

---

# Cryptography and System Security (CSS)

Course Code: CSC 604



**Subject Incharge**

Ankita Karia

Assistant Professor

Room No. 421

email: [ankitakaria@sfit.ac.in](mailto:ankitakaria@sfit.ac.in)



# Module 4: Authentication Protocols & Digital signature schemes

---

## 4.1

- User Authentication - One-way and mutual
- Needham Schroeder Authentication protocol
- Kerberos Authentication protocol.

## 4.2

- Digital Signature Schemes
  - a) RSA
  - b) ElGamal
  - c) Schnorr



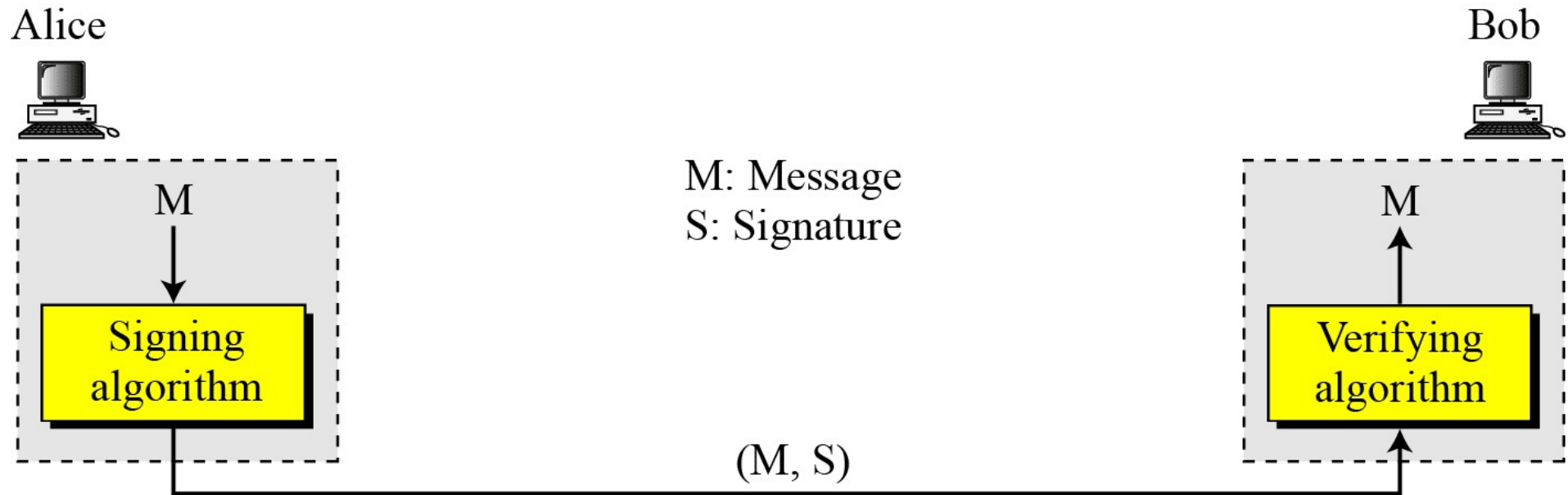
# Digital Signature

---

Conventional Signature	Digital Signature
A conventional signature is included in the document; it is part of the document.	When we sign a document digitally, we send the signature as a separate document.
When the recipient receives a document, she compares the signature on the document with the signature on file.	The recipient receives the message and the signature. The recipient needs to apply a verification technique to the combination of the message and the signature to verify the authenticity.
There is normally a one-to-many relationship between a signature and documents.	There is a one-to-one relationship between a signature and a message.
A copy of the signed document can be distinguished from the original one on file	There is no such distinction unless there is a factor of time on the document.



# Digital Signature Process

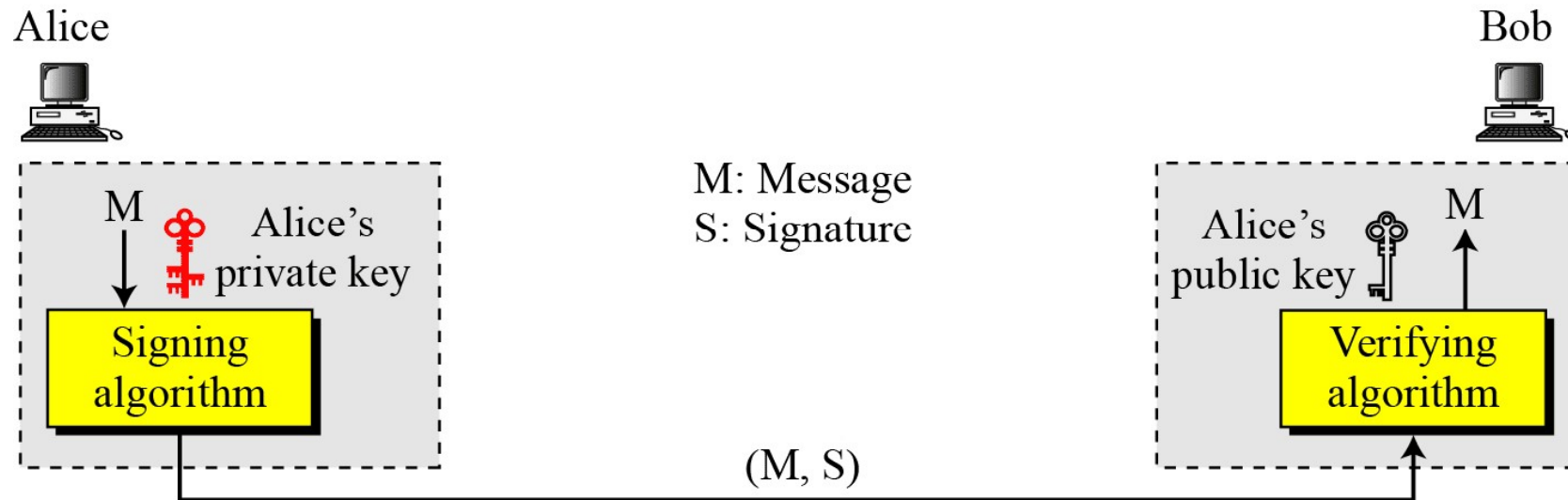


1. The sender uses a signing algorithm to sign the message.
2. The message and the signature are sent to the receiver.
3. The receiver receives the message and the signature and applies the verifying algorithm to the combination.
4. If the result is true, the message is accepted; otherwise, it is rejected.



# Digital Signature – Need for keys

1. The signer uses her Private key, applied to a signing algorithm, to sign the document.
2. The verifier uses the public key of the signer, applied to the verifying algorithm, to verify the document



When a document is signed, anyone receiving the signed document can verify it because everyone has access to sender's public key

Sender must not use her public key to sign the document, because then anyone could forge her signature.



