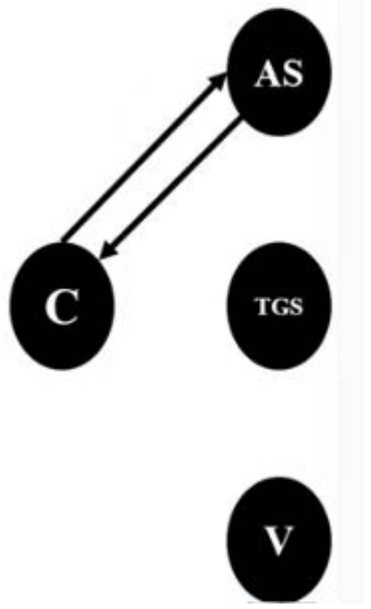
(2) AS → C $\quad Realm_C \| ID_C \| Ticket_{tgs} \| E(K_c, [K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}])$

$$Ticket_{tgs} = E(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$$

K_c = Share between C & AS

$K_{c,tgs}$ = Shared between C & TGS

K_{tgs} = Shared between TGS & AS

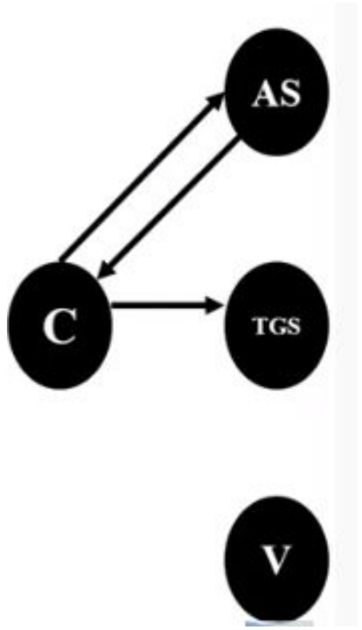


Kerberos Version 5

3) C ➔ TGS: Options || ID_v || Times || $Nonce_2$ || $Ticket_{tgs}$ || $Authenticator_c$

$Ticket_{tgs}$ = E(K_{tgs} [Flags || $K_{c,tgs}$ || $Realm_c$ || ID_c || AD_c || Times])

$Authenticator_c$ = E($K_{c,tgs}$ [ID_c|| $Realm_c$ || TS_1])



K_{tgs} = Shared between TGS & AS

$K_{c,tgs}$ = Shared between C & TGS

Kerberos Version 5

(4) **TGS → C** $Realm_c \| ID_C \| Ticket_v \| E(K_{c,tgs}, [K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_v])$

$$Ticket_{tgs} = E(K_{tgs}, [Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times])$$

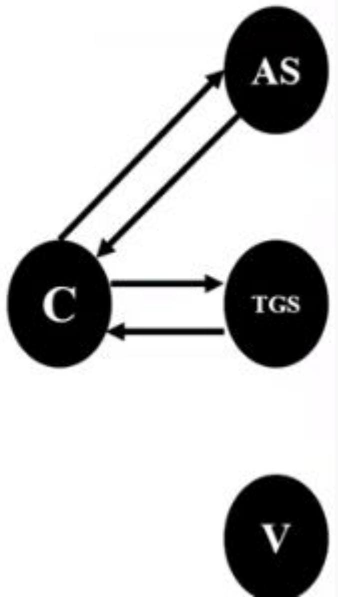
$$Ticket_v = E(K_v, [Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times])$$

