



SY BTech Semester-IV (AY 2022-23)

Computer Science and Engineering (Cybersecurity and Forensics)

Disclaimer:

- a. Information included in these slides came from multiple sources. We have tried our best to cite the sources. Please refer to the [references](#) to learn about the sources, when applicable.
- b. The slides should be used only for preparing notes, academic purposes (e.g. in teaching a class), and should not be used for commercial purposes.

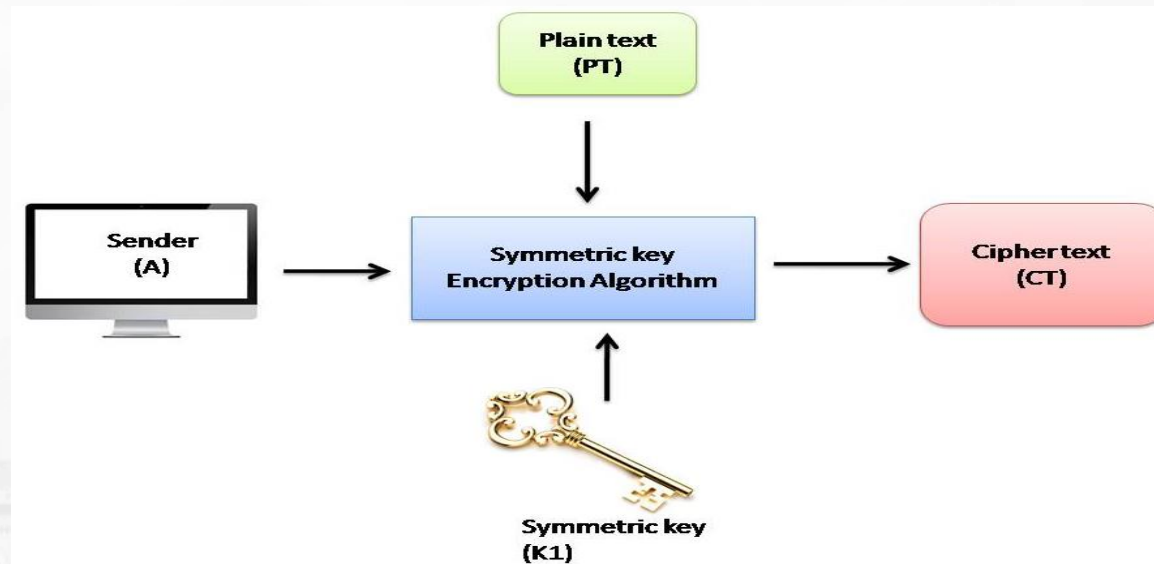
Syllabus

Unit: III	Authentication and Digital Signatures: Use of Cryptography for authentication, Secure Hash function, Key Management and Distribution: Symmetric Key Distribution Using Symmetric Encryption, Symmetric Key Distribution Using Asymmetric Encryption, Distribution of Public Keys Cryptographic Key Infrastructures, Diffie-Hellman Key Exchange, Digital Certificates x509. Authentication Protocols: Remote, Mutual Authentication, Authentication Methods: Password, Two way methods, Biometric Authentications, Kerberos Security	9 Hrs
----------------------	---	------------------

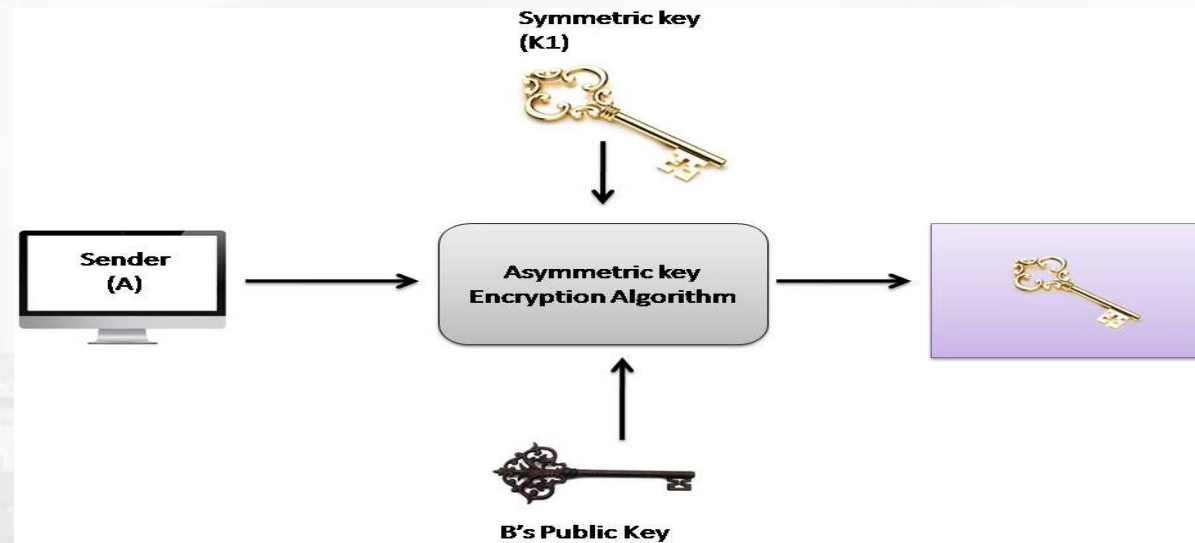
Assign No.	Name of Assignment
6	Write a program using JAVA or Python or C++ to implement Diffie Hellman Key Exchange Algorithm
7	Write a program using JAVA or Python or C++ to implement Digital signature using DSA.

