COMPUTER SECURITY AND CYBER LAW (CSCL)

LECTURER: ROLISHA STHAPIT/ DIKSHYA SINGH

UNIT 4

Unit 4: Digital Signature and Authentication Protocols LH 5

- Authentication Basic , Password (Attacking a password system, countering password guessing, Password aging), Challenge Response, Biometrics, Location, Multiple Methods, ,Mutual (Symmetric, Public Key), One-way (Symmetric, Public Key) Digital Signature, Direct Digital Signature, Arbitrated Digital Signature, Digital Certificate, X.509 Certificate, Authentication Protocols, Authentication Services, Kerberos V4, Digital Signature Standards (DSS) , DSS approach Vs RSA approach.

Authentication Basic

Definition: Authentication is the binding of an identity to a subject.

- ➢ Identity is that of external entity like users and the subject is computer entity like process
- ➢ The external entity must provide information to enable the system to confirm its identity. This information comes from one (or more) of the following:

- What the entity knows? (information like passwords or secret information)
- What the entity has? (secure tokens like badge or card)
- What the entity is? (biometrics like fingerprints or retinal characteristics)
- Where the entity is? (in particular terminal like IP Address)

- The authentication process consists of
    - obtaining the authentication information from an entity,
    - analyzing the data, and
    - determining if it is associated with that entity.
- This means that the computer must store some information about the entity.
- It also suggests that mechanisms for managing the data are required.

Authentication System

Authentication system consists of five components that are required for overall authentication process and they are as follows:

1. The set **A of authentication information** is the set of specific information with which entities prove their identities.
2. The set **C of complementary information** is the set of information that the system stores and uses to validate the authentication information.
3. The set **F of complementation functions** that generate the complementary information from the authentication information. That is, for $f \in F$, $f: A \rightarrow C$.

4. The set **L of authentication functions** that verify identity. That is, for $l \in L$, $l: A \times C \rightarrow \{true, false\}$.

5. The set **S of selection functions** that enable an entity to create or alter the authentication and complementary information.

EXAMPLE: A user authenticates himself by entering a password, which the system compares with the plaintext passwords stored online. Here, A is the set of strings making up acceptable passwords, C = A, F = { I }, and L = { eq }, where I is the identity function and eq is true if its arguments are the same and false if they are not.

Password

- Passwords are an example of an authentication mechanism based on what people know: the user supplies a password, and the computer validates it.
- If the password is the one associated with the user, that user's identity is authenticated. If not, the password is rejected and the authentication fails.
- Definition : A password is information associated with an entity that confirms the entity's identity.
- So passwords are an example of an authentication mechanism based on what people know and are usually:

-Sequence of characters, for e.g.: 10 digits (0,1,….9), a string of letters, etc generated randomly, by user, by computer with user input.

- Sequence of words, for e.g. pass-phrases.

Password Storage

- **Store as Plaintext**: If password file is compromised, all passwords are revealed.
- **Encipher File:** If encipherment keys are compromised then all passwords are revealed. There is need of deciphering process and encipherment keys to store in the memory.
- **Store One-way Hash of Password:** If the file is read, attacker must still guess passwords or invert the hash and this is very difficult task.

Attacking a Password System

- The simplest attack against a password-based system is to guess passwords.
- It may be like a *dictionary attack*; is the guessing of a password by repeated trial and error. The name of this attack comes from the list of words (a "dictionary") used for guesses.
- The dictionary may be a set of strings in random order or (more usually) a set of strings in decreasing order of probability of selection.
- If the complementary information and complementation functions are available, the dictionary attack takes each guess g and computes f(g) for each $f \in F$. If f(g) corresponds to the complementary information for entity E, then g authenticates E under f. This is a dictionary attack type 1. If either the complementary information or the complementation functions are unavailable, the authentication functions $l \in L$ may be used. If the guess g results in l returning true, g is the correct password. This is a dictionary attack type 2.

User Selection of Passwords:

For easy remembrance user may select easy to guess passwords called weak password.

 Based on account names, user names, computer names, place names

 Dictionary words (also reversed, odd capitalizations, control characters, —elite-speak, conjugations, swear words, Torah/Bible/Koran/… words)

 Too short, digits only, letters only

 License plates, acronyms, social security numbers

 Personal characteristics (pet names, nicknames, job characteristics, etc.)

 String PASSWORD and number 0123456789, 00000 etc.

- **Picking Good Passwords:** A password containing at least one digit, one letter, one punctuation symbol, and one control character is usually strong password.

Countering Password Guessing

- Guessing of passwords requires that access to the complement, the complementation functions, and the authentication functions be obtained.
- If none of these have changed by the time the password is guessed, then the attacker can use the password to access the system.
- To prevent this we must maximize the time need to guess password.
- Hence for security such password is expired before the required password guessing time period.

Password Aging

- **Password aging** is the requirement that a password be changed after some period of time has passed or after some event has occurred.
- There are problems involved in implementing password aging.
- Easier technique is to force users to change to a password. The second is providing notice of the need to change and a user-friendly method of changing passwords.
- Password aging is useless if a user can simply change the current password to the same thing. This defeats the goal of password aging.
- An alternative approach is based on time. In this implementation, the user must change the password to one other than the current password.
- The password cannot be changed for a minimum period of time. This prevents the rapid cycling of passwords.
- However, it also prevents the user from changing the password should it be compromised within that time period

Challenge-Response

- Passwords have the fundamental problem that they are reusable.
- If an attacker sees a password, she can later replay the password.
- The system cannot distinguish between the attacker and the legitimate user, and allows access.
- An alternative is to authenticate in such a way that the transmitted password changes each time.
- Then, if an attacker replays a previously used password, the system will reject it.

- Let user U desire to authenticate himself to system S. Let U and S have an agreed-on secret function f. A challenge-response authentication system is one in which S sends a random message m (the challenge) to U, and U replies with the transformation r = f(m) (the response). S validates r by computing it separately.
- This process is shown as:

  user → system: authentication request

  system → user: random message r (challenge)

  user → system: f(r) (response)

- Challenge-response algorithms are similar to the IFF (identification—friend or foe) techniques that military airplanes use to identify allies and enemies.

One-Time Password:

- An alternative is to authenticate in such a way that the transmitted password changes each time. Then, if an attacker replays a previously used password, the system will reject it.
- A one-time password is a password that is invalidated as soon as it is used.
- The problems in any one-time password scheme are the generation of random passwords and the synchronization of the user and the system.

Biometric

Biometrics is the automated measurement of biological or behavioral features that identify a person. Whenever the user accesses the system, the biometric authentication mechanism verifies the identity. Some of the biometrics characteristics are:

- **Fingerprints:**

➢ It can use either optical or electrical techniques.

➢ In this authentication mechanism fingerprints are converted into graph by using the points of the ridges of the print and then comparison with database is done.

➢ Graph matching is very hard problem so the fingerprint matching algorithms gives us approximated matches and hence measurement is imprecise.

- **Voices**:

- Authentication by voice, also called speaker verification or speaker recognition, involves recognition of a speaker's voice characteristics or verbal information verification.
- The verification (speaker independent) uses statistical techniques to test the hypothesis that the speaker's identity is as claimed.
- The recognition checks the contents of answers and it is speaker independent.

- **Eyes:**

- ➢ Authentication by eye characteristics uses the iris and the retina.
- ➢ Patterns within the iris are unique for each person. Hence, one verification approach is to compare the patterns statistically and ask whether the differences are random.
- ➢ A second approach is to correlate the images using statistical tests to see if they match.
- ➢ Retinal scans rely on the uniqueness of the patterns made by blood vessels at the back of the eye.
- ➢ This requires a laser beaming onto the retina, which is highly intrusive. This method is typically used only in the most secure facilities.

- **Faces:**

- Face recognition consists of several steps.
- First, the face is located. If the user places her face in a predetermined position the problem becomes bit easier.
- However, facial features such as hair and glasses may make the recognition harder.
- Techniques for doing this include the use of neural networks and templates. The resulting image is then compared with the relevant image in the database.
- The correlation is affected by the differences in the lighting between the current image and the reference image, by distortion, by "noise," and by the view of the face.
- The correlation mechanism must be "trained." Several different methods of correlation have been used, with varying degrees of success.
- An alternative approach is to focus on the facial features such as the distance between the nose and the chin, and the angle of the line drawn from one to the other.

- **Keystroke Dynamics:**

➢ Keystroke dynamics requires a signature based on keystroke intervals, keystroke pressure, keystroke duration, and where the key is struck (on the edge or in the middle).

➢ This signature is believed to be unique in the same way that written signatures are unique.

➢ Keystroke recognition can be both static and dynamic.

➢ Static recognition is done once, at authentication time, and usually involves typing of a fixed or known string. Once authentication has been completed, an attacker can capture the connection without detection.

➢ Dynamic recognition is done throughout the session, so the above-mentioned attack is not feasible. However, the signature must be chosen so that variations within an individual's session do not cause the authentication to fail.

Location

- If you know where user is, validate identity by seeing if person is where the user is. It requires special-purpose hardware to locate user.
- **GPS (Global Positioning System) Satellites:** The physical location of the user which changes with respect to a time is found out by GPS satellite system for obtaining a signature. The signature is transmitted to authenticate the user.
- **Location Signature Sensor (LSS):** Host also obtains a similar signature of the user. If the signatures disagree, the authentication fails. If the LSS is stolen, the thief would have to log in from an authorized geographic location.

Multiple Methods

- It is a combination of two or more authentication approaches to make the system more secure.
- Example: Authenticating by location generally uses special-purpose hardware. Although the key feature of this technique is physical location, without the LSS it will not work. It combines location with a token or with what one possesses like password or smart card etc.

Authentication Protocol

- In this section, we focus on two general areas (mutual authentication and one-way authentication) and examine some of the implications of authentication techniques in both.

Mutual Authentication (Symmetric and Public Key)

- An important application area is that of mutual authentication protocols.
- Such protocols enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys.
- Central to the problem of authenticated key exchange are two issues: confidentiality and timeliness.
- To prevent masquerade and to prevent compromise of session keys, essential identification and session key information must be communicated in encrypted form.
- This requires the prior existence of secret or public keys that can be used for this purpose.
- The second issue, timeliness, is important because of the threat of message replays.
- Such replays, at worst, could allow an opponent to compromise a session key or successfully impersonate another party.
- At minimum, a successful replay can disrupt operations by presenting parties with messages that appear genuine but are not.