# Digital Signature – Need for keys

---

## *Note*

**A digital signature needs a public-key system. The signer signs with her private key; the verifier verifies with the signer's public key.**

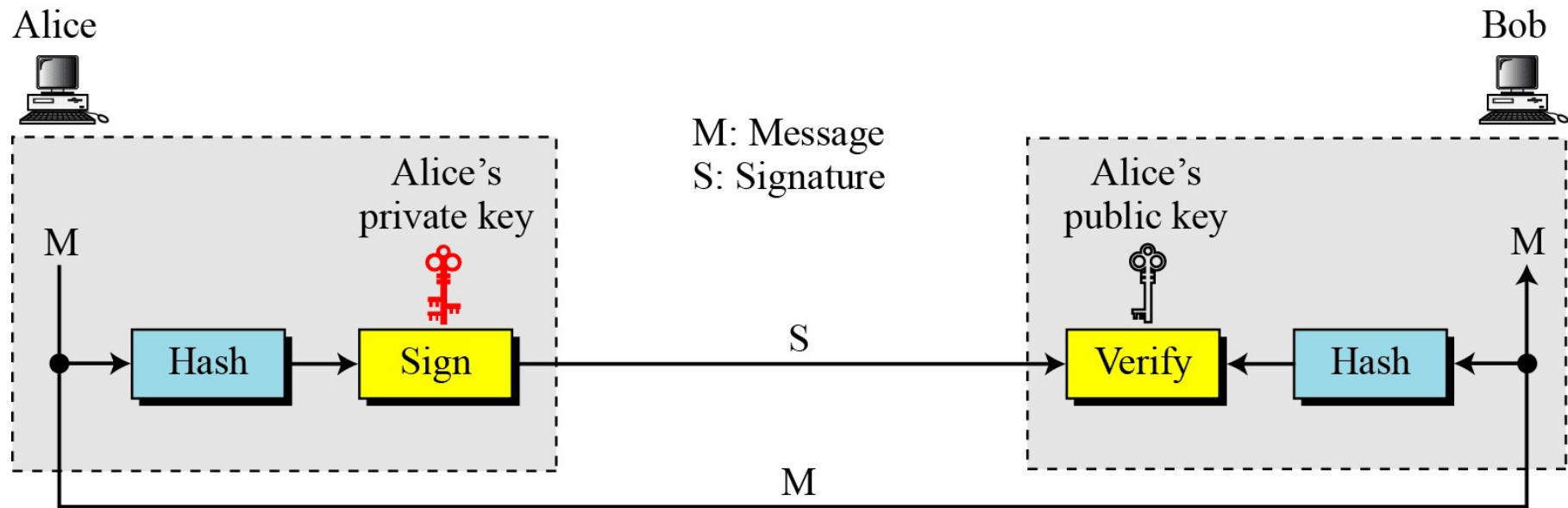
## *Note*

**A cryptosystem uses the private and public keys of the receiver: a digital signature uses the private and public keys of the sender.**



# Signing the digest

1. Asymmetric key cryptosystem are very inefficient when dealing with long messages.
2. In digital signature system, the messages are normally long, but we have to use asymmetric key schemes
3. The solution is to sign a digest of the message, which is much shorter than the message
4. The sender can sign the message digest and the receiver can verify the message digest



# SERVICES

---

1. Various services provided by Digital Signature are: Message Confidentiality, Message Authentication, Message Integrity, and Non-repudiation.
2. A digital signature can directly provide the last three; for message confidentiality we still need encryption/decryption.

## MESSAGE AUTHENTICATION

A secure digital signature scheme, like a secure conventional signature can provide message authentication.

## MESSAGE INTEGRITY

- The integrity of the message is preserved even if we sign the whole message because we cannot get the same signature if the message is changed.
- The digital signature uses a hash function in the signing and verifying algorithms that preserve the integrity of the message.

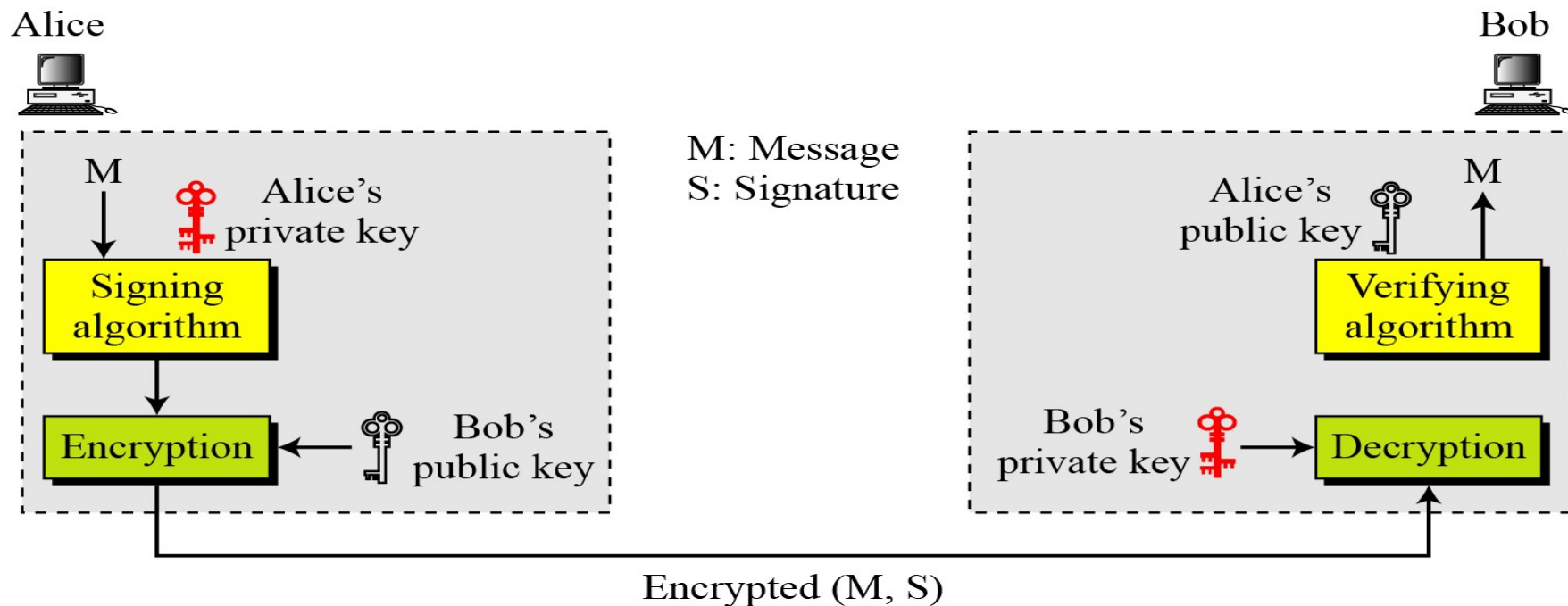




# SERVICES (Contd...)

## CONFIDENTIALITY

- ✓ A digital signature does not provide confidential communication.
- ✓ If confidential communication is required, then the message and the signature must be encrypted using either a Secret key or Public key cryptosystem.



Attacker has access to one or more message-signature pair. In other words, attacker has access to some documents previously signed by Sender. Attacker tries to create another message and forge attacker's signature on it.

## Digital Signature

### Attacks

1. Key-Only Attack
2. Known-Message Attack
3. Chosen-Message Attack

### Forgery

1. Existential Forgery
2. Selective Forgery

Attacker has access only to the public information released by the sender. To forge a message, attacker needs to create Sender's signature to convince receiver that is message is coming from authentic sender.