Symmetric and Asymmetric Key Cryptography Together

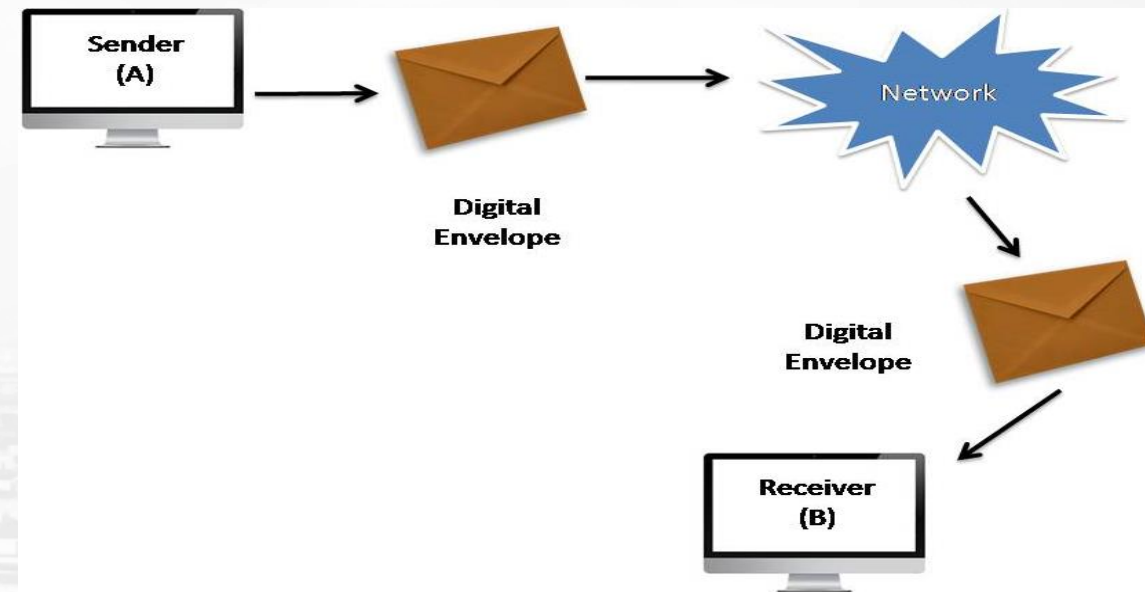
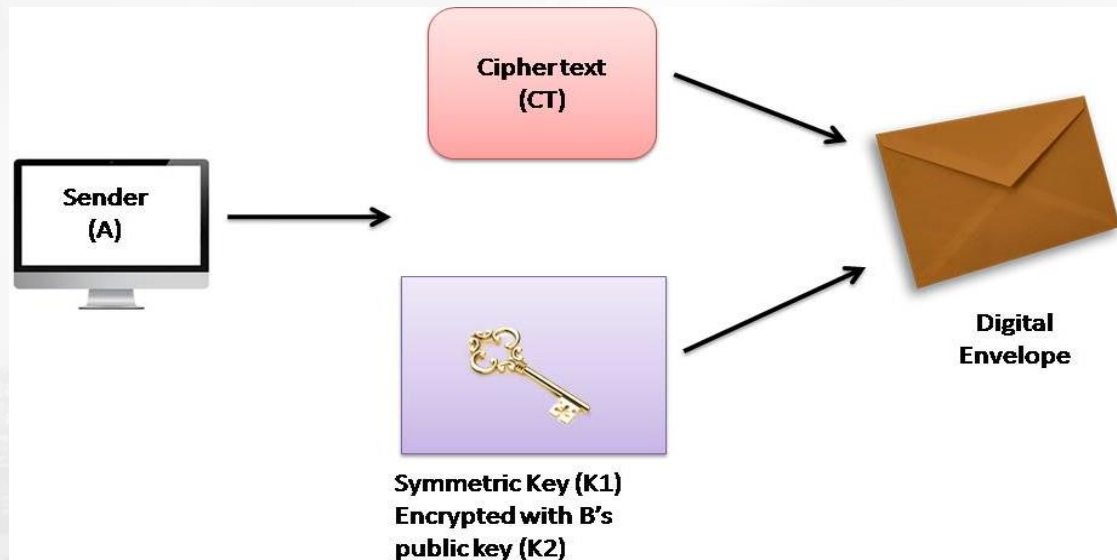
❖ Sender (A) and Receiver (B)

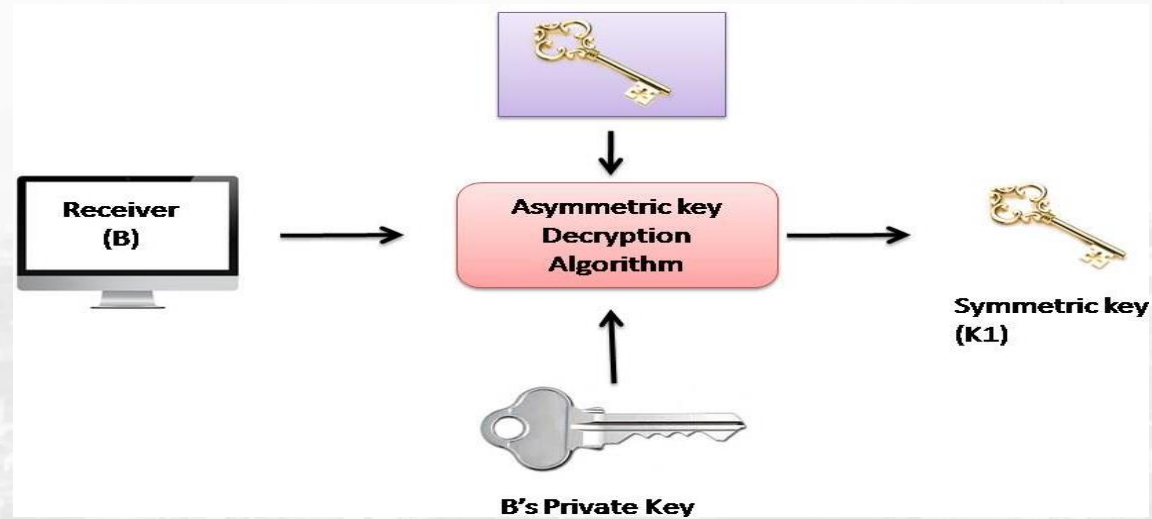
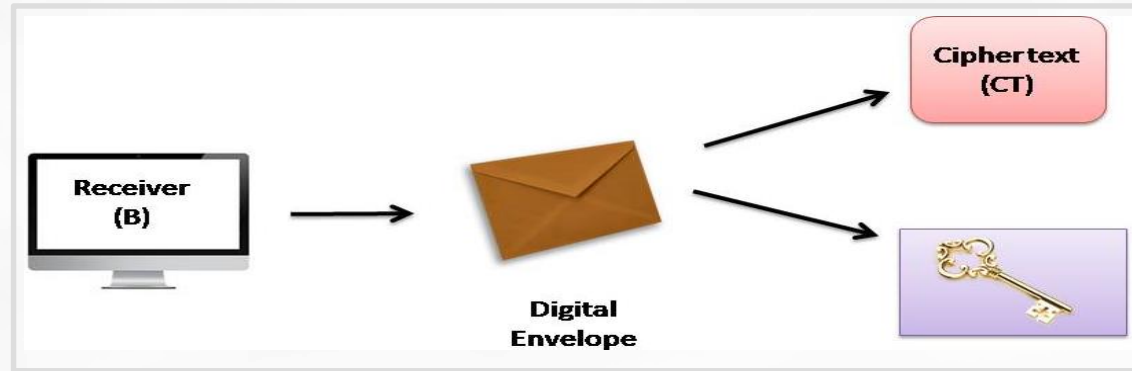


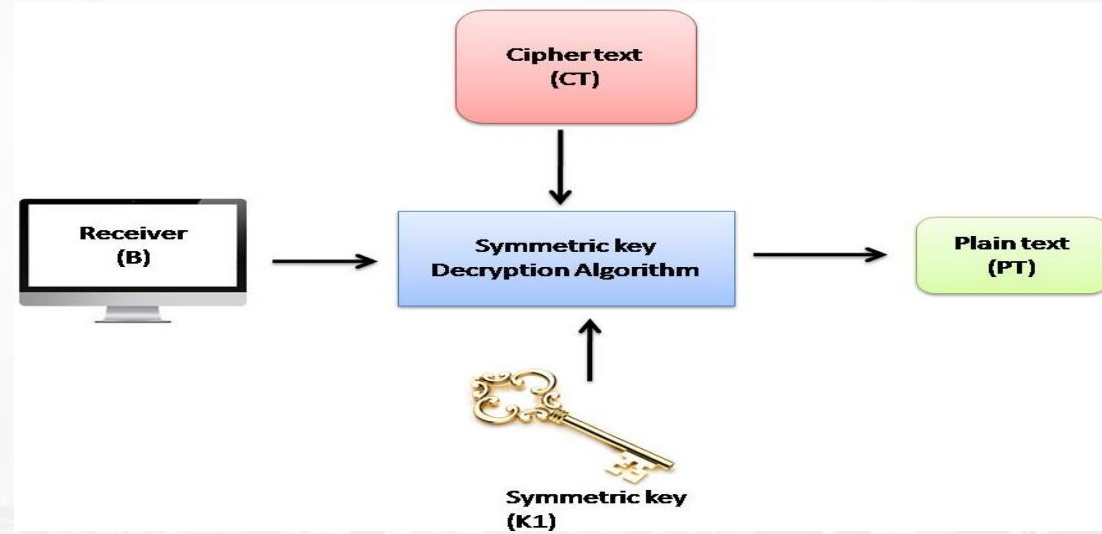
- ❖ Takes the one-time symmetric key (i.e. $K1$), and encrypts $K1$ with B's public key ($K2$). This process is called **key wrapping** of the symmetric key