One-Way Authentication (Symmetric, Public Key)

- The recipient assures that a message is from the alleged sender. Sender & receiver are not in communications at same time (e.g. email)
    - have header in clear so can be delivered by email system
    - may want contents of body protected & sender authenticated
- One application for which encryption is growing in popularity is electronic mail (e-mail).
- The very nature of electronic mail, and its chief benefit, is that it is not necessary for the sender and receiver to be online at the same time.
- Instead, the e-mail message is forwarded to the receiver's electronic mailbox, where it is buffered until the receiver is available to read it.
- The "envelope" or header of the e-mail message must be in the clear, so that the message can be handled by the store-and-forward e-mail protocol, such as the Simple Mail Transfer Protocol (SMTP).
- However, it is often desirable that the mail-handling protocol not require access to the plaintext form of the message, because that would require trusting the mail-handling mechanism.

- Accordingly, the e-mail message should be encrypted such that the mail-handling system is not in possession of the decryption key. A second requirement is that of authentication. Typically, the recipient wants some assurance that the message is from the alleged sender.

Digital Signature

- The most important development from the work on public-key cryptography is the digital signature.
- 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.
- It must have the following properties:
  - It must verify the author and the date and time of the signature
  - It must to authenticate the contents at the time of the signature
  - It must be verifiable by third parties, to resolve disputes

- Thus, the digital signature function includes the authentication function.
- A digital signature is a construct that authenticates both the origin and contents of a message in a manner that is provable to a disinterested third party.
- Suppose that Bob wants to send a message to Alice, and although it is not important that the message be kept secret, he wants Alice to be certain that the message is indeed from him. In this case, Bob uses his own private key to encrypt the message. When Alice receives the ciphertext, she finds that she can decrypt it with Bob‘s public key, thus proving that the message must have been encrypted by Bob.

- No one else has Bob‘s private key, and therefore no one else could have created a ciphertext that could be decrypted with Bob‘s public key. Therefore, the entire encrypted message serves as a digital signature. In addition, it is impossible to alter the message without access to Bob‘s private key, so the message is authenticated both in terms of source and in terms of data integrity.

- Any digital signature involves following operations:

1. Key generation: It generates private public key used in signature.

2. Signing Algorithm: It is the process of creating signature with keys.

3. Signature Verification: The signature should be verifiable by third party. So any signature scheme needs to have signature verification process.

How it works?

- The use of digital signatures usually involves two processes, one performed by the signer and the other by the receiver of the digital signature:
- Digital signature creation uses a hash result derived from and unique to both the signed message and a given private key. For the hash result to be secure, there must be only a negligible possibility that the same digital signature could be created by the combination of any other message or private key.
- Digital signature verification is the process of checking the digital signature by reference to the original message and a given public key, thereby determining whether the digital signature was created for that same message using the private key that corresponds to the referenced public key.

Example

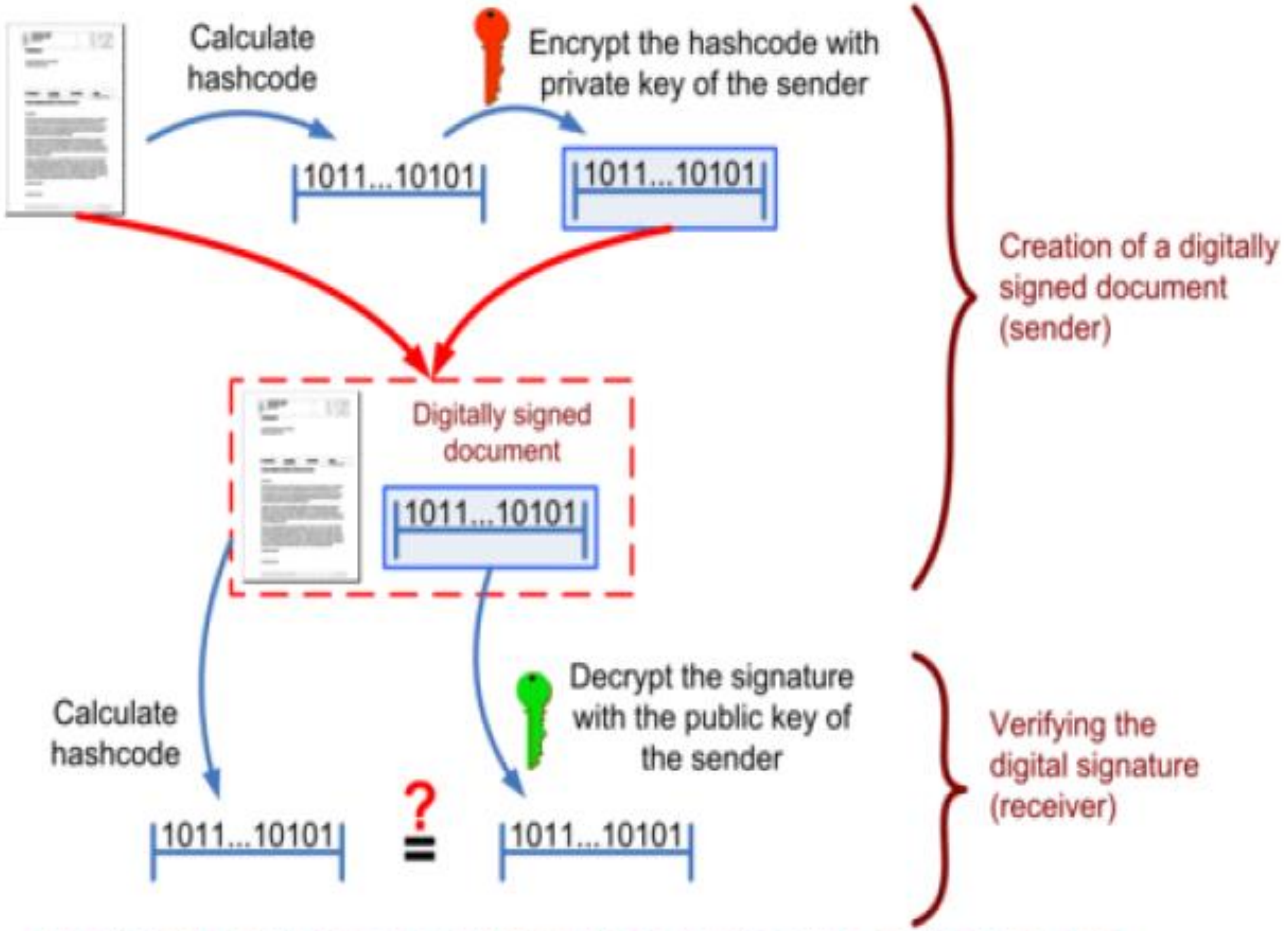
- Assume you were going to send the draft of a contract to your lawyer in another town. You want to give your lawyer the assurance that it was unchanged from what you sent and that it is really from you.

1. You copy-and-paste the contract (it's a short one!) into an e-mail note.
2. Using special software, you obtain a message hash (mathematical summary) of the contract.
3. You then use a private key that you have previously obtained from a public-private key authority to encrypt the hash.
4. The encrypted hash becomes your digital signature of the message. (Note that it will be different each time you send a message.)

- At the other end, your lawyer receives the message.

1. To make sure it's intact and from you, your lawyer makes a hash of the received message.
2. Your lawyer then uses your public key to decrypt the message hash or summary.
3. If the hashes match, the received message is valid.

Creating and verifying a digital signature


Calculate hashcode
Encrypt the hashcode with private key of the sender
1011...10101
1011...10101
Creation of a digitally signed document (sender)
Digitally signed document
1011...10101
Calculate hashcode
Decrypt the signature with the public key of the sender
1011...10101
?
=
1011...10101
Verifying the digital signature (receiver)

If the calculated hashcode does not match the result of the decrypted signature, either the document was changed after signing, or the signature was not generated with the private key of the alleged sender.

Benefits

These are common reasons for applying a digital signature to communications:

- Authentication

Although messages may often include information about the entity sending a message, that information may not be accurate. Digital signatures can be used to authenticate the source of messages. When ownership of a digital signature secret key is bound to a specific user, a valid signature shows that the message was sent by that user. The importance of high confidence in sender authenticity is especially obvious in a financial context. For example, suppose a bank's branch office sends instructions to the central office requesting a change in the balance of an account. If the central office is not convinced that such a message is truly sent from an authorized source, acting on such a request could be a grave mistake.

- Integrity

In many scenarios, the sender and receiver of a message may have a need for confidence that the message has not been altered during transmission. Although encryption hides the contents of a message, it may be possible to change an encrypted message without understanding it. (Some encryption algorithms, known as nonmalleable ones, prevent this, but others do not.) However, if a message is digitally signed, any change in the message will invalidate the signature. Furthermore, there is no efficient way to modify a message and its signature to produce a new message with a valid signature, because this is still considered to be computationally infeasible by most cryptographic hash functions.

Drawbacks

Despite their usefulness, digital signatures do not alone solve all the problems we might wish them to.

- Non-repudiation

In a cryptographic context, the word repudiation refers to the act of disclaiming responsibility for a message. A message's recipient may insist the sender attach a signature in order to make later repudiation more difficult, since the recipient can show the signed message to a third party (eg, a court) to reinforce a claim as to its signatories and integrity. However, loss of control over a user's private key will mean that all digital signatures using that key, and so allegedly 'from' that user, are suspect. Nonetheless, a user cannot repudiate a signed message without repudiating their signature key.