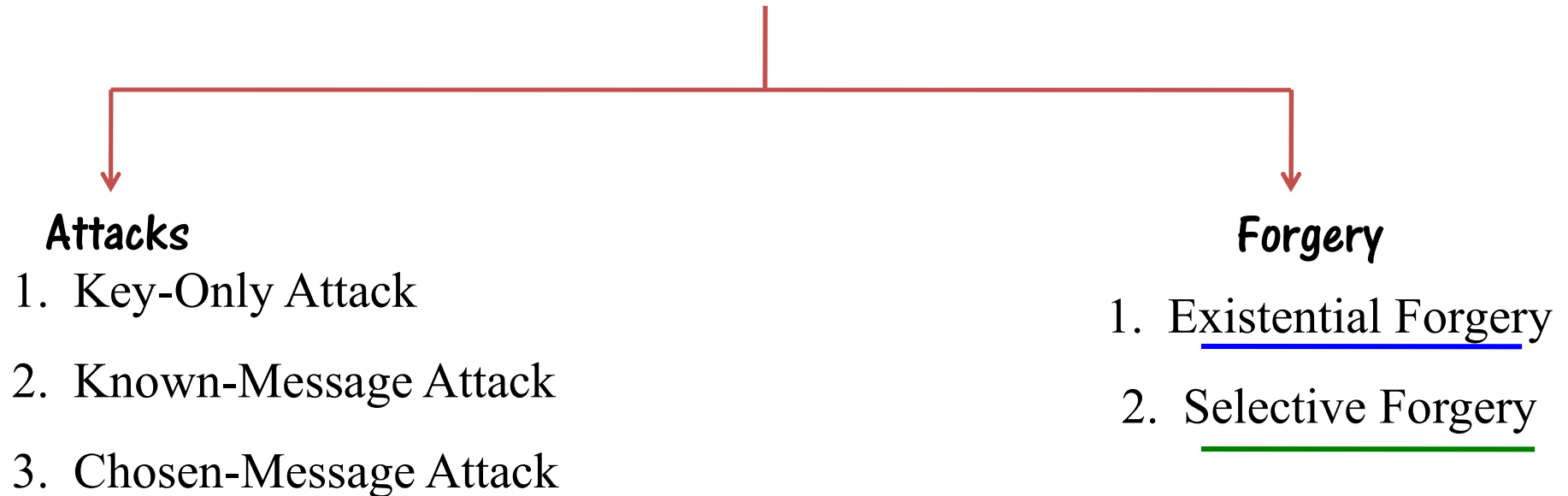
Attacker somehow makes the sender sign one or messages for her. Attacker will now have a chosen message-signature pair. Attacker later creates another message, with the content it wants and forges Sender's signature on it.



# ATTACKS ON DIGITAL SIGNATURE

---

## Digital Signature



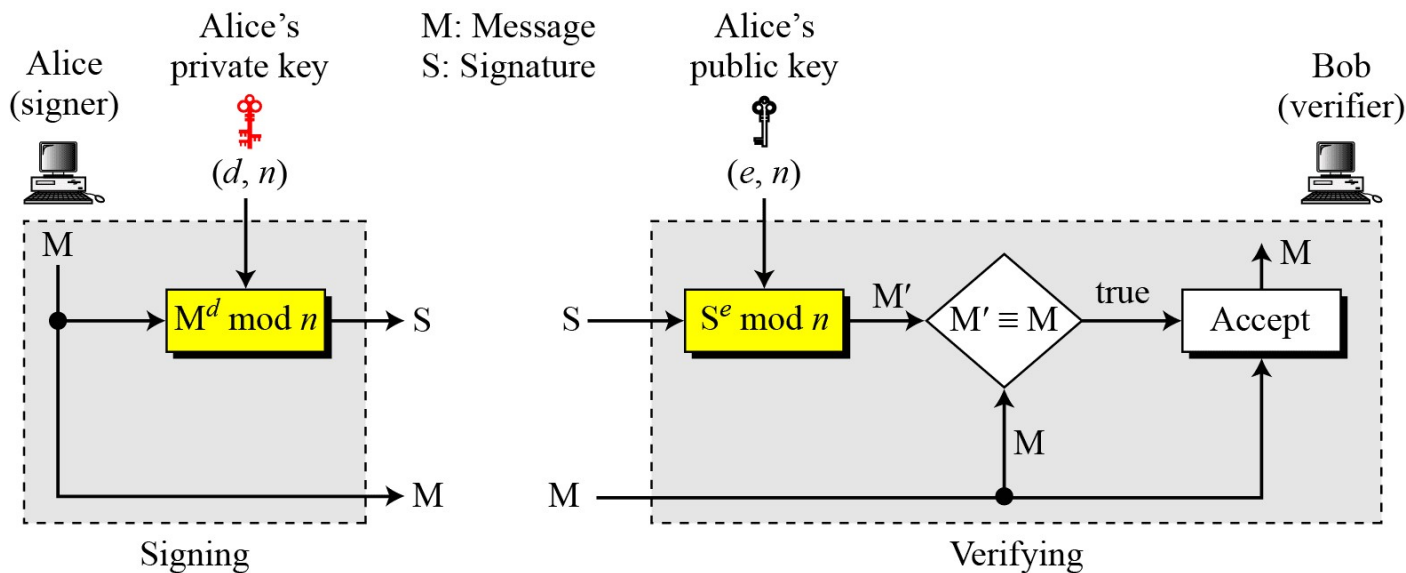
Attacker may be able to create a valid message-signature pair, but cannot really use

Attacker may be able to forge sender's signature on a message with the content selectively chosen by attacker. This is beneficial to the attacker but the probability of such forgery is very low, but not negligible.



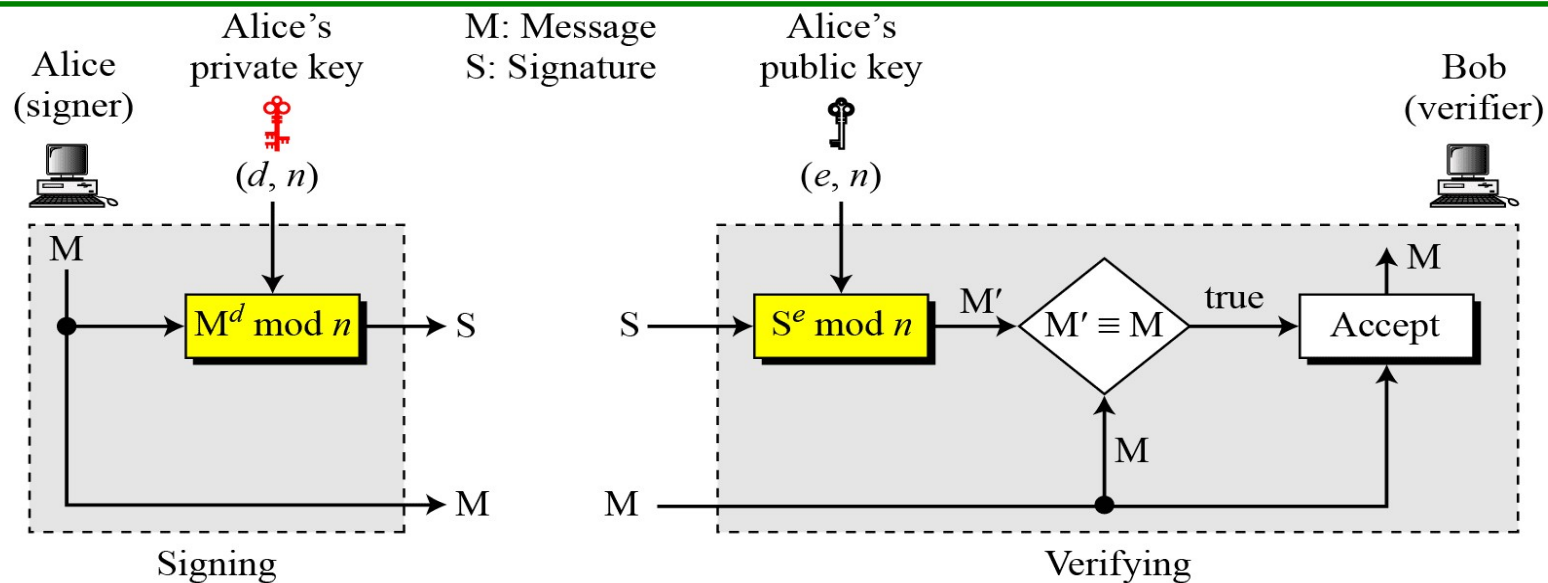
# RSA Digital Signature Scheme

1. The idea of RSA can also be used for signing and verifying a message.
2. This technique is known as **RSA Digital Signature Scheme**.
3. The digital signature scheme changes the role of PRIVATE and PUBLIC key
4. Here, the Private and Public key of the SENDER is used.
5. SENDER uses her own PRIVATE KEY to sign the document.
6. Receiver used the sender's PUBLIC KEY to verify it.