- ❖ A puts the cipher text CT and the encrypted symmetric key together inside a digital envelope







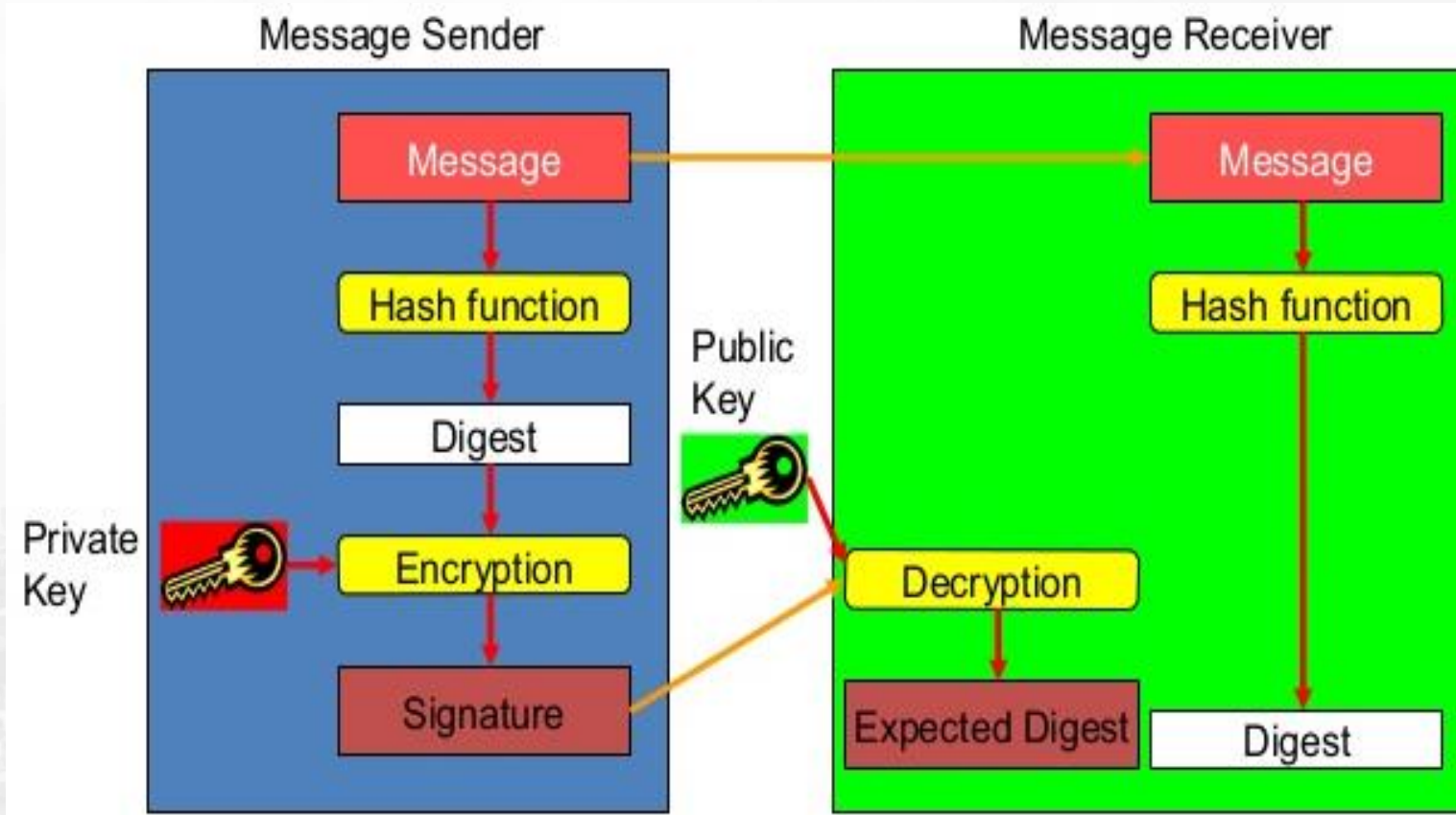
Digital Signature Techniques

- ❖ Digital Signature Standard (DSS) uses SHA-1
- ❖ RSA and DSA

Digital signature is used for communication to verify:

- ❖ Authentication of the sender
- ❖ Integrity of the message received
- ❖ Non-repudiation

RSA and Digital Signature



Digital Signature Algorithm (DSA)

- ❖ creates a 320 bit signature
- ❖ smaller and faster than RSA
- ❖ security depends on difficulty of computing discrete logarithms

DSA Key Generation

❖ have shared **global public key** values (p, q, g):

– choose a large prime p with $2^{L-1} < p < 2^L$

• where $L = 512$ to 1024 bits and is a multiple of 64

– choose q

• such that q is a 160 bit prime divisor of $(p-1)$

– choose $g = h^{(p-1)/q}$

• where $1 < h < p-1$ and $h^{(p-1)/q} \bmod p > 1$

❖ users choose **private key** & compute **public key**:

– choose random private key: $x < q$

– compute public key: $y = g^x \bmod p$

Note: L will be one member of the set $\{512, 576, 640, 704, 768, 832, 896, 960, 1024\}$

DSA Signature Creation

- ❖ to **sign** a message M the sender:
 - generates a random signature key k , $k < q$
- ❖ then computes signature pair:
$$r = (g^k \bmod p) \bmod q$$
$$s = [k^{-1}(\text{SHA}(M) + x * r)] \bmod q$$
- ❖ sends **signature** (r, s) with message M

DSA Signature Verification

- ❖ having received M & signature (r, s)
- ❖ to **verify** a signature, recipient computes:

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

$$u_1 = [\text{SHA}(M) * w] \bmod q$$

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

$$v = [(g^{u_1} * y^{u_2}) \bmod p] \bmod q$$

- ❖ if **v = r** then signature is verified

Secure Hash Function

- ❖ A hash function H accepts a variable-length block of data as input and produces a fixed-size hash value $h = H(M)$
- ❖ Does not take a secret key as input
- ❖ APPLICATIONS OF CRYPTOGRAPHIC HASH FUNCTIONS