Types

- **The Direct Digital Signature**

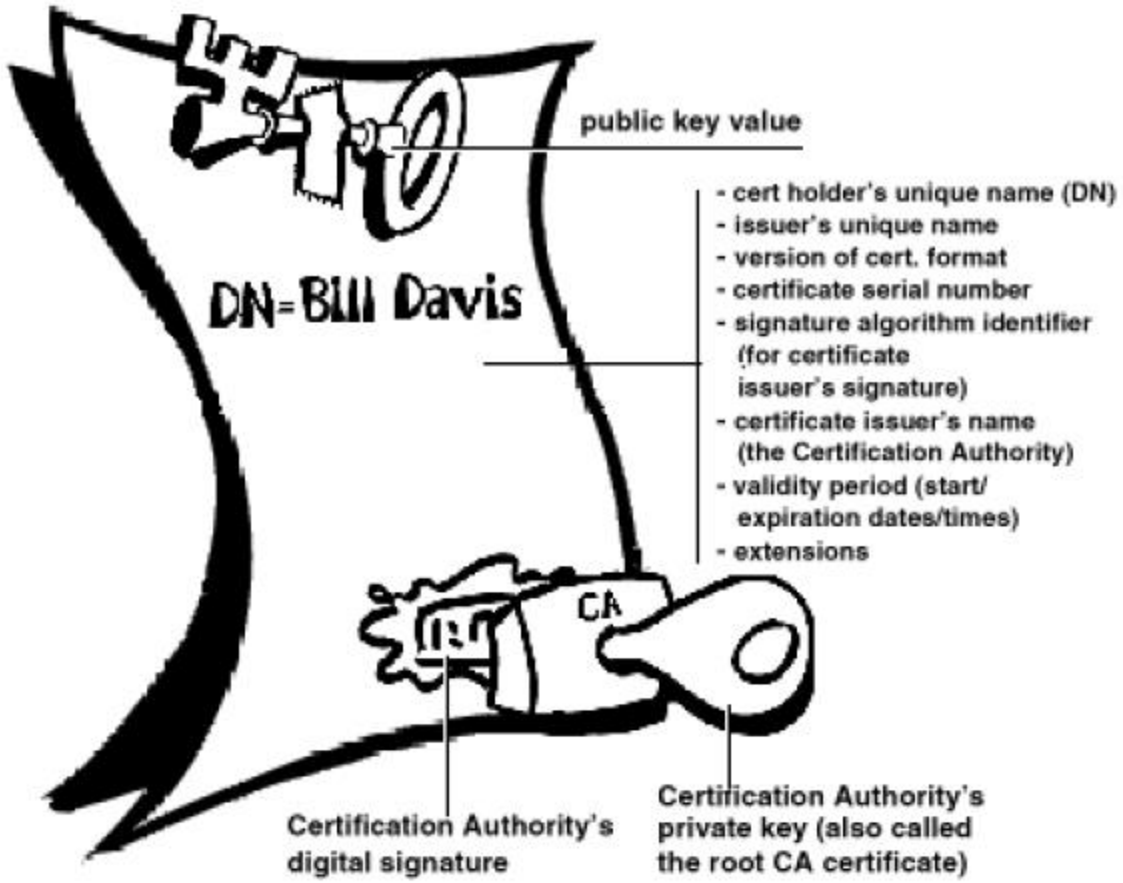
  Understanding a direct digital signature begins by recognizing there are only two parties involved in the passing of the signed information: the sender and the receiver. Direct digital signatures only require these two entities because the receiver of the data (digital signature) knows the public key used by the sender. And the sender of the signature trusts the receiver not to alter the document in any way.

- **The Arbitrated Digital Signature**

  Implementing an arbitrated digital signature invites a third party into the process called a "trusted arbiter." The role of the trusted arbiter is usually twofold: first this independent third party verifies the integrity of the signed message or data. Second, the trusted arbiter dates, or time-stamps, the document, verifying receipt and the passing on of the signed document to its intended final destination.

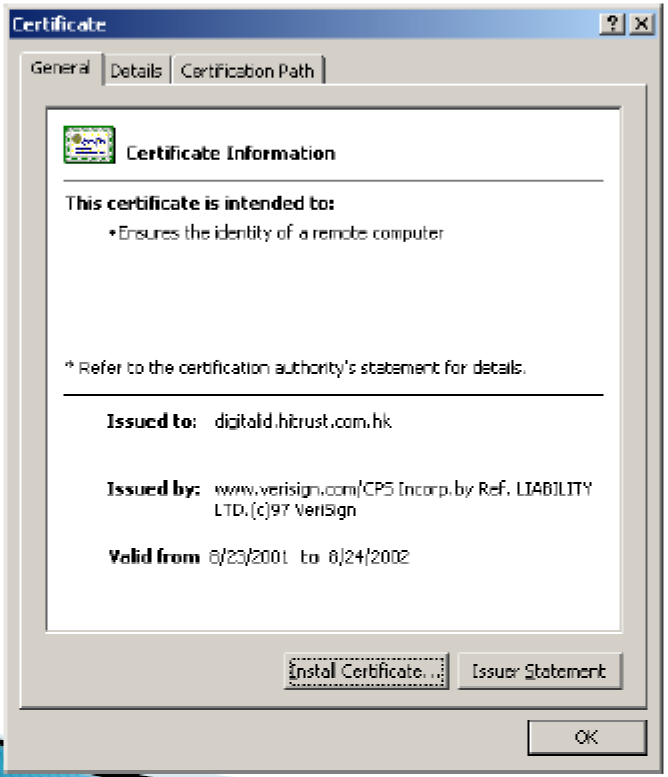
Digital Certificates

- Digital Certificate is a data with digital signature from one trusted Certification Authority (CA).
- This data contains:
    - Who owns this certificate
    - Who signed this certificate
    - The expired date
    - User name & email address
- A certificate is a token that binds an identity to a cryptographic key.
- When B wants to communicate with A, B obtains A's certificate CA. for e.g. Create token (message) containing: Identity of principal A, Corresponding public key, Issue timestamp and Other information like identity of signer signed by trusted authority C as CA = { eA || A || T }dC


public key value
- cert holder's unique name (DN)
- issuer's unique name
- version of cert. format
- certificate serial number
- signature algorithm identifier (for certificate issuer's signature)
- certificate issuer's name (the Certification Authority)
- validity period (start/ expiration dates/times)
- extensions
DN=Bill Davis
CA
Certification Authority's digital signature
Certification Authority's private key (also called the root CA certificate)

Elements of Digital Certificate

- A Digital ID typically contains the following information:
  - ◦Your public key, Your name and email address
  - ◦Expiration date of the public key, Name of the CA who issued your Digital ID



Certificate Information

This certificate is intended to:

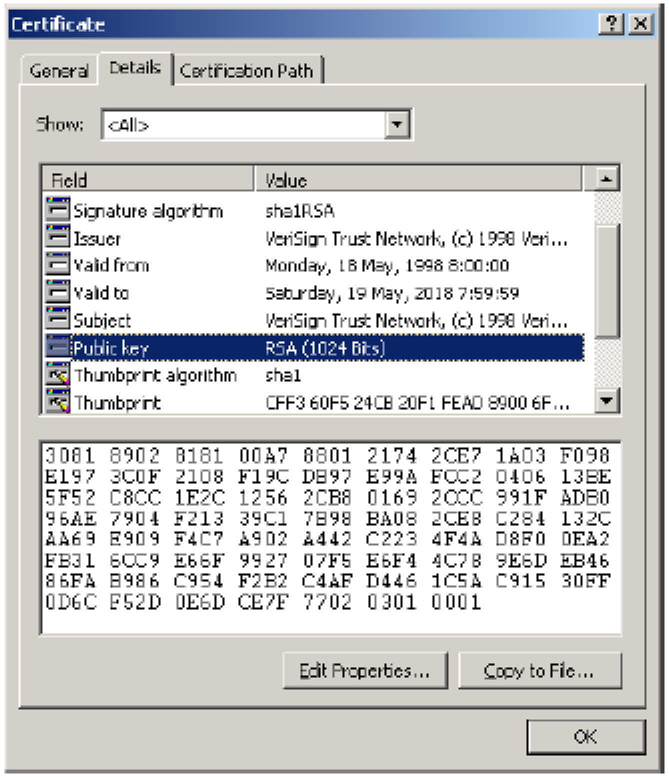
• Ensures the identity of a remote computer

* Refer to the certification authority's statement for details.

Issued to: digitalid.hitrust.com.hk

Issued by: www.verisign.com/CPS Incorp.by Ref. LIABILITY LTD.(c)97 VeriSign

Valid from 8/23/2001 to 8/24/2002



Show: <All>

Field	Value
Signature algorithm	sha1RSA
Issuer	VeriSign Trust Network, (c) 1998 Veri...
Valid from	Monday, 18 May, 1998 8:00:00
Valid to	Saturday, 19 May, 2018 7:59:59
Subject	VeriSign Trust Network, (c) 1998 Veri...
Public key	RSA (1024 Bits)
Thumbprint algorithm	sha1
Thumbprint	CFF3 60F5 24CB 20F1 FEAD 8900 6F...

```
3081 8902 8181 00A7 8801 2174 2CE7 1A03 F098
E197 3C0F 2108 F19C DB97 E99A FCC2 0406 13BE
5F52 C8CC 1E2C 1256 2CB8 0169 2CCC 991F ADB0
96AE 7904 F213 39C1 7B98 BA08 2CEB C284 132C
AA69 E909 F4C7 A902 A442 C223 4F4A D8F0 0EA2
FB31 6CC9 E66F 9927 07F5 E6F4 4C78 9E6D EB46
86FA B986 C954 F2B2 C4AF D446 1C5A C915 30FF
0D6C F52D 0E6D CE7F 7702 0301 0001
```