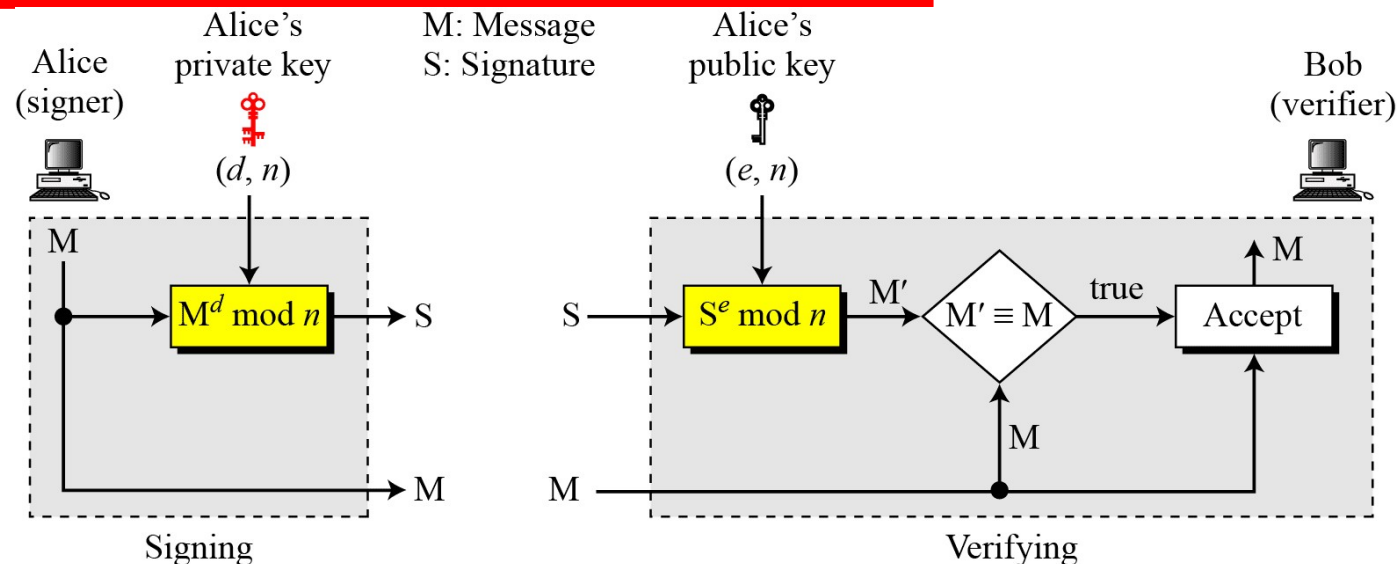


# RSA Digital Signature Scheme – KEY GENERATION

1. Key generation in the RSA digital signature scheme is exactly the same as key generation in the RSA
2. Sender selects 2 prime numbers  $p$  and  $q$
3. Calculate  $\phi(n) = (p-1)(q-1)$
4. Sender then chooses  $e$ , calculates  $d$  such that  $e * d = 1 \bmod \phi(n)$
5. Sender keeps  $d$  and *announces*  $n$  and  $e$



# RSA Digital Signature Scheme – Example



For  $p = 7$  ,  $q = 13$  and  $e = 5$

Calculate  $d$  and Signature  $S$  for Message  $m = 35$

Verify the signature on receiver side



# Elgamal Digital Signature Scheme-General Idea

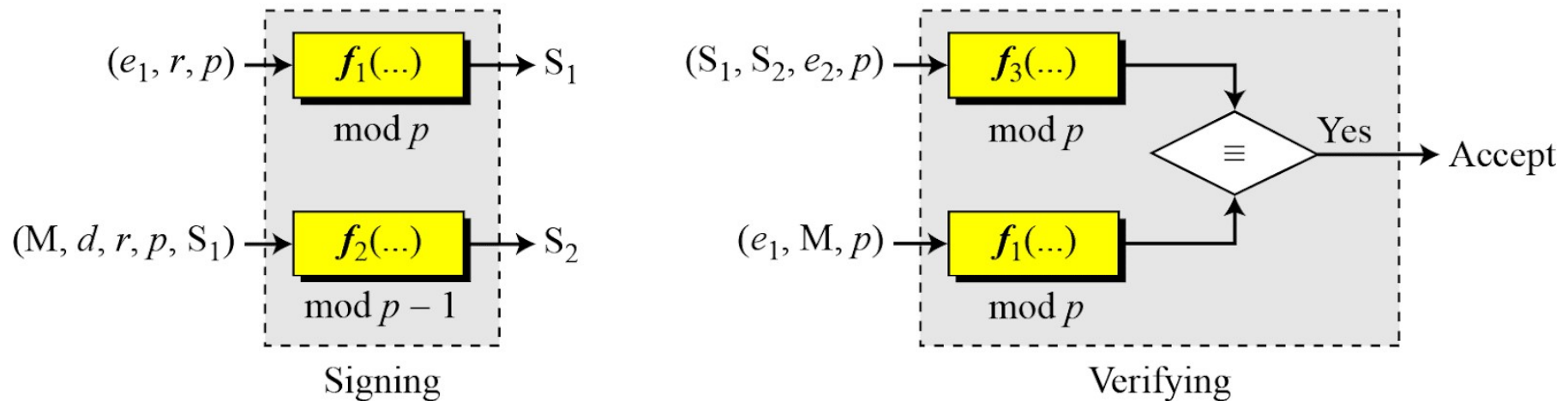
$S_1, S_2$ : Signatures

$M$ : Message

$(e_1, e_2, p)$ : Alice's public key

$d$ : Alice's private key

$r$ : Random secret



**Note**

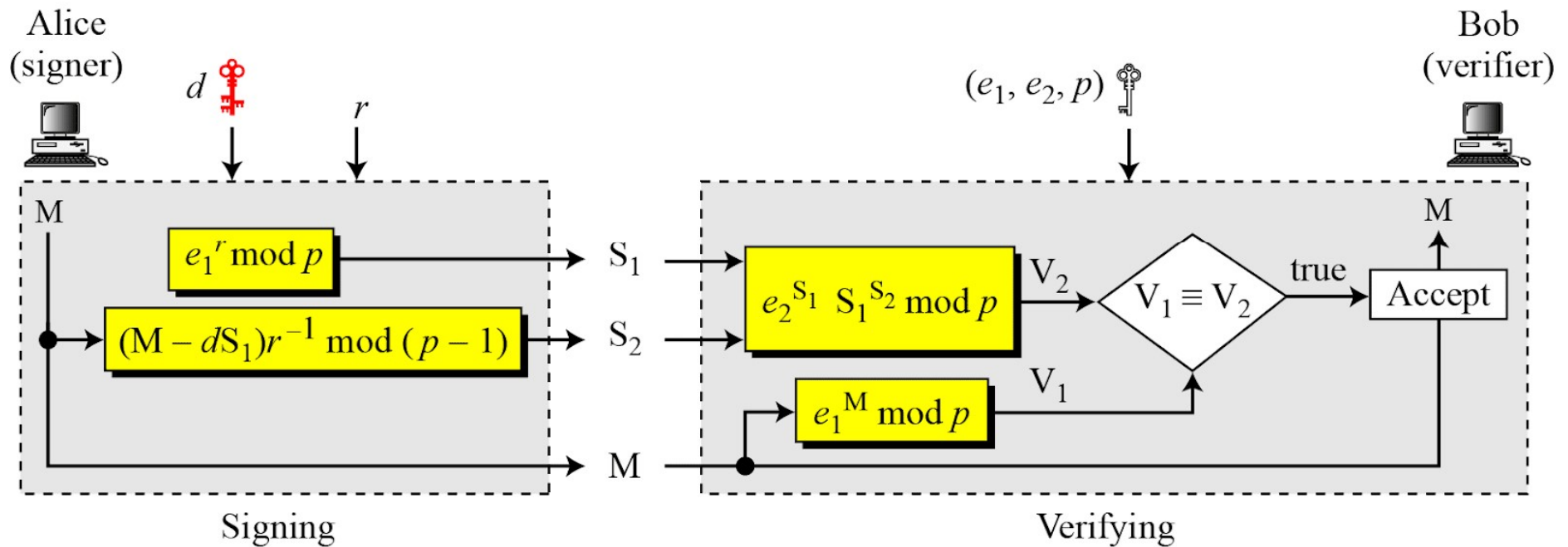
**In ElGamal digital signature scheme,  $(e_1, e_2, p)$  is Alice's public key;  $d$  is her private key.**