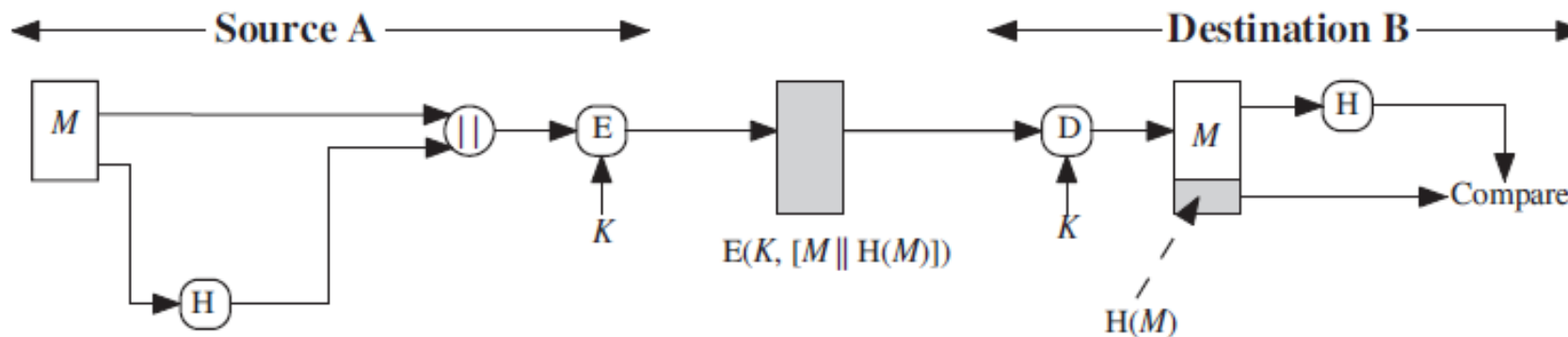
1. Message Authentication

2. Digital Signatures

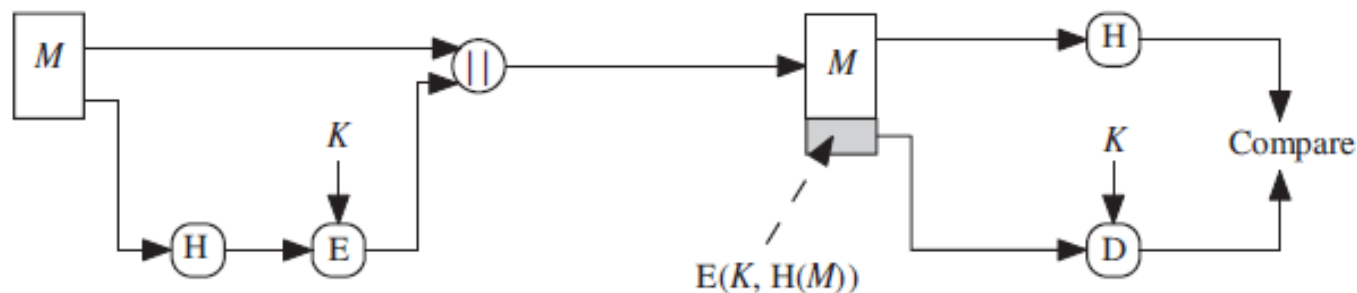
1. Message Authentication

- Message authentication is a mechanism or service used to **verify the integrity of a message**.
i.e. Integrity Checking: Virus and Malware Scanning
- Message authentication assures that data received are exactly as sent
i.e. contain no modification, insertion, deletion, or replay

Simplified Examples of the Use of a Hash Function for Message Authentication



(a) Message Integrity and Confidentiality

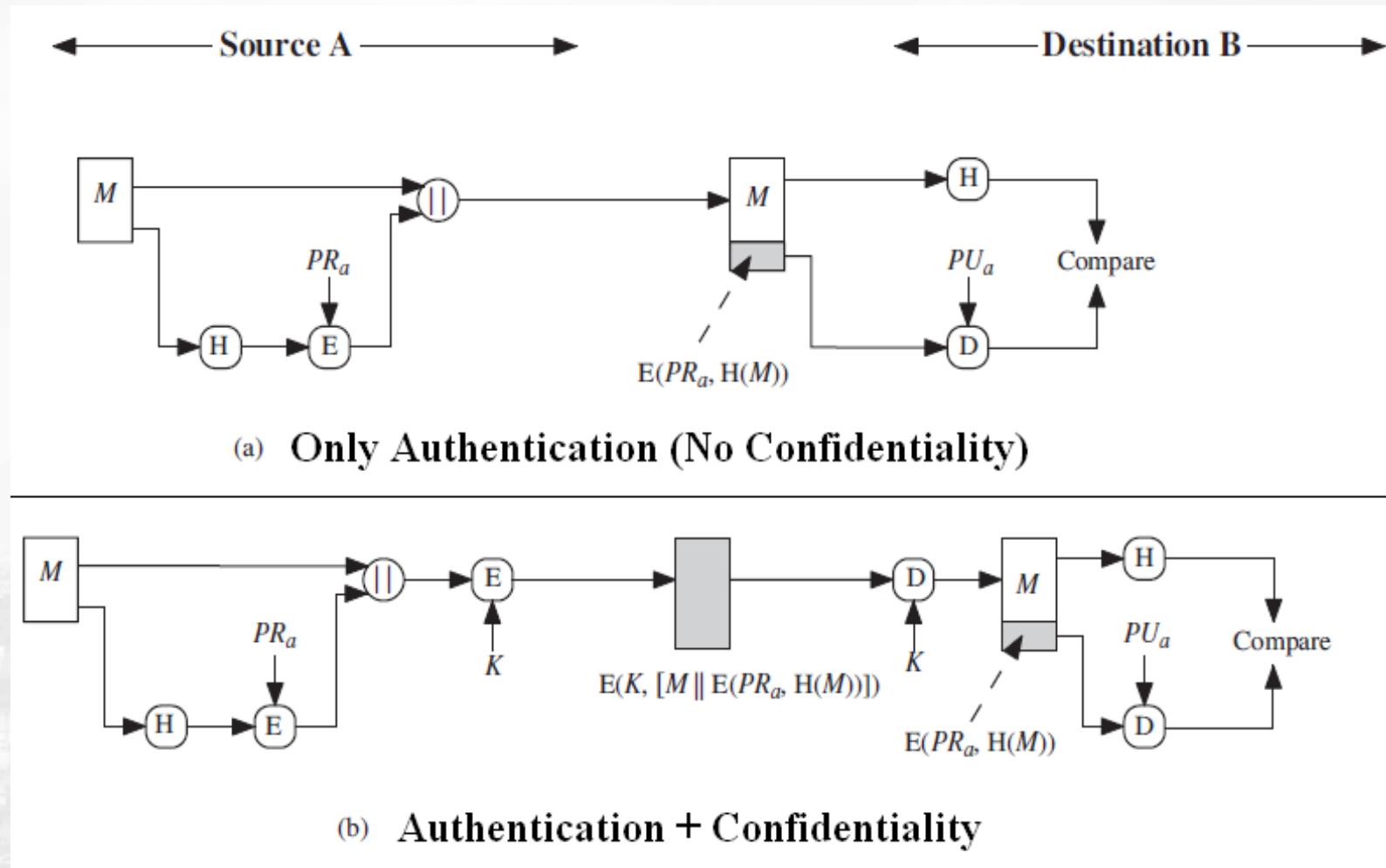


(b) Only Message Integrity ; No Confidentiality

2. Digital Signatures

- In the case of the digital signature, the hash value of a message is encrypted with a user's private key.
- 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.