$$Authenticator_c = E(K_{c,tgs}, [ID_C \| Realm_c \| TS_1])$$

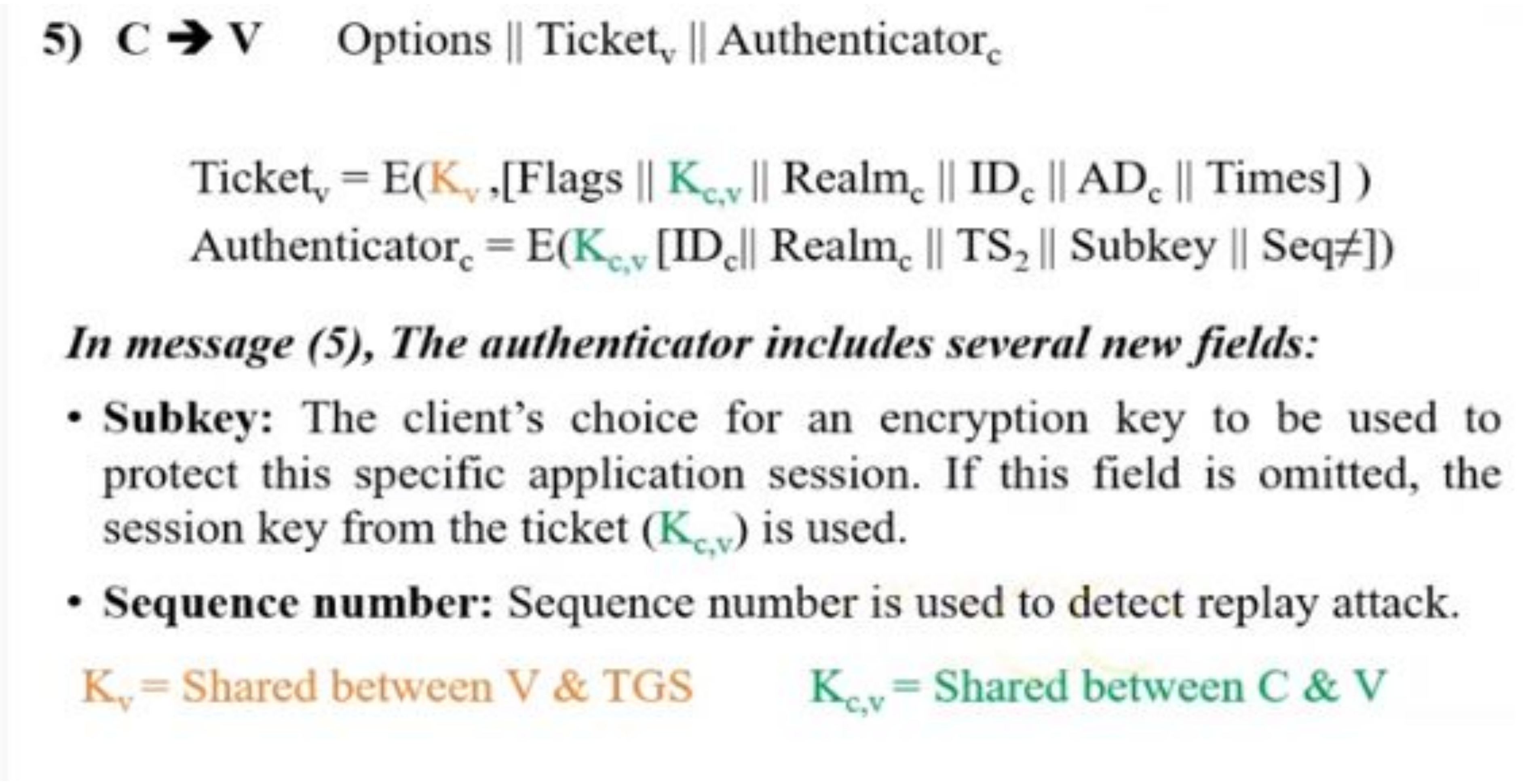
K_{tgs} = Shared between TGS & AS

$K_{c,tgs}$ = Shared between C & TGS

K_v = Shared between V & TGS



Kerberos Version 5



5) C ➔ V   Options || Ticket_v || Authenticator_c

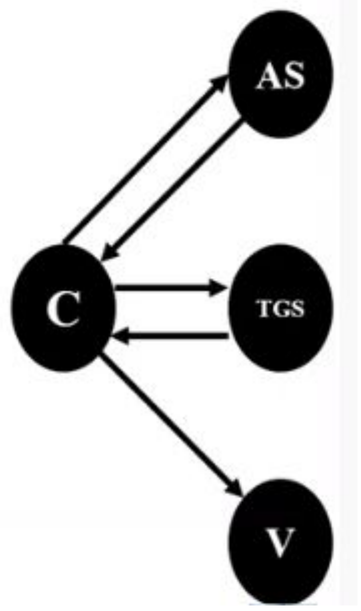
$\text{Ticket}_v = E(K_v, [\text{Flags} \,||\, K_{c,v} \,||\, \text{Realm}_c \,||\, ID_c \,||\, AD_c \,||\, \text{Times}])$

$\text{Authenticator}_c = E(K_{c,v} [ID_c || \text{Realm}_c \,||\, TS_2 \,||\, \text{Subkey} \,||\, \text{Seq}\neq])$

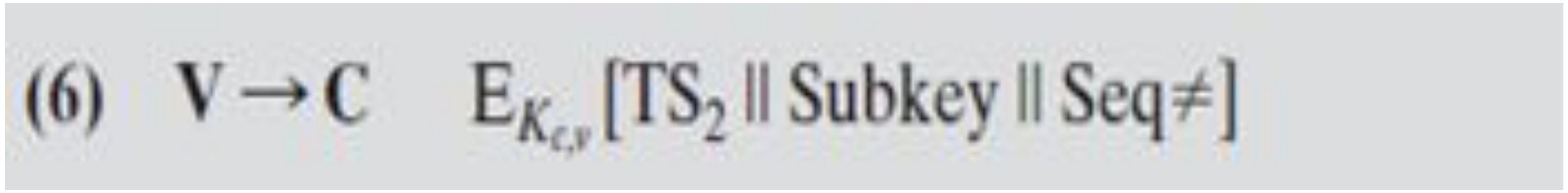
In message (5), The authenticator includes several new fields:

- **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.
- **Sequence number:** Sequence number is used to detect replay attack.

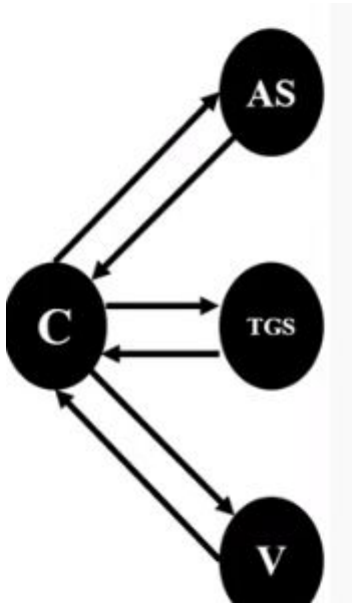
K_v = Shared between V & TGS $K_{c,v}$ = Shared between C & V



Kerberos Version 5



(6) $V \rightarrow C \quad E_{K_{c,v}}[TS_2 \,\|\, Subkey \,\|\, Seq\#]$



THANK YOU