# Elgamal Digital Signature Scheme

## Verifying and Signing

M: Message  
 S<sub>1</sub>, S<sub>2</sub>: Signatures  
 V<sub>1</sub>, V<sub>2</sub>: Verifications  
 r: Random secret  
 d: Alice's private key  
 (e<sub>1</sub>, e<sub>2</sub>, p): Alice's public key





# Elgamal Digital Signature Scheme

## Verifying and Signing

SIGNING
Alice chooses a secret random number $r$
Alice calculates the first Signature $S1 = e1^r \text{ mod } p$
Alice calculates the Second Signature $S2 = (M - d * S1) * r^{-1} \text{ mod } (p-1)$ Where $r^{-1}$ is the multiplicative inverse of $r$ modulo $p - 1$
Alice Sends $M, S1, S2$ to Bob

**ALICE: SENDER**

VERIFYNG
Bob checks if $0 < S1 < p$
Bob checks if $0 < S2 < p-1$
Bob calculates $V1 = e1^M \text{ mod } p$
Bob calculates $V2 = e2^{S1} * S1^{S2} \text{ mod } p$

**BOB : RECEIVER**

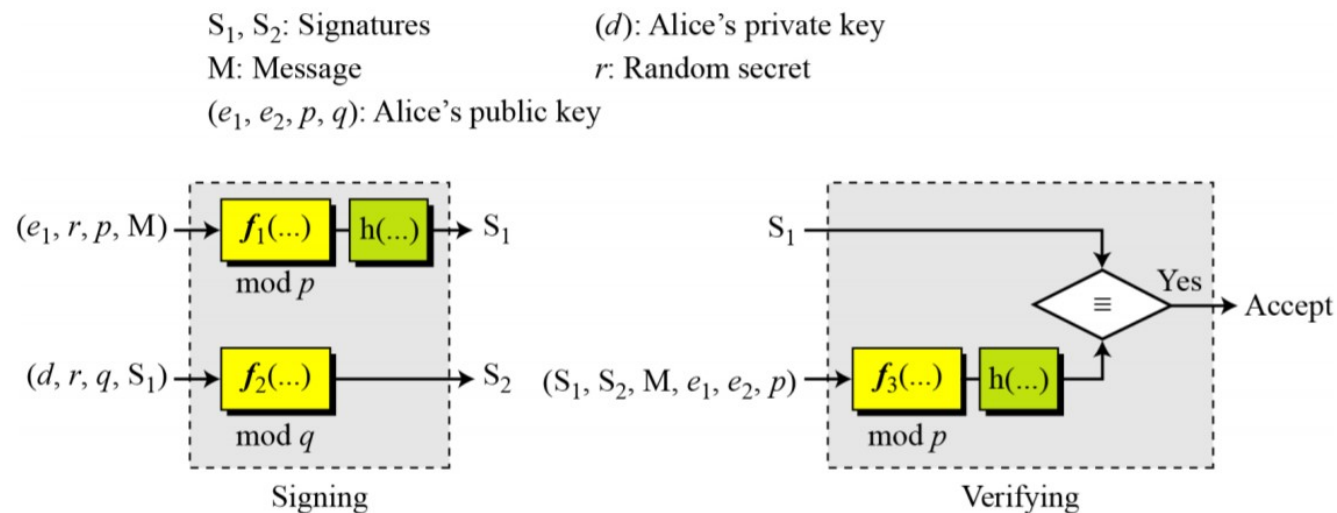
$$e2 = e1^d \text{ mod } p$$

**Example: Using the Elgamal Scheme, let  $p = 19$  and  $d = 16$ . Find  $e2$  if  $e1 = 10$  . Choose  $r = 5$ . Find the values of  $S1$  and  $S2$  if  $M = 14$**



# Schnorr Digital Signature Scheme

1. The problem with the ElGamal digital signature scheme is that  $p$  needs to be very large to guarantee that the discrete log problem is intractable in  $Z_p^*$ .
2. The recommendation is a  $p$  of at least 1024 bits, which could make the signature as large as 2048 bits.
3. To reduce the size of the signature, Schnorr proposed a new scheme based on ElGamal, but with reduced signature size.



# Schnorr Digital Signature Scheme – Key Generation

---

Before signing a message, Alice needs to generate keys and announce the public comes to the public.

1. Alice selects a **prime  $p$**
2. Alice selects **another prime  $q$** , which is the same size as the digest created by the cryptographic has functions (  **$p-1$  should be divisible by  $q$**  )
3. Calculate  **$e1 = e_0^{(p-1)/q} \bmod p$**  ( $e_0$  is a primitive element in  $Z_p$  )
4. Alice chooses an integer  **$d$**  as her private key.
5. Alice calculates  **$e2 = e_1^d \bmod p$**
6. **Alice public key is  $(e1, e2, p, q)$**
7. **Alice private key is  $(d)$**



# Schnorr Digital Signature Scheme

## Signing and Verifying

---

### SIGNING

Alice chooses a secret random number  $r$

Alice calculates the first Signature

$$S1 = h(M \parallel e1^r \bmod p)$$

Alice calculates the Second Signature

$$S2 = r + d * S1 \bmod q$$

Alice Sends  $M$ ,  $S1$ ,  $S2$  to Bob

### VERIFYNG

Bob calculates  $V = h(M \parallel e1^{S2} e2^{-S2} \bmod p)$

If  $S1$  is congruent to  $V$  modulo  $p$ , the message is accepted; otherwise it is rejected.



# Module 4: Authentication Protocols & Digital signature schemes

---

## 4.1

- User Authentication - One-way and mutual
- Needham Schroeder Authentication protocol
- Kerberos Authentication protocol.

## 4.2

- Digital Signature Schemes
  - a) RSA
  - b) ELGamal
  - c) Schnorr



# User Authentication

---

1. User authentication is the fundamental building block and the primary line of defence.
2. User authentication is the basis for most types of access control and for user accountability.
3. User Authentication: Is the process of verifying an identity claimed by or for a system entity.
4. An authentication process consists of two steps:
  - a) Identification Step: Presenting an identifier to the security system.
  - b) Verification step: Presenting or generating authentication information that corroborates the binding between the entity and the identifier.



# Means of authenticating User's Identity

---

There are 4 means of authenticating a user's identity, which can be used alone or in combination:

- A. Something the individual knows
- B. Something the individual possesses
- C. Something the individual is (Static Biometrics)
- D. Something the individual does (Dynamic Biometrics)

All of the above methods, if properly implemented and used, can provide secure user authentication.

However, each method has problems



# MUTUAL AUTHENTICATION

---

- Mutual Authentication 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:
  1. **Confidentiality:** To prevent masquerade and to prevent compromise of session keys, essential identification and session-key must be communicated in an encrypted form
  2. **Timeliness:** Is important because of the threat of MESSAGE REPLAYS. Such replays could allow an opponent to compromise a session key or successfully impersonate another party
- To achieve mutual authentication following two general approaches are used;
  - a) **Timestamps**
  - b) **Challenge/Response**





# MUTUAL AUTHENTICATION

---

## a) Timestamps

Party A accepts a message as fresh only if the message contains a **TIMESTAMP** that, In A's judgement, is close enough to A's knowledge of current time. This approach requires that clocks among the various participants be synchronized.