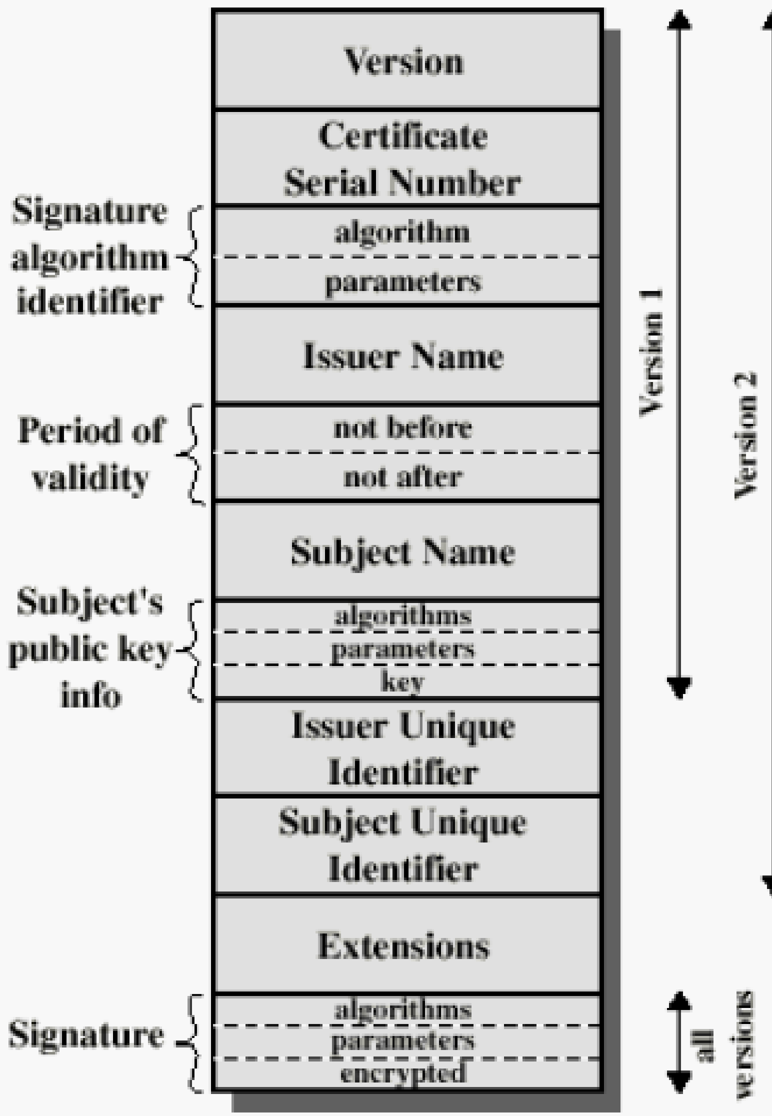
Certification Authority (CA)

- A trusted agent who certifies public keys for general use (Corporation or Bank).
    - User has to decide which CAs can be trusted.
- The model for key certification based on friends and friends of friends is called “Web of Trust”.
    - The public key is passing from friend to friend.
    - Works well in small or high connected worlds.

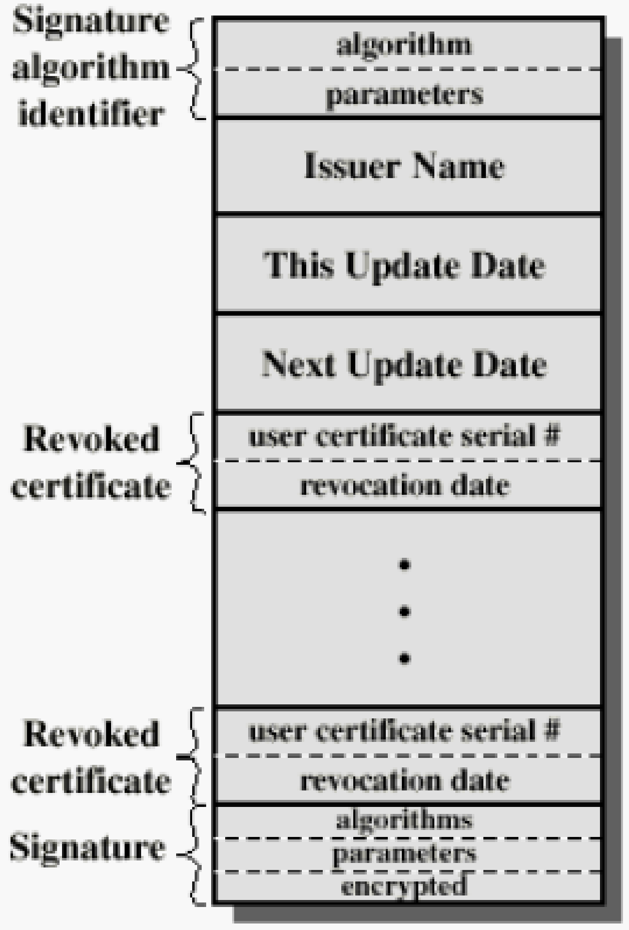
X.509 Certificates

- ITU-T recommendation X.509 is part of the X.500 series of recommendations that define a directory service.
- The directory is, in effect, a server or distributed set of servers that maintains a database of information about users.
- The information includes a mapping from user name to network address, as well as other attributes and information about the users.
- X.509 defines a framework for the provision of authentication services by the X.500 directory to its users.
- The directory may serve as a repository of public-key certificates.
- Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority

- In addition, X.509 defines alternative authentication protocols based on the use of public-key certificates.
- X.509 is an important standard because the certificate structure and authentication protocols defined in X.509 are used in a variety of contexts.
- 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.



(a) X.509 Certificate



(b) Certificate Revocation List

- **Version:** The version number 1, 2, or 3.
- **Serial Number:** This must be unique among the certificates issued by this issuer.
- **Signature Algorithm Identifier:** Identifies algorithm and any parameters used to sign the certificate.
- **Issuer's Distinguished Name**: Name that uniquely identifies the issuer.
- **Validity Interval:** Gives the times at which the certificate becomes valid and expires.
- **Subject's Distinguished Name:** Unique name that identifies subject to whom the certificate is issued.
- **Subject's Public Key Information:** Identifies the algorithm, its parameters, and the subject's public key.

- **Issuer's Unique Identifier (Version 2 and 3 certificates only):** Under some circumstances, issuer Distinguished Names may be recycled (for example, when the Distinguished Name refers to a role, or when a company closes and a second company with the same Distinguished Name opens). This field allows the issuer to disambiguate among entities with the same issuer name.
- **Subject's Unique Identifier (Version 2 and 3 certificates only):** This field is like field 8, but for the subject.
- **Extensions (Version 3 certificates only):** X.509v3 defines certain extensions in the areas of key and policy information, certification path constraints, and issuer and subject information. For example, if an issuer has multiple certification keys, the "authority key identifier" allows the certificate to indicate which key should be used. The "basic constraints" extension indicates if the certificate holder can issue certificates.

- **Signature:** This field identifies the algorithm and parameters used to sign the certificate, followed by the signature (an enciphered hash of fields 1 to 10) itself.

X.509 Certificate Validation

- Obtain issuer‘s public key - the one for the particular signature algorithm
- Decipher signature - gives hash of certificate
- Re-compute hash from certificate and compare - if they differ, there‘s a problem
- Check interval of validity - this confirms that certificate is current

X.509 Certificate Authentication Service

- Distributed set of servers that maintains a database about users.
- Each certificate contains the public key of a user and is signed with the private key of a CA.
- Is used in IP Security, SET.
- RSA is recommended to use.

Kerberos

- trusted key server system from Massachusetts Institute of Technology (MIT)
- provides centralized private-key third-party authentication in a distributed network
- Provides a centralized authentication server to authenticate users to servers and servers to users.
- Relies on conventional encryption, making no use of public-key encryption
- Two versions: version 4 and 5 where version 4 makes use of DES

An Authentication Service:

- Based on client-server model (user and server provider)
- Mutual authentication support: between user and server

Basics

- Based on KDC-based symmetric key
- Use “ticket” to distribute the key
- Use “authenticator" to prove the identity of a user

Idea

- User u authenticates to Kerberos server and obtains ticket Tu,TGS for ticket granting service (TGS).
- For using service s by u: User sends authenticator Au, ticket Tu,TGS to TGS asking for ticket for service. TGS sends ticket Tu,s to user and user sends Au, Tu,s to server as request to use s.

Ticket

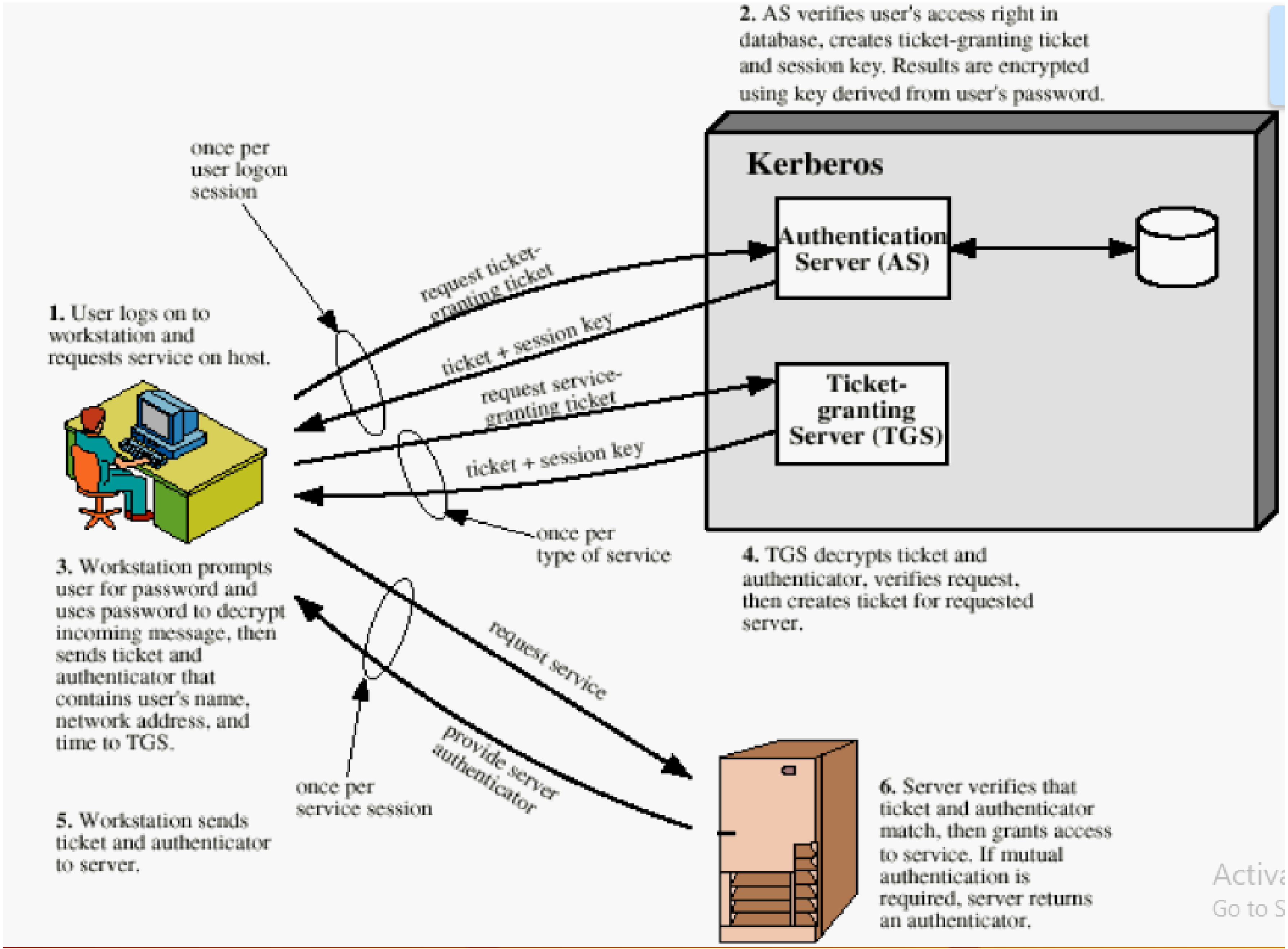
- Ticket issued to user u for service s **Tu,s = s || { u || u's address || valid time || ku,s } ks**

  where ku,s is session key for user and service

Authenticator

- Authenticator user u generates for service s **Au,s = { u || generation time || kt } ku,s**

  where kt is alternate session key


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

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.