Simplified Examples of the Use of a Hash Function for Digital Signatures



Key Management

- ❖ Problem of key management
- ❖ Role of Random Numbers
- ❖ Key Types
- ❖ Key Related Issues
- ❖ Public Key Infrastructure
- ❖ Symmetric Key Infrastructure
- ❖ Diffie Hellman Key exchange
- ❖ Authentication Protocols
- ❖ Digital Certificates

Types of Keys



- **Public** Key
- **Private** Key
- **Session** Key
- **Permanent** Key
- **One time** Key

<https://www.youtube.com/watch?v=PbOtZ1J34mY>

Key Management Issues



- Key Generation
- Key Distribution
- Key Storage
- Key Renewal
- Key Revoking
- Key Verification
- Key Expiry
- Key Misuse

Key Management and Distribution

- ❖ Symmetric Key Distribution Using Symmetric Encryption
 - A Key Distribution Scenario
 - Hierarchical Key Control
 - Session Key Lifetime
 - A Transparent Key Control Scheme
 - Decentralized Key Control
- ❖ Symmetric Key Distribution Using Asymmetric Encryption
- ❖ Simple Secret Key Distribution
- ❖ Secret Key Distribution with Confidentiality and Authentication
- ❖ Distribution of Public Keys
 - Public Announcement of Public Keys
 - Publicly Available Directory
 - Public-Key Authority
 - Public-Key Certificates

- ❖ topics of cryptographic key management / key distribution are complex
 - cryptographic, protocol, & management issues
- ❖ Symmetric schemes require both parties to share a common secret key
- ❖ Public key schemes require parties to acquire valid public keys
 - ❖ issue is how to securely distribute this key
 - ❖ whilst protecting it from others

Symmetric Key Distribution using Symmetric Encryption

❖ Given parties A and B have various **key distribution** alternatives:

1. A can select key and physically deliver to B
2. Third party can select the key and physically deliver key to A and B
3. If A and B have communicated previously can use previous key to encrypt a new key
4. If A & B have secure communications with a third party C, C can deliver key between A & B

❖ A **key distribution center (KDC)** is responsible for distributing keys to pairs of users (hosts, processes, applications) as needed.

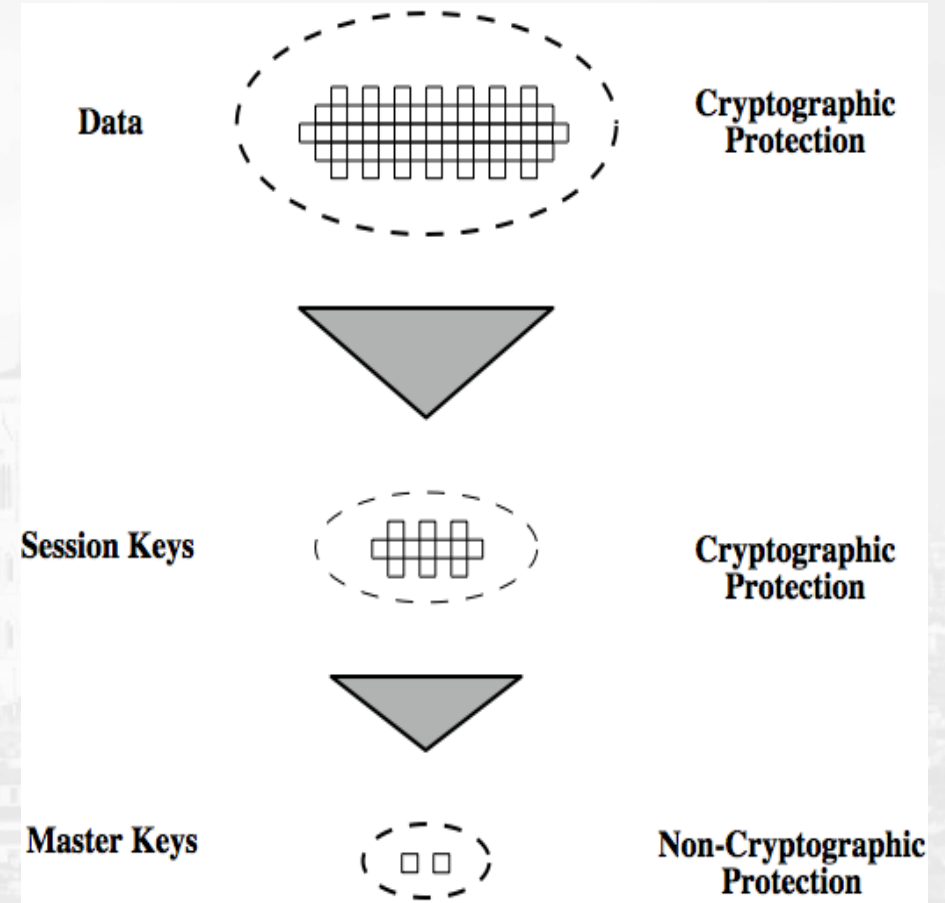
Key Hierarchy

❖ The use of a **KDC** is based on the use of a hierarchy of keys

- ❖ typically have a hierarchy of keys
- ❖ session key
 - temporary key
 - used for encryption of data between users
 - for one logical session then discarded
- ❖ master key
 - used to encrypt session keys
 - shared by user & key distribution center