## b) Challenge/Response

Party A, expecting a fresh message from B, first sends B a **NONCE** and requires that the subsequent message (Response) received from B contain the correct **NONCE** value.



# Difficulties involved in following approach.....

---

## a) Timestamps

- Should not be used for CONNECTION-ORIENTED Applications
- **Reasons:**
  - a) Some sort of protocol is needed to maintain synchronization among the various processor clocks. This protocol must be Fault Tolerant, should be able to cope with network errors and hostile attacks.
  - b) Because of variable and unpredictable nature of network delays, distributed clocks cannot be expected to maintain precise synchronization.

## b) Challenge/Response

- Should not be used for CONNECTION-LESS Applications
- Since it requires the overhead of a handshake before any connectionless transmission, effectively negating the chief characteristic of a connectionless transaction



# Key Distribution Centre : KDC

---

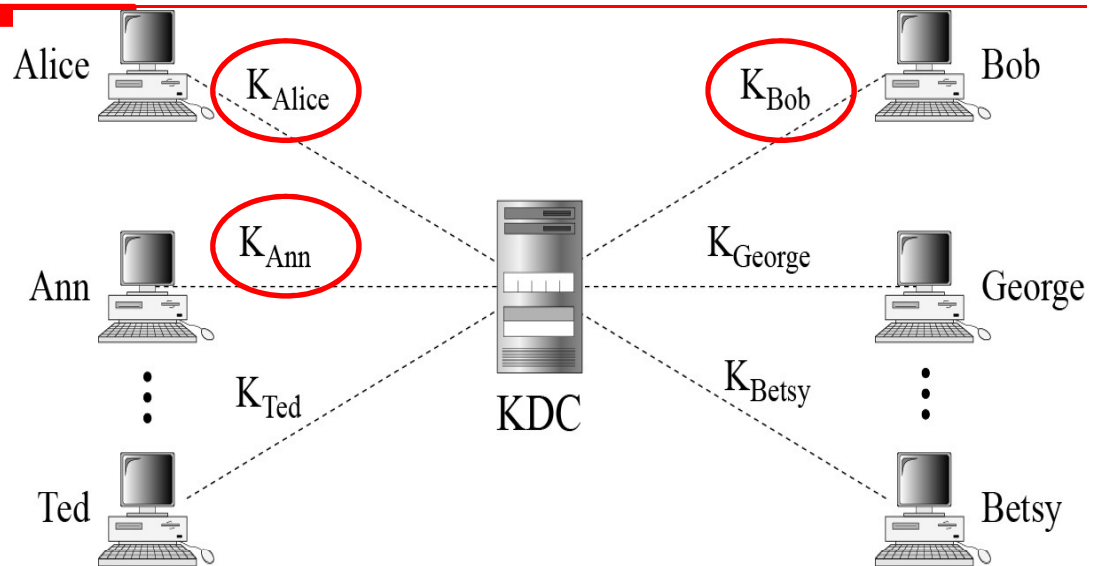
- Symmetric-key cryptography is more efficient than asymmetric-key cryptography for enciphering large messages.
- Symmetric-key cryptography, however, needs a shared secret key between two parties.
- If Alice needs to exchange confidential messages with  $N$  people, she needs  $N$  different keys.
- Thus, if  $N$  people need to communicate with each other, then a total of  $N(N-1)$  keys are required.
- An efficient way is required to maintain and distribute Secret Keys.
- **To reduce the number of keys and to distribute the secret keys securely, Trusted third party called as KDC is used**



# Key Distribution Centre : KDC

The following process is taken when Alice wants to send a confidential message to Bob.

1. Alice sends a request to the KDC, that she needs a **session key** between herself and Bob.
2. The KDC informs Bob about Alice's request.
3. If Bob agrees a session key is created between the two



**The SESSION KEY between Alice and Bob that is established with the KDC is used to authenticate Alice and Bob to the KDC and to prevent Attacker from impersonating either of them**



# Key Distribution Centre : KDC

---

## Secret Keys :

A KDC creates a secret key for each member.

This secret key can be used only between the member and the KDC, not between two members.

## Session Keys:

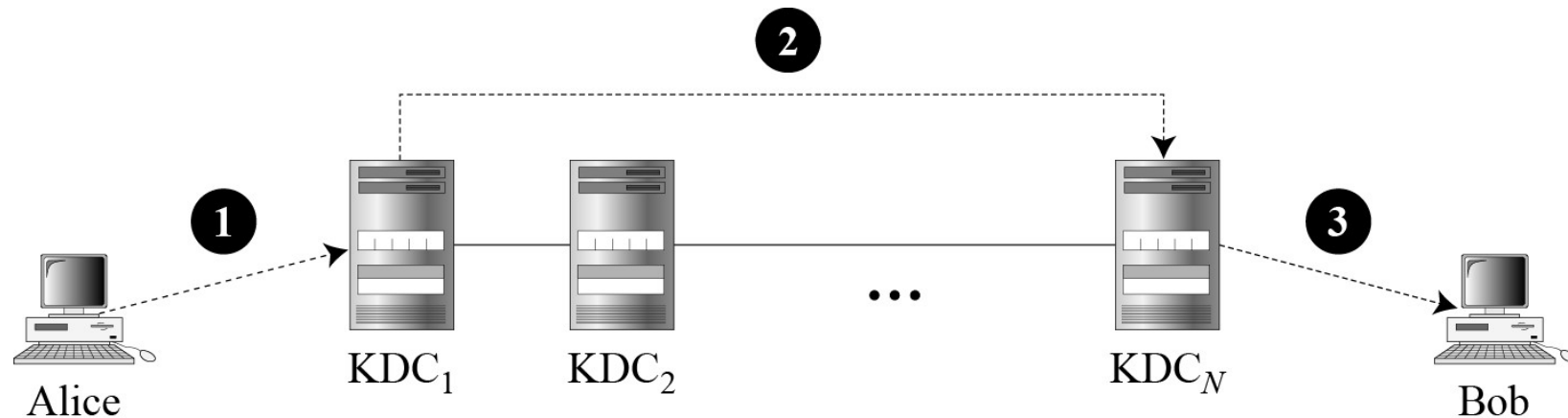
Session Keys is created between 2 members ( Eg. Alice and Bob)

A session symmetric key between two parties is used only once.



# Flat Multiple KDCs.

- When the number of people using KDC increases, the system becomes unmanageable and a bottleneck can result.
- To solve this problem, we need to have multiple KDC's
- We can divide the world into domains, each domain can have one or more KDC's (for redundancy in case of failure)