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

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.

A Simple Authentication Dialogue

- (1) $C \rightarrow AS : ID_C \,||\, P_C \,||\, ID_V$
  - C = client
  - AS = authentication server
  - ID_C = identifier of user on C
  - P_C = password of user on C
  - ID_V = identifier of server V
  - C asks user for the password
  - AS checks that user supplied the right password

Message 2

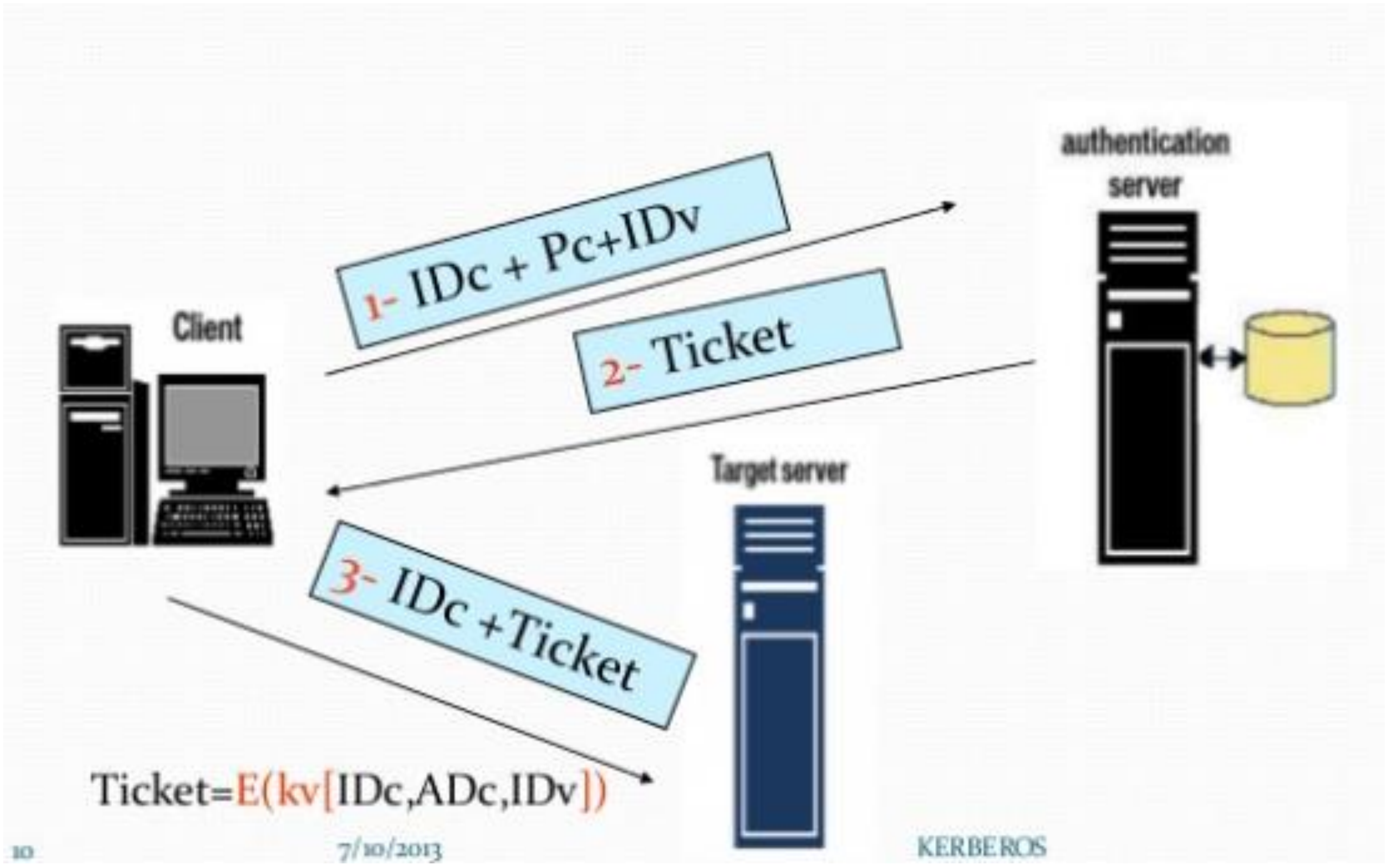
- (2) AS -> C : Ticket
- Ticket = $E_{K(V)} [ID_C || AD_C || ID_V]$
  - K(V) = secret encryption key shared by AS and V
  - AD_C = network address of C
  - Ticket cannot be altered by C or an adversary

Message 3

- (3) C -> V: ID_C || Ticket
  - Server V decrypts the ticket and checks various fields
  - AD_C in the ticket *binds* the ticket to the network address of C
  - However this authentication scheme has problems

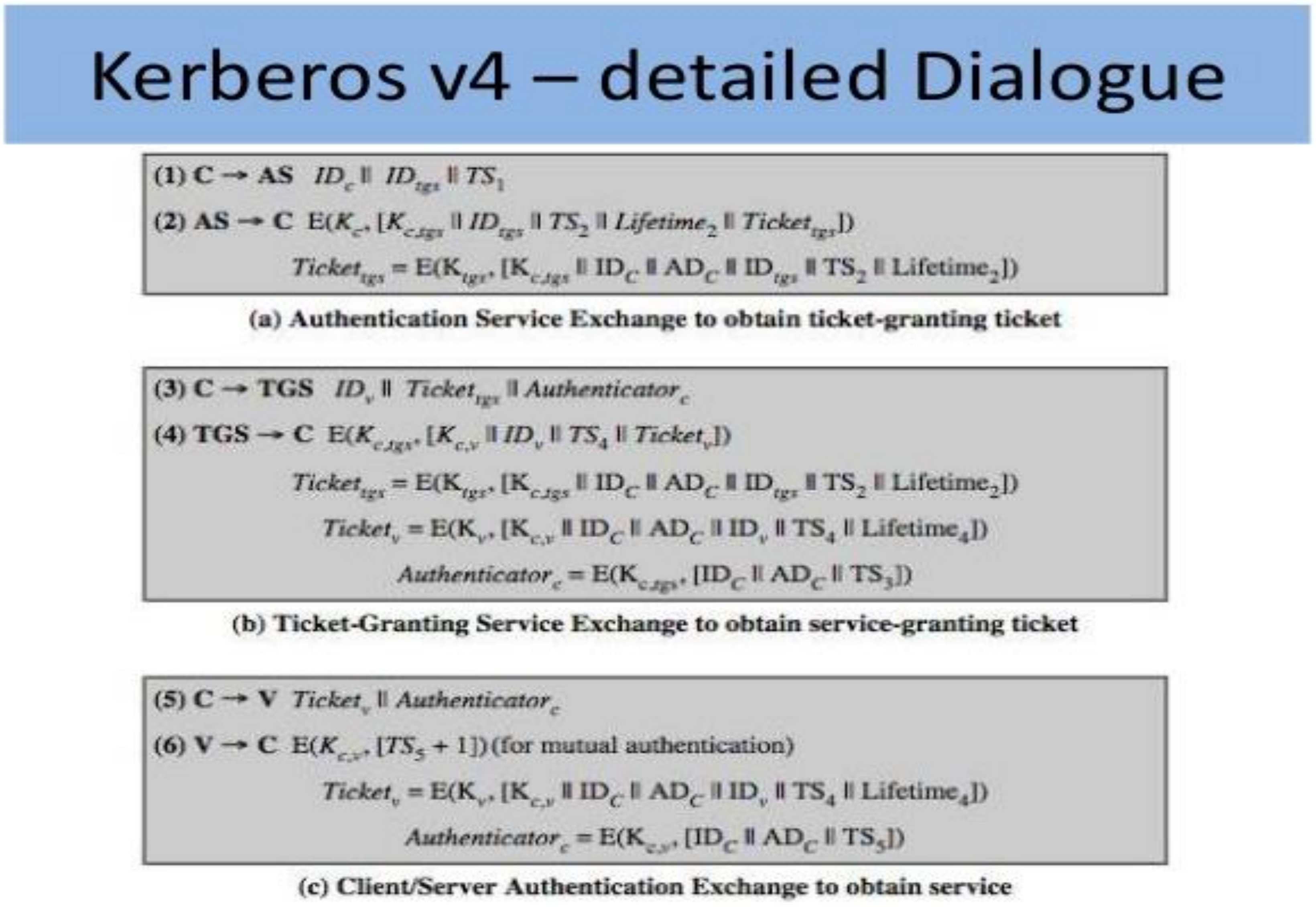
Problems

- Each time a user needs to access a different service he/she needs to enter their password
    - ◦Read email several times
- ◦Print, mail, or file server
- ◦Assume that each ticket can be used only once (otherwise open to replay attacks)
- Password sent in the clear



Ticket=E(kv[IDc,ADc,IDv])