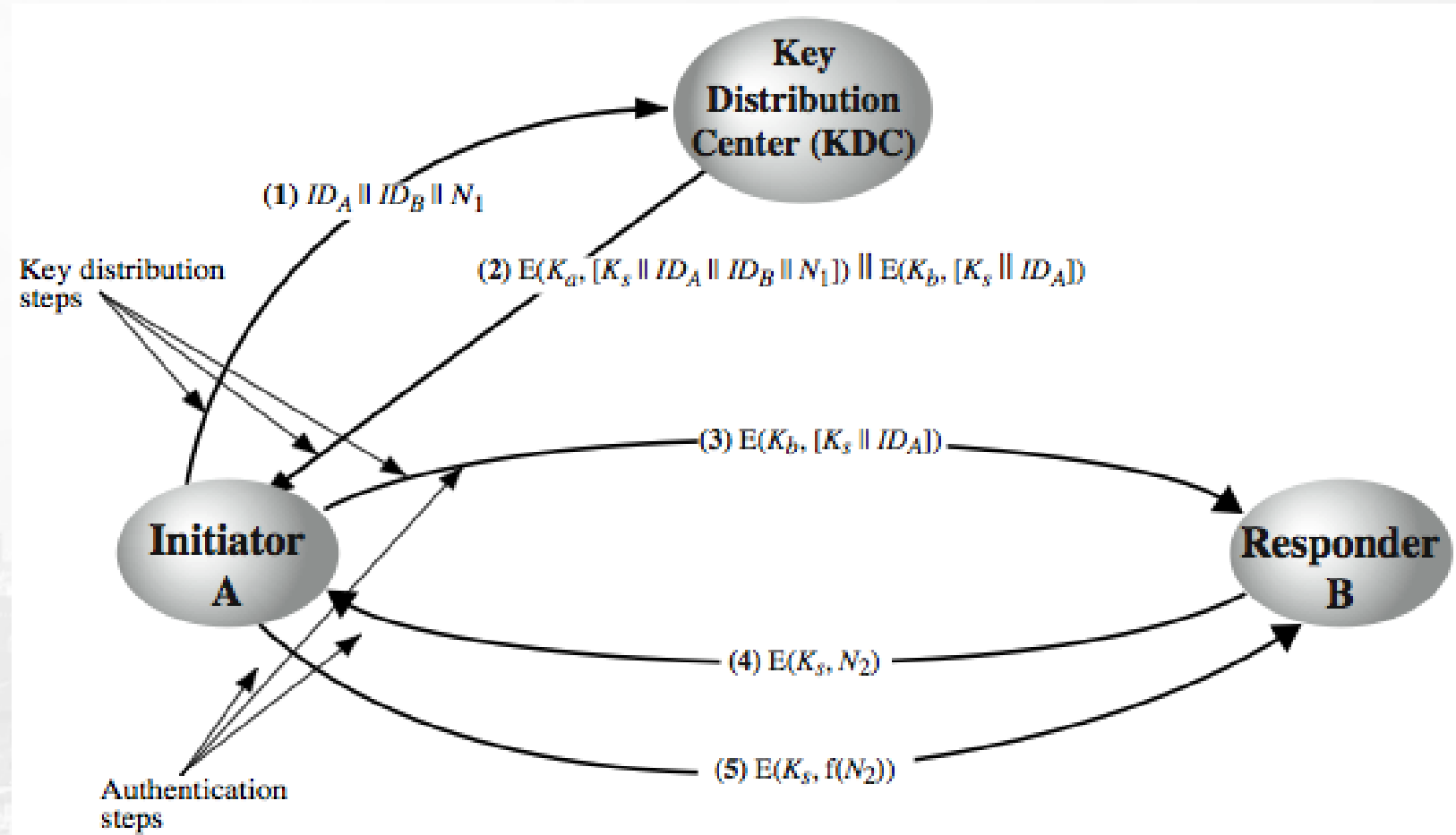
❖ $[N(N-1)]/2$ --- Session key

❖ N --- Master key



Key Distribution Scenario

❖ N_1 - Nonce i.e. timestamp, a counter or random number



Key Distribution Issues

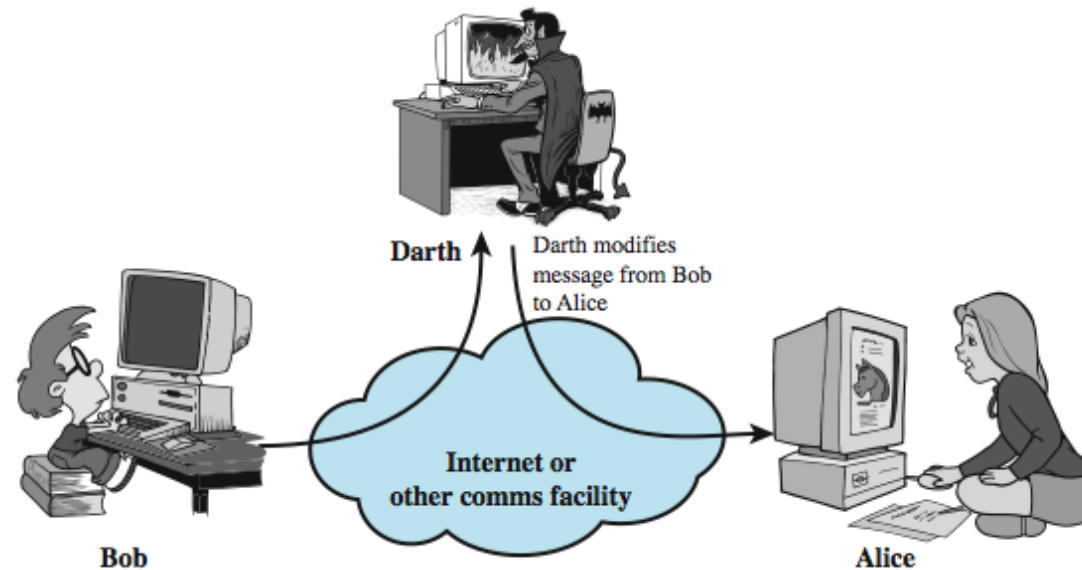
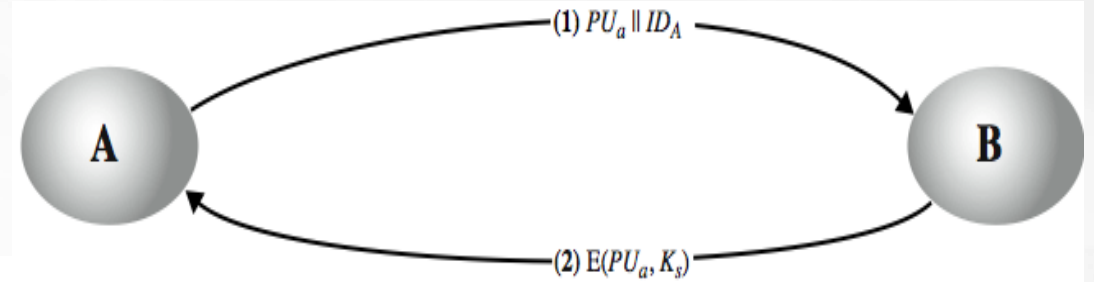
- ❖ hierarchies of KDC's required for large networks, but must trust each other
- ❖ session key lifetimes should be limited for greater security
- ❖ use of automatic key distribution on behalf of users, but must trust system

Symmetric Key Distribution Using Public Keys

- ❖ public key cryptosystems are inefficient
 - so almost never use for direct data encryption
 - rather use to encrypt secret keys for distribution

Simple Secret Key Distribution

- ❖ Merkle proposed this very simple scheme-
 - allows secure communications
 - no keys before/after exist