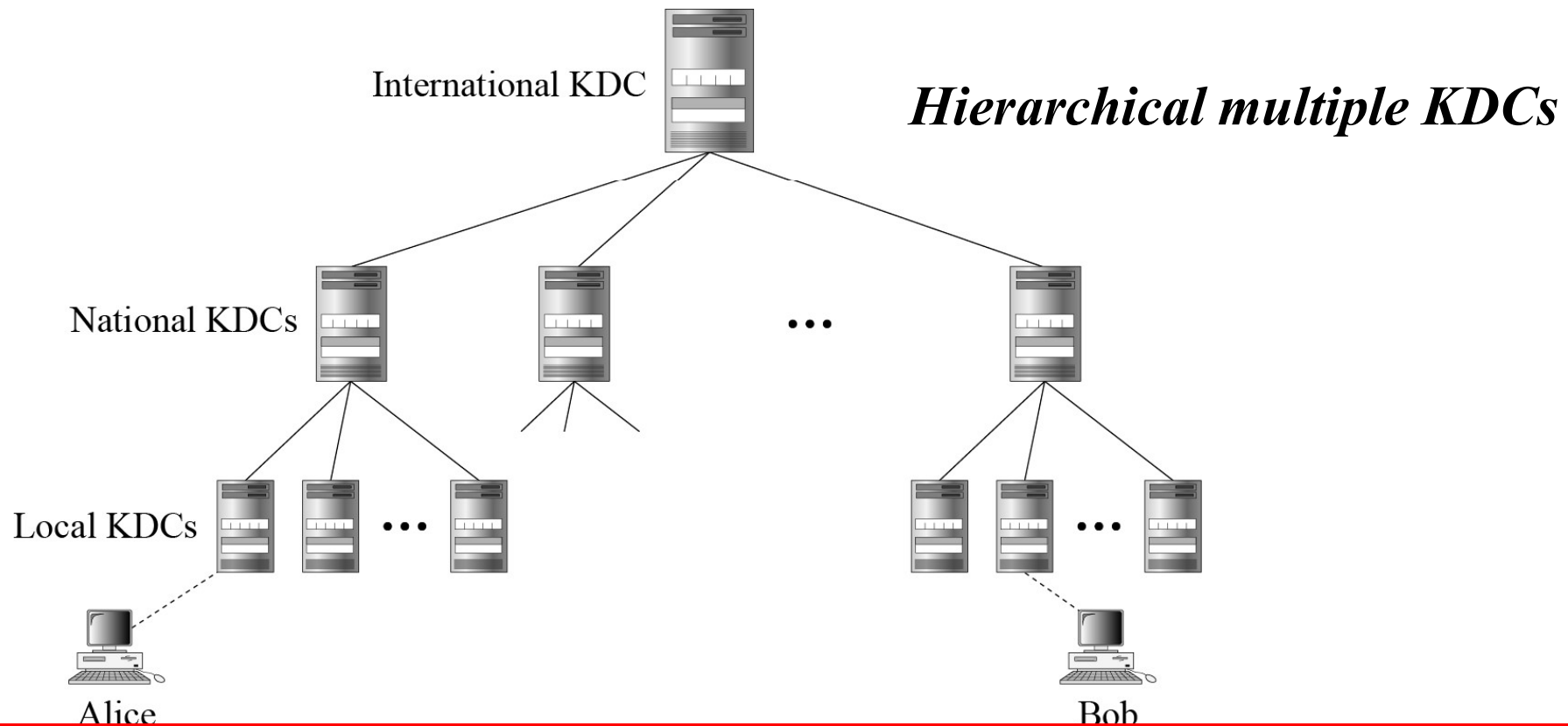


**The above concept is referred as FLAT MULTIPLE KDC**



# Hierarchical Multiple KDCs

- The concept of flat multiple KDC's can be extended to a hierarchical system of KDC's, with one or more KDC's at the top of the hierarchy.



KDC receives the message and creates  
what is called a **TICKET**


## *A Simple Protocol Using a KDC*

### *First approach using KDC*

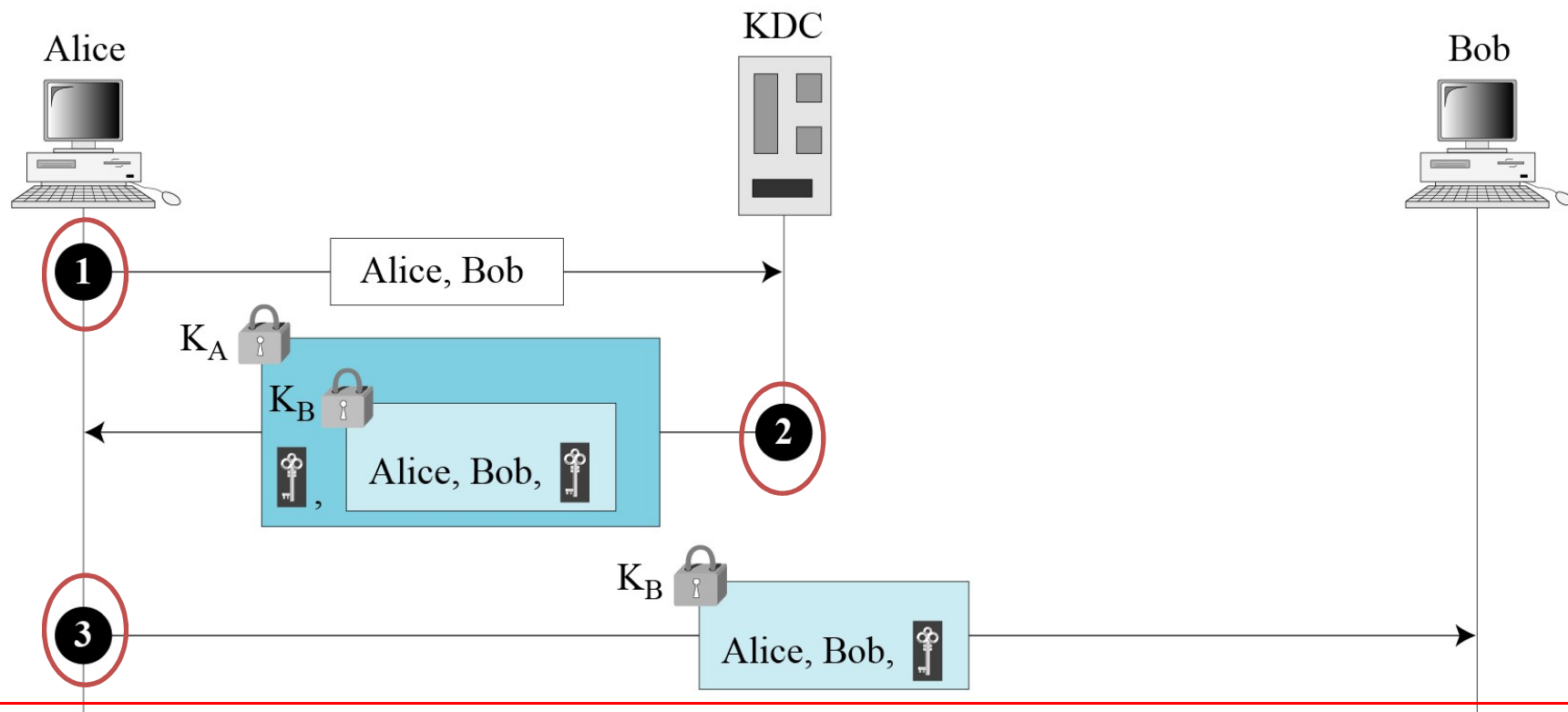
$K_A$   Encrypted with Alice-KDC secret key