Authentication Dialogue II



Kerberos v4 – detailed Dialogue

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

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

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

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

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

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

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

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

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

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

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

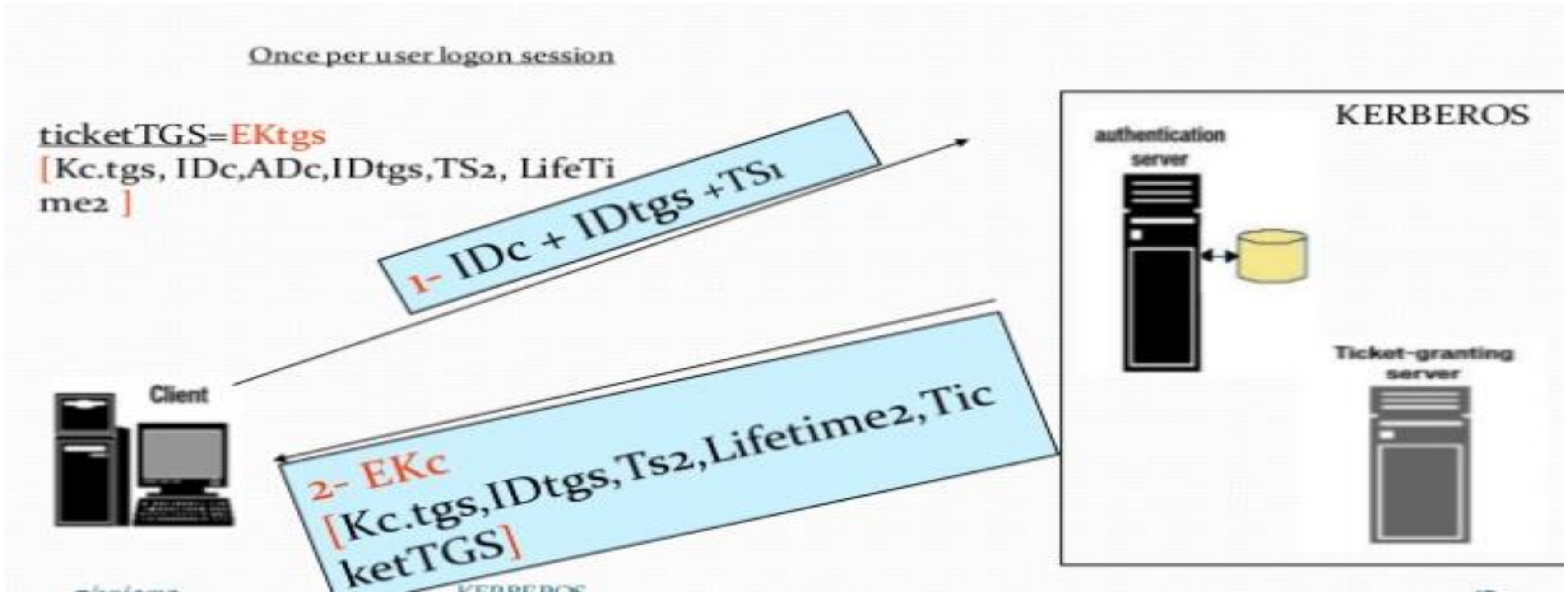
(6) **V → C** $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

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

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

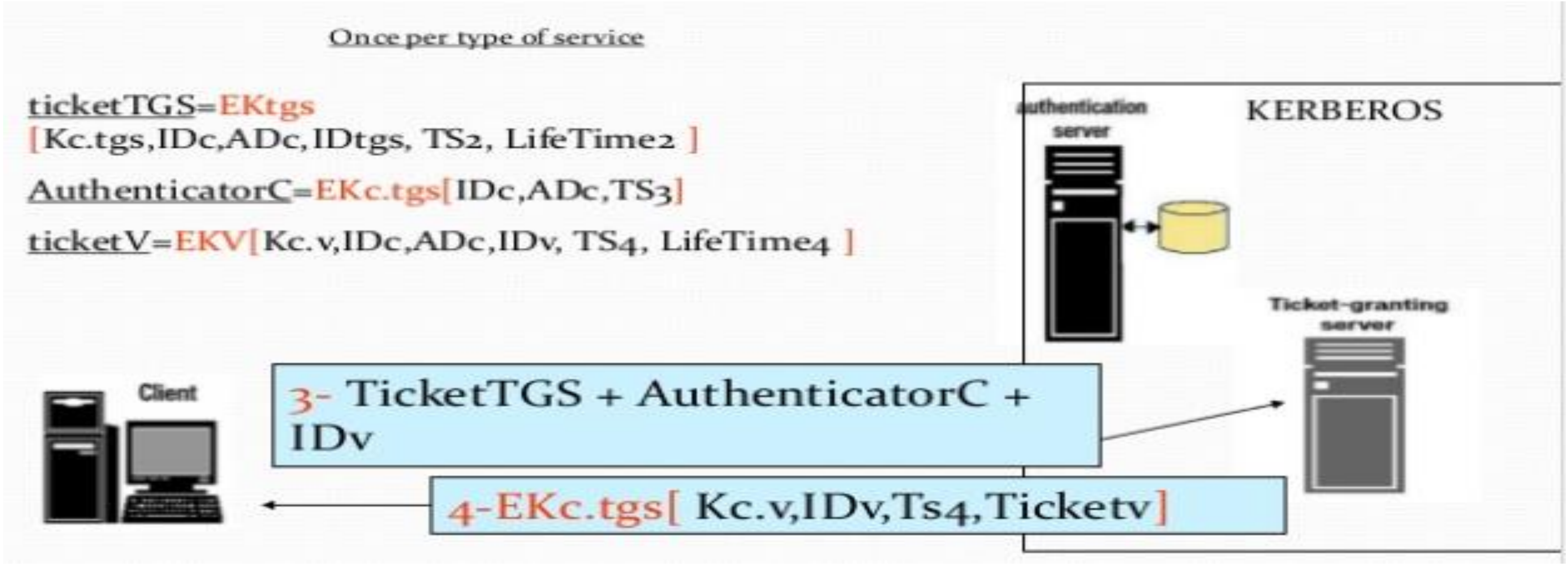
(c) Client/Server Authentication Exchange to obtain service

- C Client
- AS authentication server
- V server
- IDc identifier of user on C
- IDv identifier of V
- Kv secret encryption key shared by AS an V
- Pc password of user on C
- ADc network address of C
- TS timestamp
- || concatenation



Once per user logon session

ticketTGS=EKtgs [Kc.tgs, IDc,ADc,IDtgs,TS2, LifeTime2]



Once per type of service

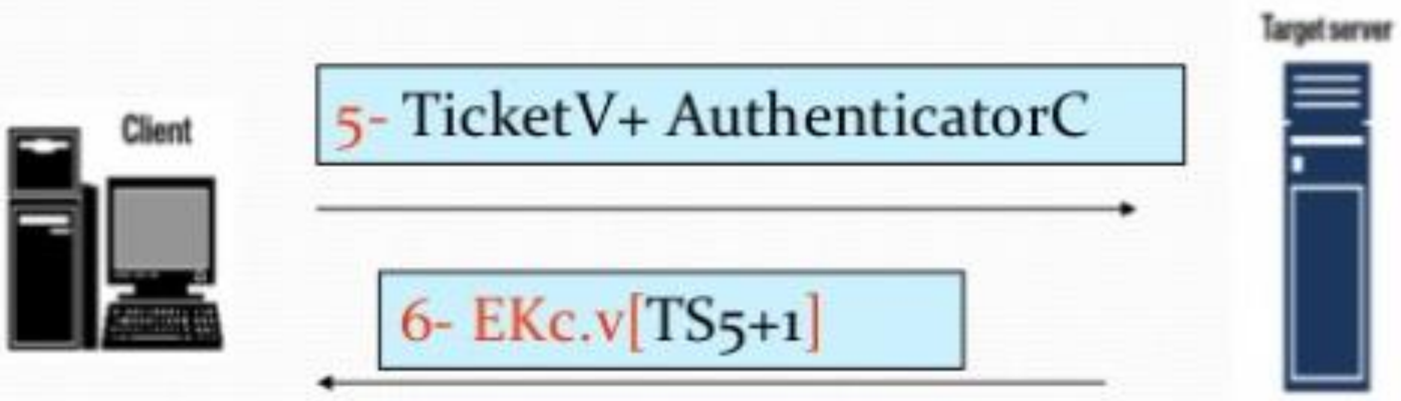
ticketTGS=EKtgs [Kc.tgs,IDc,ADc,IDtgs, TS2, LifeTime2]

AuthenticatorC=EKc.tgs[IDc,ADc,TS3]

ticketV=EKV[Kc.v,IDc,ADc,IDv, TS4, LifeTime4]

3- TicketTGS + AuthenticatorC + IDv
4-EKc.tgs[Kc.v,IDv,Ts4,Ticketv]
Client
KERBEROS
authentication server
Ticket-granting server

Once per service session



TicketV=EKv [Kv.c, IDc, ADc, IDv, TS4, Lifetime4]

AuthenticatorC=EKc.v [IDc,ADc,TS5]

Problems

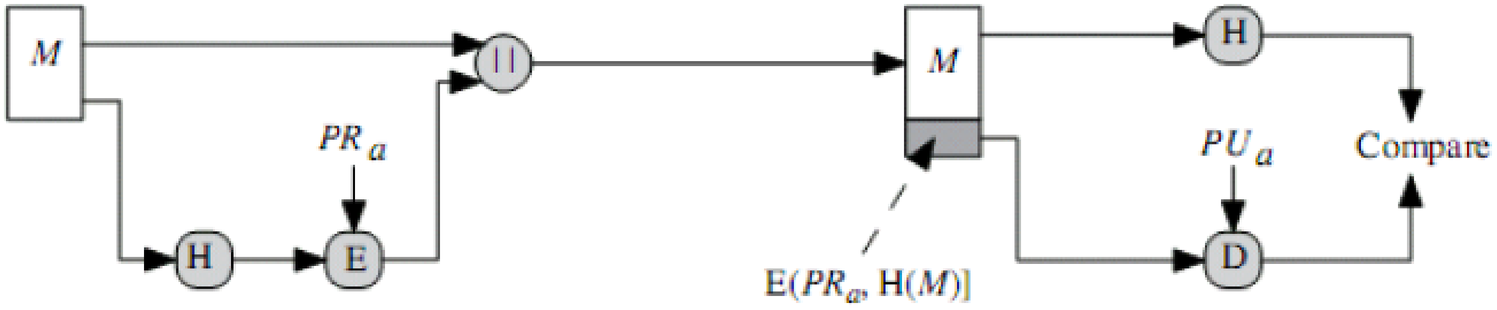
- Lifetime associated with the ticket-granting ticket
- If too short - repeatedly asked for password
- If too long - greater opportunity to replay
- The threat is that an opponent will steal the ticket and use it before it expires

Digital Signature Standard (DSS)
Digital Signature Algorithm (DSA)

- 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).
- The DSS uses an algorithm that is designed to provide only the digital signature function.
- Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique.