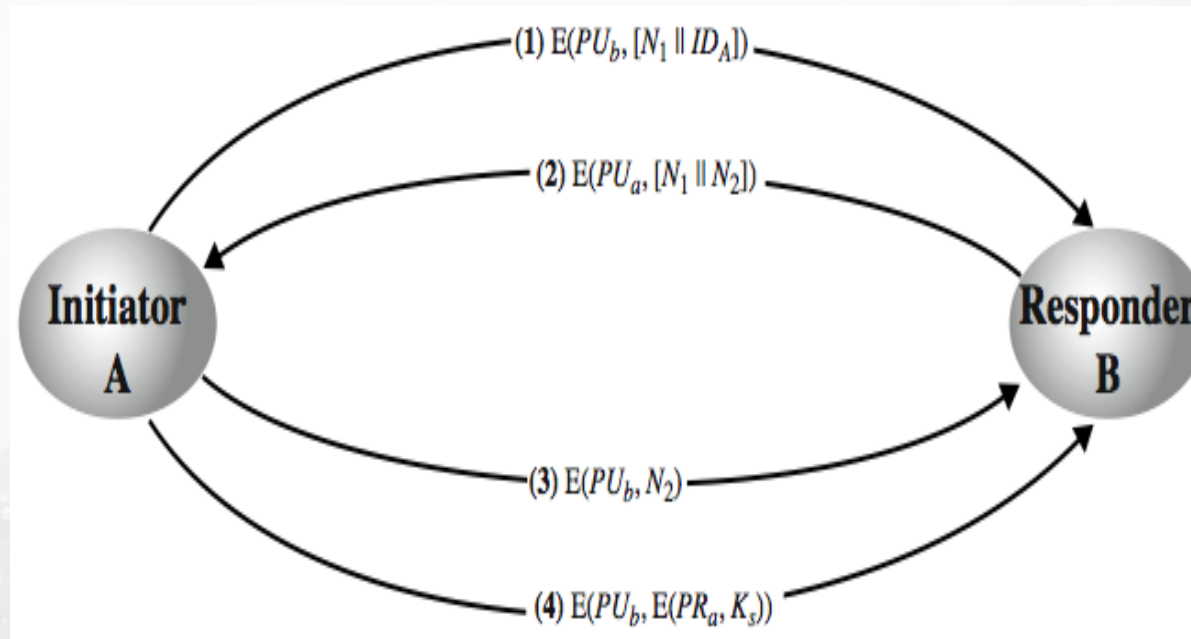


Man-in-the-Middle Attack

- ❖ this very simple scheme is vulnerable to an active man-in-the-middle attack
- 1. **A** generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a message intended for **B** consisting of and an identifier of **A**, ID_A .
- 2. **E** intercepts the message, creates its own public/private key pair $\{PU_e, PR_e\}$ and transmits $PU_e // ID_A$ to **B**.
- 3. **B** generates a secret key, K_s , and transmits $E(PU_e, K_s)$.
- 4. **E** intercepts the message and learns by computing $D(PR_e, E(PU_e, K_s))$.
- 5. **E** transmits $E(PU_a, K_s)$ to **A**.

Secret Key Distribution with Confidentiality and Authentication

- ❖ Provides protection against both active and passive attacks



Distribution of Public Key

- Two aspects to the use of public-key encryption:
 - The distribution of public keys.
 - The use of public-key encryption to distribute secret keys.
- ❖ Public announcement
- ❖ Publicly available directory
- ❖ Public-key authority
- ❖ Public-key certificates

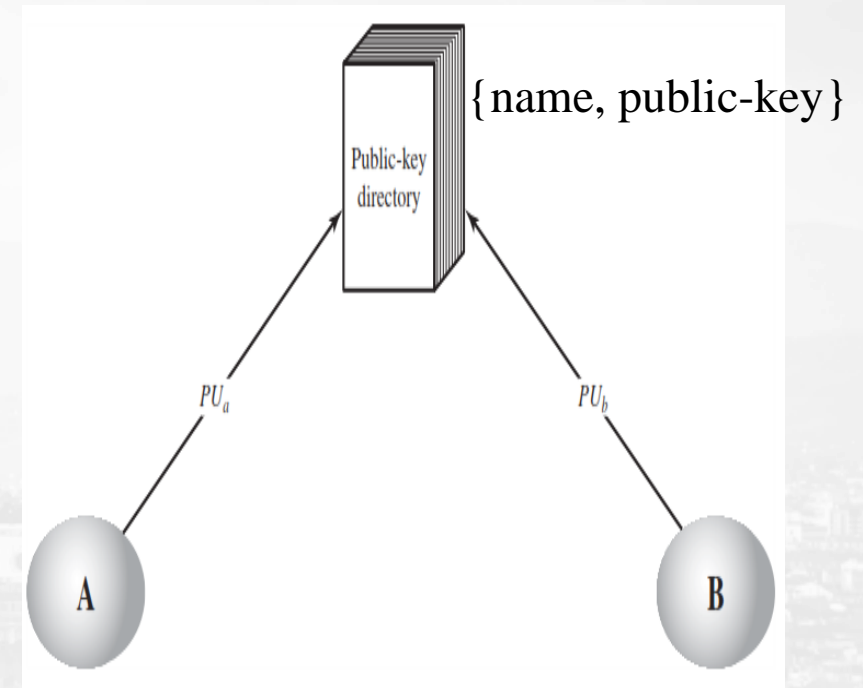
Public Announcement

- ❖ users distribute public keys to recipients or broadcast to community at large
 - eg. append PGP keys to email messages or post to news groups or email list
- ❖ 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



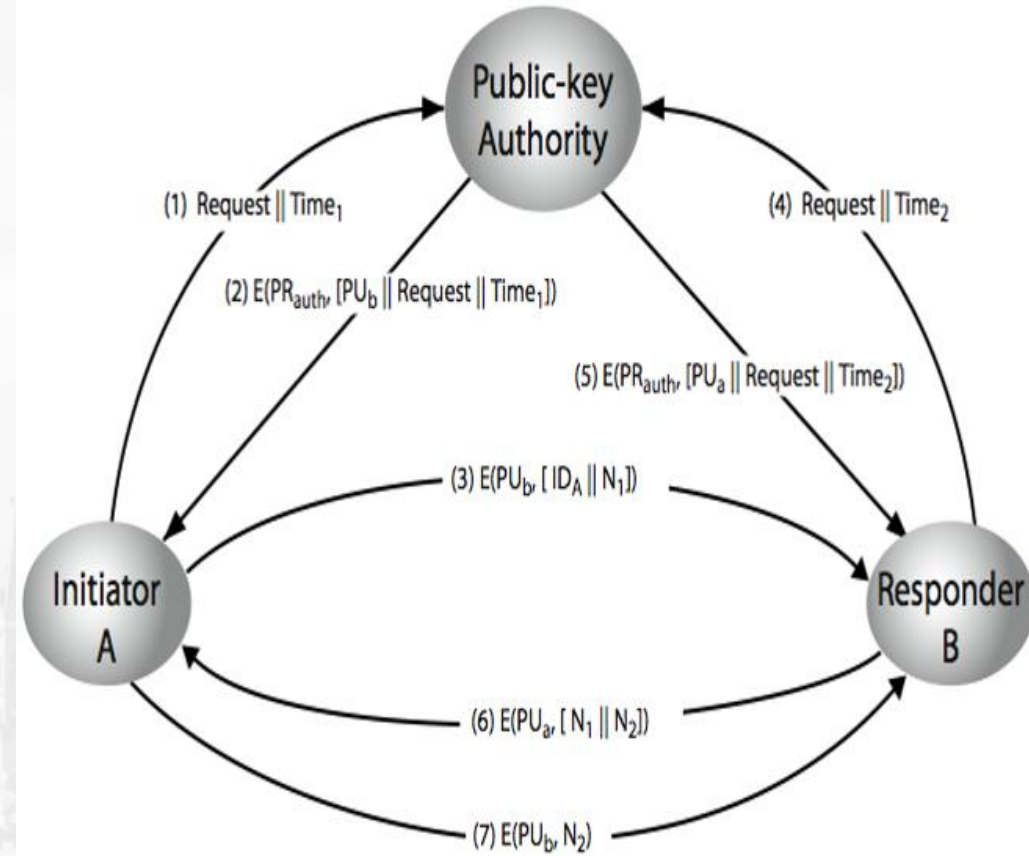
Publicly Available Directory

- ❖ can obtain greater security by registering keys with a public directory
- ❖ directory must be trusted with properties:
 - contains {name, public-key} entries
 - 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