Session key between Alice and Bob

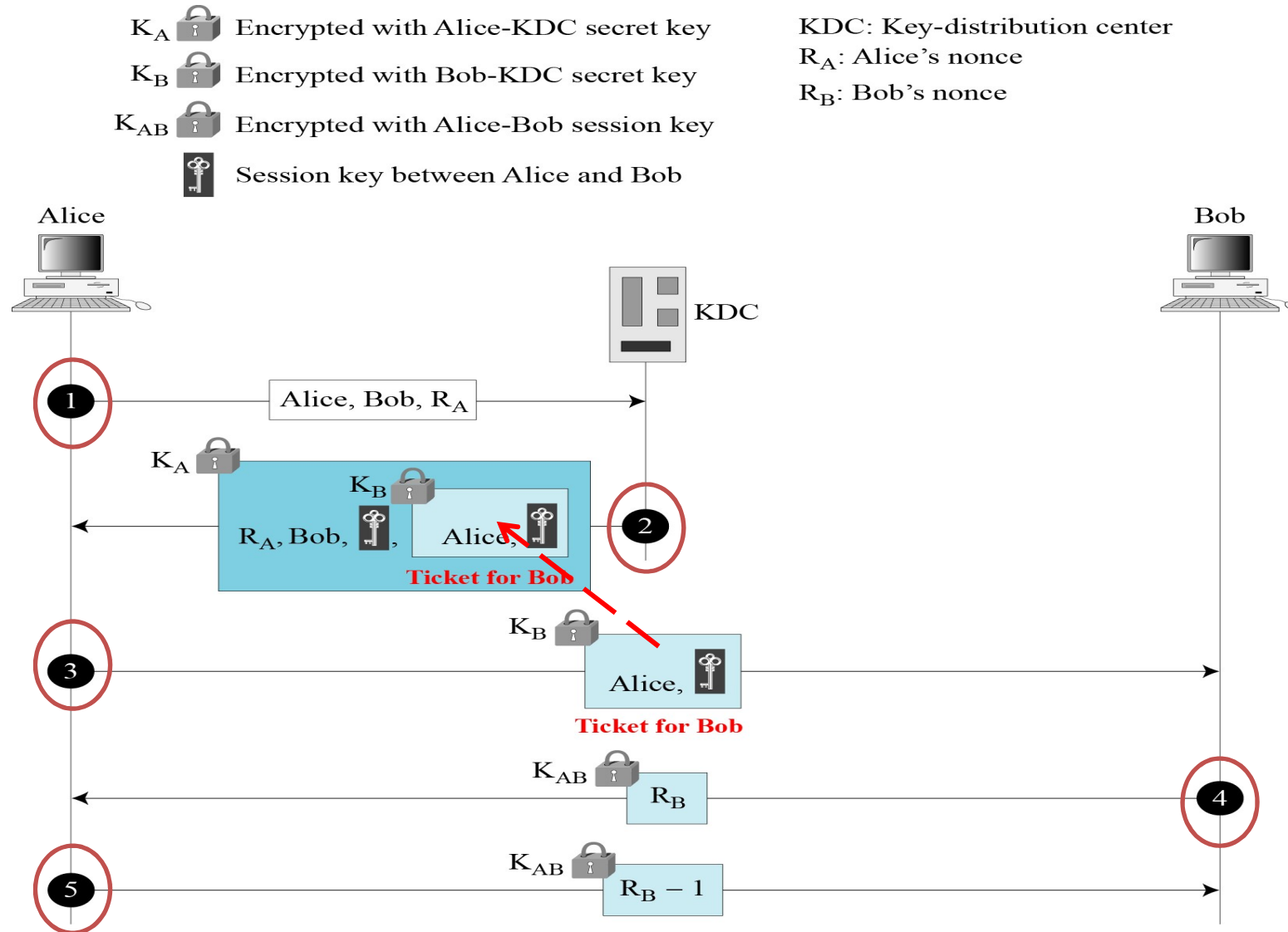
$K_B$   Encrypted with Bob-KDC secret key

KDC: Key-distribution center





# Needham-Schroeder Protocol



# KERBEROS

---

- Kerberos is an authentication protocol, and at the same time a KDC, that has become very popular.
- Several systems, including Windows 2000, use Kerberos.
- Originally designed at MIT, it has gone through several versions.

## ***SERVICES***

Three servers are involved in the Kerberos protocol:

- **Authentication Server (AS):**

The authentication server (AS) is the KDC in the Kerberos protocol. Each user registers with the AS and is granted a user identity and a password. The AS has a database with these identities and the corresponding passwords. The AS verifies the user, issues a session key to be used between Alice and TGS, and send a ticket for the TGS

- **Ticket Granting Server (TGS):**

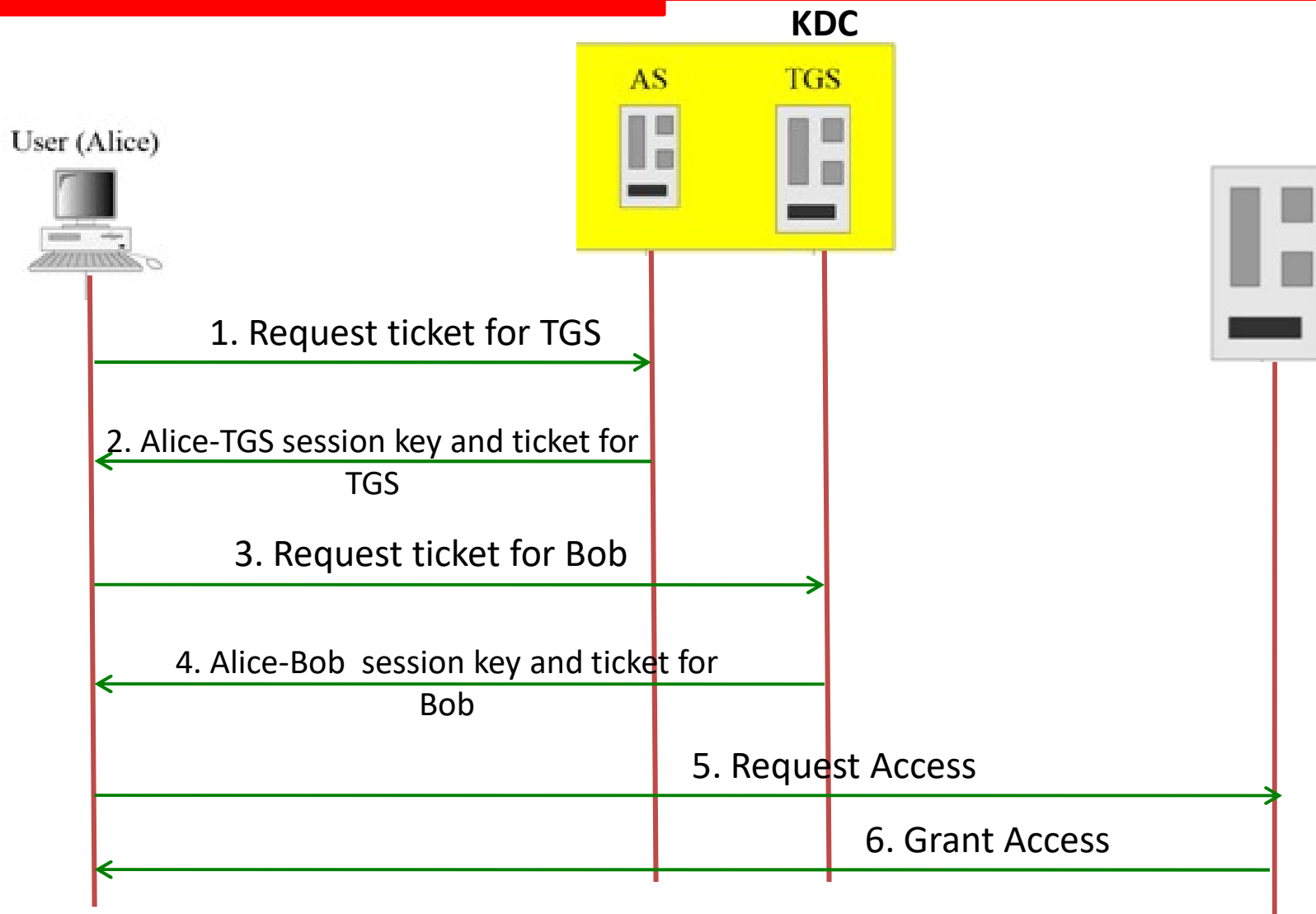
The TGS issues a ticket for the real server (Bob). It also provides the session key K<sub>AB</sub> between Alice and Bob. Kerberos has separated user verification from the issuing of tickets. In this way, though Alice verifies her ID just once with AS, she can contact TGS multiple times to obtain tickets for different real servers

- **A Real (data) Server :**

The real server (Bob) provides services for the user (Alice).



# KERBEROS



# KERBEROS