Digital Signature using RSA

- In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length.
- This hash code is then encrypted using the sender's private key to form the signature.
- Both the message and the signature are then transmitted.
- The recipient takes the message and produces a hash code.
- The recipient also decrypts the signature using the sender's public key.
- If the calculated hash code matches the decrypted signature, the signature is accepted as valid.
- Because only the sender knows the private key, only the sender could have produced a valid signature.

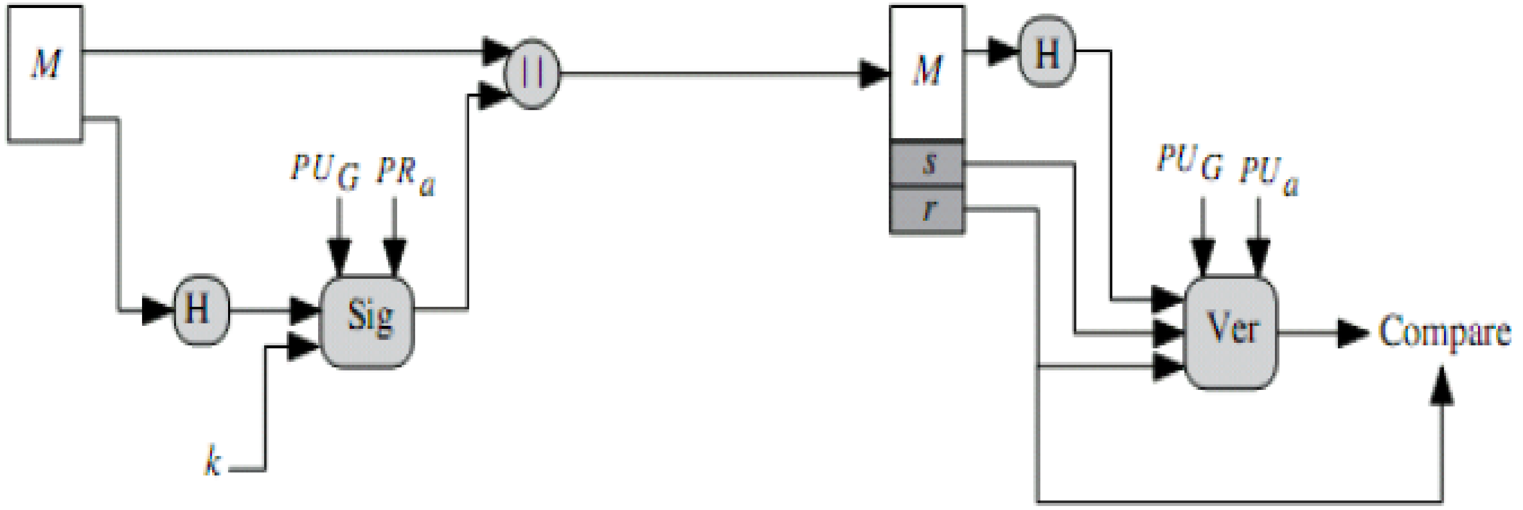


(a) RSA Approach

RSA Based Digital Signature

- ❖ To generate RSA signature keys, one simply generates an RSA key pair.
- ❖ To sign a message m, the signer computes $\sigma = m^d \bmod n$. To verify, the receiver checks that $\sigma^e = m \bmod n$. where (e, n) is the public key and (d, n) is the private key.

Digital Signature using DSS or DSA

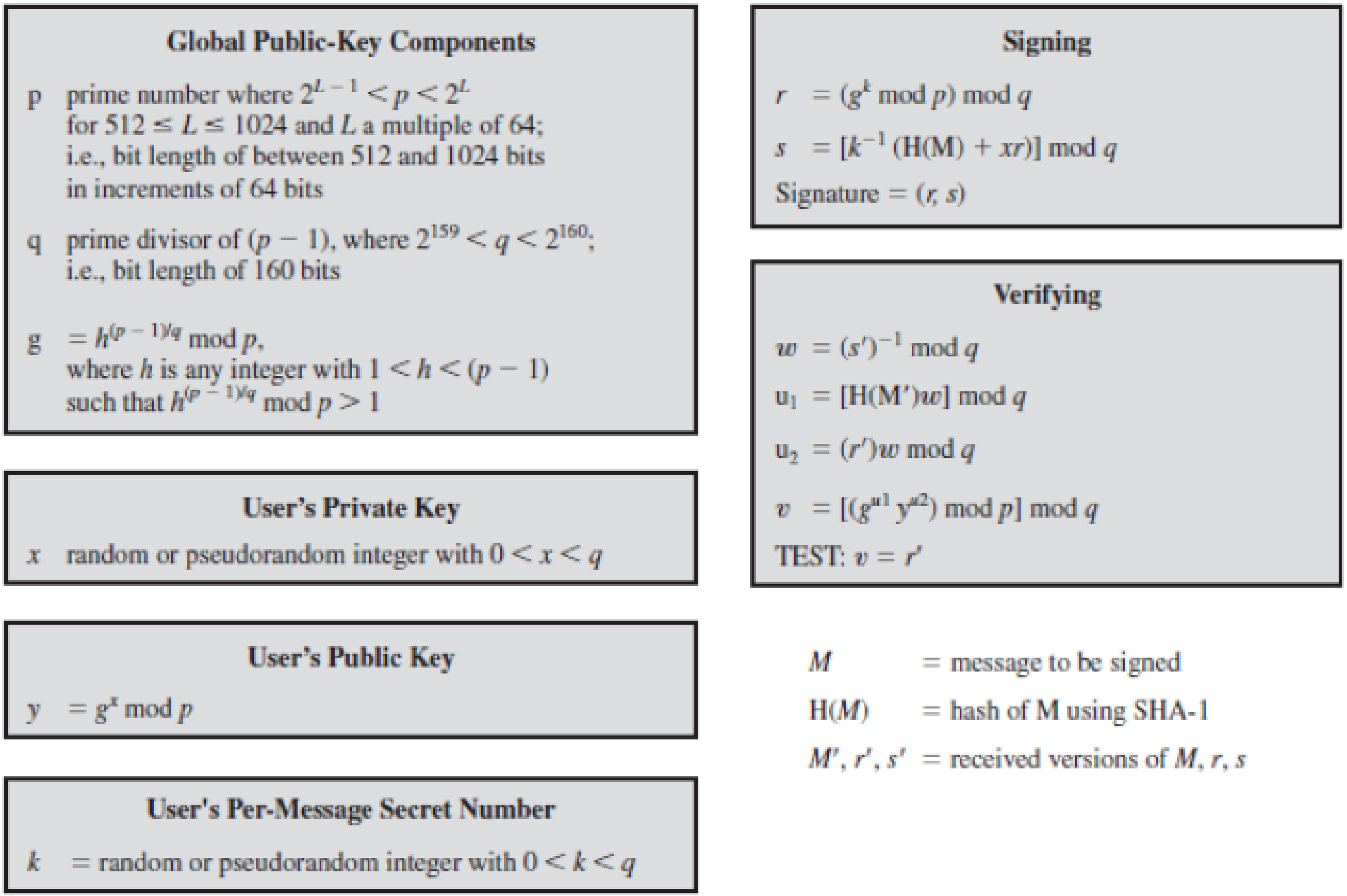


(b) DSS Approach

- The DSS approach also makes use of a hash function.
- The hash code is provided as input to a signature function along with a random number *k* generated for this particular signature.
- The signature function also depends on the sender's private key (PR a) and a set of parameters known to a group of communicating principals.
- We can consider this set to constitute a global public key (PU_G).
- The result is a signature consisting of two components, labeled s and r.
- At the receiving end, the hash code of the incoming message is generated.
- This plus the signature is input to a verification function. The verification function also depends on the global public key as well as the sender's public key (PUa), which is paired with the sender's private key. The output of the

- The verification function also depends on the global public key as well as the sender's public key (PUa), which is paired with the sender's private key.
- The output of the verification function is a value that is equal to the signature component r if the signature is valid.
- The signature function is such that only the sender, with knowledge of the private key, could have.

Algorithm



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 $h^{(p-1)/q} \bmod p > 1$

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 $0 < k < q$

Signing

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

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

Signature = (r, s)

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

DSS (DSA) Vs. RSA Approaches

- Both DSS and RSA are based on public key technique (Private and public keys are used. Private Key is used on sender's side and public key on the receiver side)
- DSS provides digital signatures but does not provide key exchange and encryption. RSA provides digital signatures, encryption and key exchange.
- DSA differs from RSA in how the message signature is generated and validated.
- RSA signatures encrypt the message hash with the private key to create a signature, which is then verified by being decrypted with the public key to compare to a recreated hash value.

- DSA signatures use the message hash, global public values, private key & random k to create a 2 part signature (s,r). This is verified by computing a function of the message hash, public key, r and s, and comparing the result with r.

DSS approach

- The hash function is applied to the message.
- A hash code is produced.
- This hash code, along with a random number generated is given as input to the signature function.
- The sender's private key and a global public key is also given as input to the signature function.
- The output of this signature function is concatenated with the original message and sent.
- During reception, the hash function is generated.
- Along with this, the signature is sent to a verification function.
- The verification function depends on the sender's public key & global public key.
- This is compared with the signature component of the incoming message.
- If they both match, the incoming message is authenticated.

RSA Approach

- Message is input to a hash function.
- Hash value thus generated is encrypted using the sender's private key.
- This encrypted content serves as the signature. It is concatenated with the original message and transmitted.
- Receiver generates a hash code and decrypts the signature component using the sender's public key.
- If the generated hash code and the decrypted value matches, the message is authenticated.

Mutual Authentication Protocol Using Symmetric Encryption

- A two-level hierarchy of symmetric encryption keys can be used to provide confidentiality for communication in a distributed environment.
- usually with a trusted Key Distribution Center (KDC)

  □ each party shares own master key with KDC

  □ KDC generates session keys used for connections between parties

  □ master keys used to distribute these to them

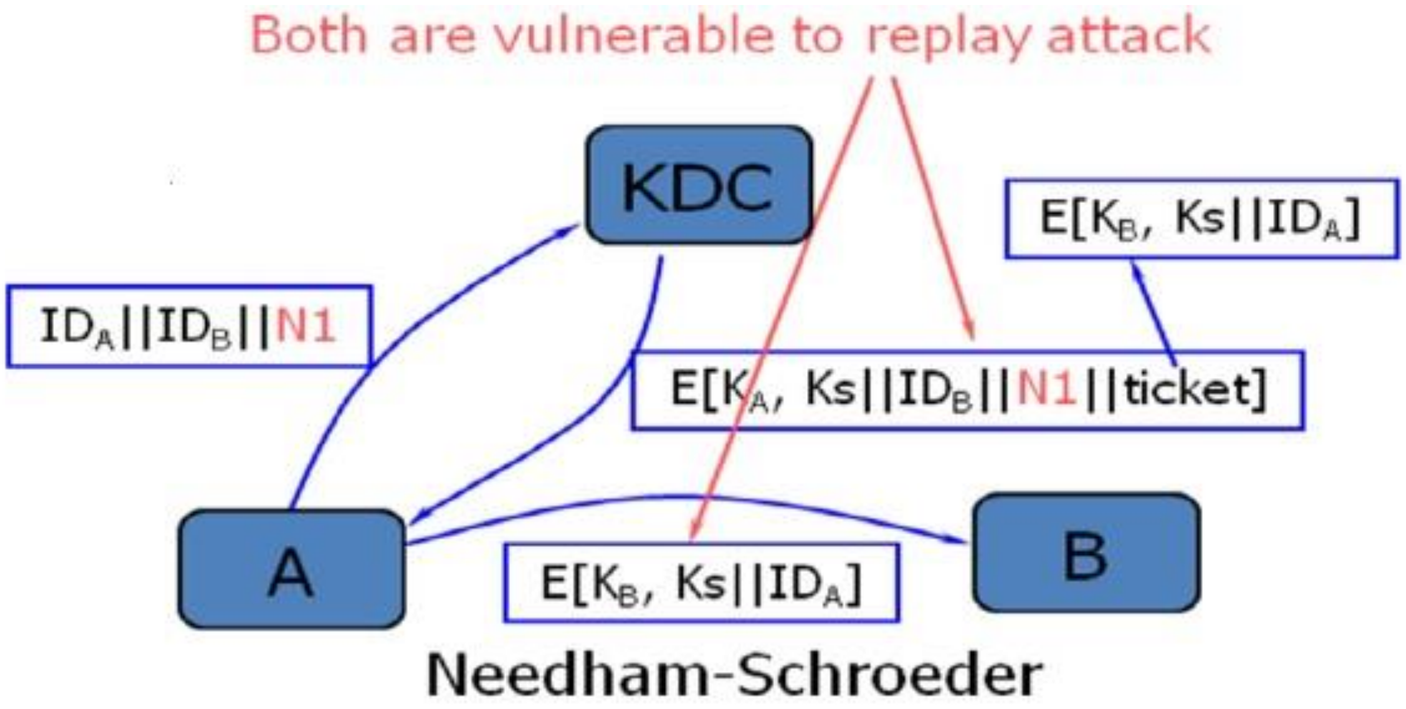
Example Needham-Schroeder Protocol

Needham-Schroeder Protocol

- Original third-party key distribution protocol for session between A B mediated by KDC
- The protocol overview is:

1. $A \rightarrow KDC: ID_A \| ID_B \| N1$ (A sends message to KDC to identify herself and B)
2. $KDC \rightarrow A: EKa[Ks \| ID_B \| N1 \| EK_b[Ks\|ID_A]\ ]$ (KDC generates session key and sends encrypted session key copies to A)
3. $A \rightarrow B: EK_b[Ks\|ID_A]$ (A sends the encrypted session key to B where B can decrypt it using B's secret key)
4. $B \rightarrow A: EKs[N2]$ (B recovers the session key and send back nonce encrypted using session key)
5. $A \rightarrow B: EKs[f(N2)]$ (A replies to B confirming session key and send back fresh message.

- In this protocol, N1 and N2 are two numbers generated at random, except that they cannot repeat between different protocol exchanges. These numbers are called nonces. (If A begins the protocol anew, her N1 in the first exchange will not have been used there before.)



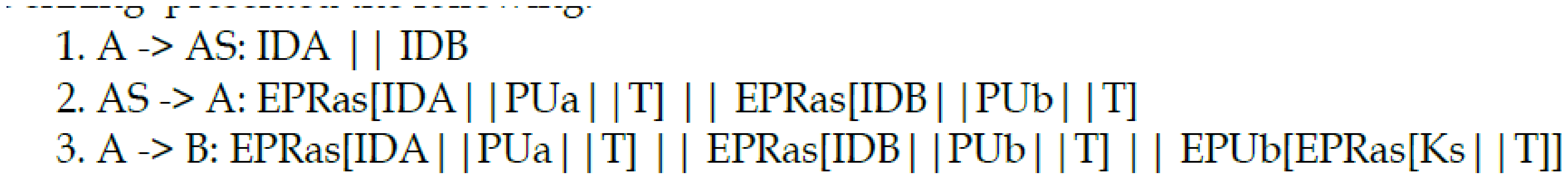
- Used to securely distribute a new session key for communications between A & B
- But is vulnerable to a replay attack if an old session key has been compromised

 Then message 3 can be resent convincing B that is communicating with A

Mutual Authentication Protocol Using Public Key Encryption:

- Have a range of approaches based on the use of public-key encryption
- Need to ensure have correct public keys for other parties
- Using a central authentication server (AS)
- Various protocols exist using timestamps
- Example: Denning AS Protocol

- Denning presented the following:



1. A -> AS: IDA | | IDB
2. AS -> A: EPRas[IDA | |PUa | |T] | | EPRas[IDB | |PUb | |T]
3. A -> B: EPRas[IDA | |PUa | |T] | | EPRas[IDB | |PUb | |T] | | EPUb[EPRas[Ks | |T]]

- note session key is chosen by A, hence AS need not be trusted to protect it
- timestamps prevent replay but require synchronized clocks

One-way Authentication Protocol Using Symmetric Encryption

- KDC strategy is a candidate for encrypted electronic mail.
- For a message with content M, the sequence is as follows:

1. $A \longrightarrow KDC$: $ID_A || ID_B || N_1$
2. $KDC \longrightarrow A$: $E(K_a, [K_s || ID_B || N_1 || E(K_b, [K_s || ID_A])])$
3. $A \longrightarrow B$: $E(K_b, [K_s || ID_A]) || E(K_s, M)$

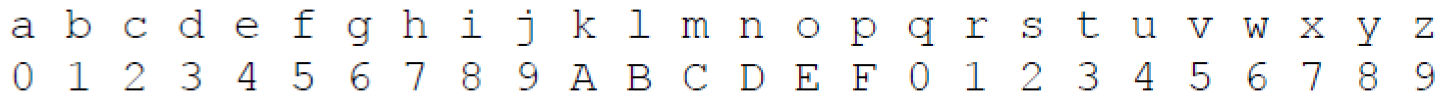
This approach guarantees that only the intended recipient of a message will be able to read it. It also provides a level of authentication that the sender is A. As specified, the protocol does not protect against replays. Some measure of defense could be provided by including a timestamp with the message.
However, because of the potential delays in the e-mail process, such timestamps may have limited usefulness.

One-way Authentication Protocol Using Public Key Encryption

- If confidentiality is the primary concern, then the message can be encrypted with a one-time secret key, which in in turn is encrypted with B‘s public key.
- A->B: EPUb[Ks] || EKs[M]
- has encrypted session key, encrypted message
- To achieve authentication, and to validate the senders public key, the signature can be encrypted with the recipient‘s public key, and for assurance A‘s public key is sent in a digital certificate, as shown. To obtain confidentiality as well, the message can be encrypted with a session key, combining both options above. If authentication is needed use a digital signature with a digital certificate:
- A->B: M || EPRa[H(M)] || EPRas[T||IDA||PUa]
- with message, signature, certificate

- Timestamps: Party A accepts a message as fresh only if the message contains a timestamp that, in A's judgment, is close enough to A's knowledge of current time. This approach requires that clocks among the various participants be synchronized.
- Challenge/response: Party A, expecting a fresh message from B, first sends B a nonce (challenge) and requires that the subsequent message (response) received from B contain the correct nonce value.

- Pass Algorithms Definition: Let there be a challenge-response authentication system in which the function f is the secret. Then f is called a pass algorithm. Under this definition, no cryptographic keys or other secret information may be input to f. The algorithm computing f is itself the secret.
- e.g. Suppose that we have a challenge-response system where the agreed upon transformation is the mapping below:



a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9

- In this mapping a random string of lower-case letters is generated and presented to the user for authentication. The user has the above mapping and translates the lower-case letters in the upper line into the corresponding characters in the lower line. The user then presents those characters to the system for authentication. If the challenge is

The response would be

h	a	r	o	l	d
7	0	1	E	B	3

Assignment

- Define the terms digital signature and digital certificate.
- What are the properties a digital signature should have?
- What requirements should a digital signature scheme satisfy?
- What is Kerberos? Write authentication dialog of Kerberos Version 4.
- Write the differences between Kerberos v4 and v5?
- Differentiate between Direct Digital Signature and Arbitrated Digital Signature.
- What are some threats associated with a direct digital signature scheme?
- Write difference between DSS and RSA.
- Explain Biometrics in detail.
- Explain DSA Algorithm