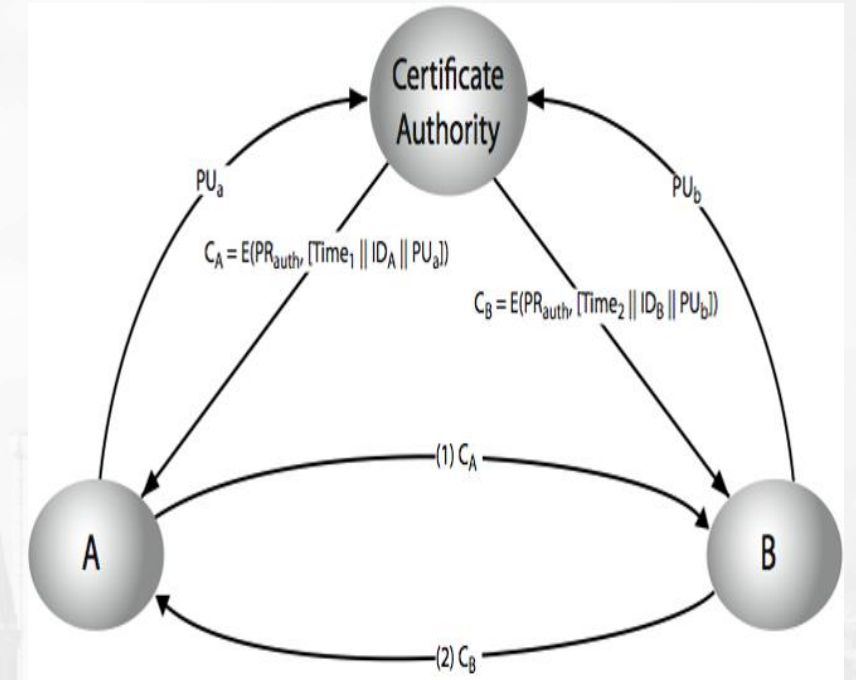
Public-Key Authority

- ❖ improve security by tightening control over distribution of keys from directory
 - ❖ has properties of directory
 - ❖ and requires users to know public key for the directory
 - ❖ then users interact with directory to obtain any desired public key securely
 - does require real-time access to directory when keys are needed
 - may be vulnerable to tampering
- User must appeal to the authority for a public key for every other user that it wishes to contact which makes the **system slow**.



Public-key Certificates

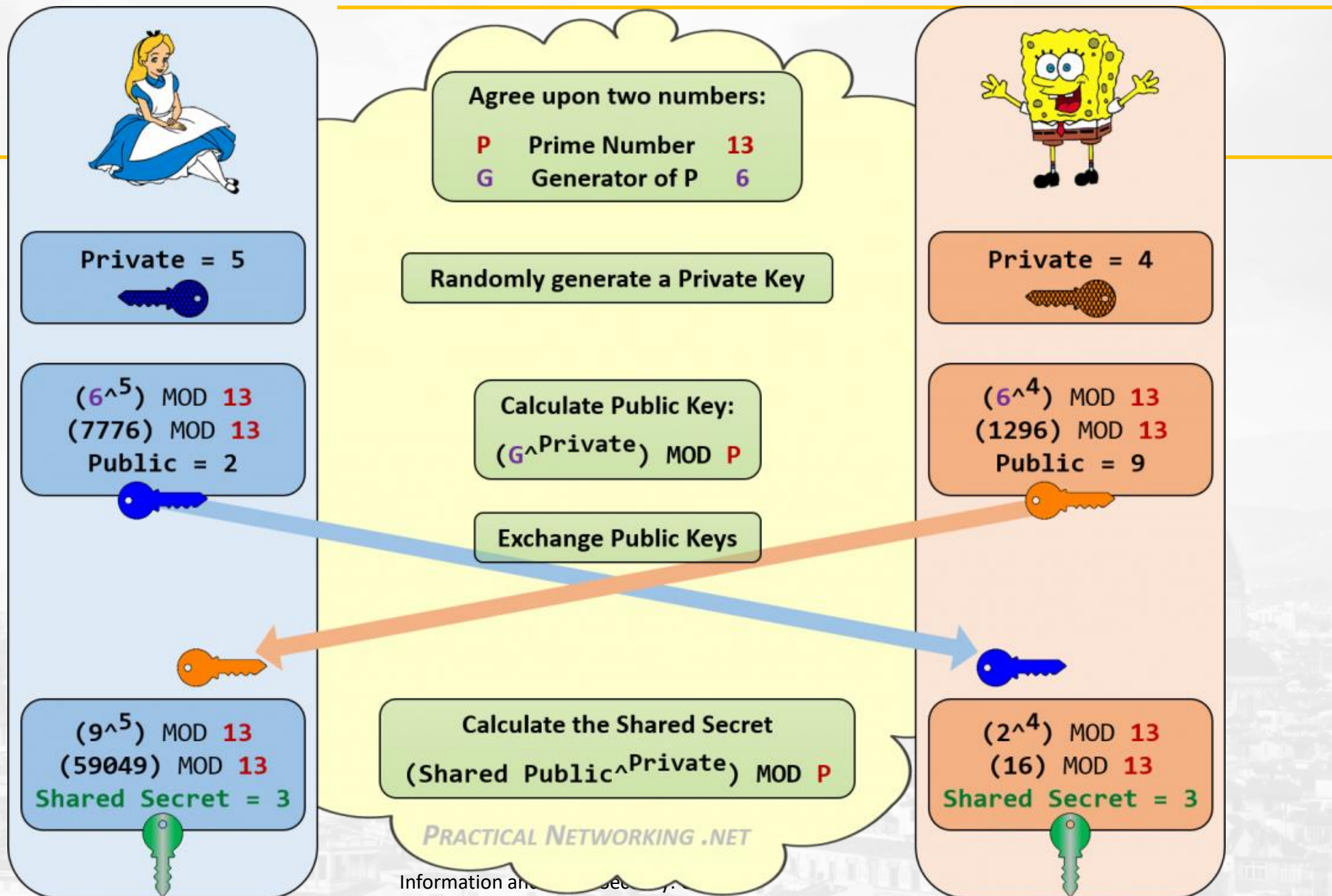
- ❖ certificates allow key exchange without real-time access to public-key authority
- ❖ a certificate binds identity to public key
 - usually with other info such as period of validity, rights of use etc
- ❖ with all contents signed by a trusted Public-Key or Certificate Authority (CA)
- ❖ can be verified by anyone who knows the public-key authorities public-key



Diffie-Hellman Key Exchange

1. Choose two **prime** numbers **n** and **g** where g is a primitive root of n.
2. User A selects **X_A** as his **private key randomly**. i.e. **$X_A < n$**
3. User B selects **X_B** as his **private key randomly**. i.e. **$X_B < n$**
4. User A computes his **public key** i.e. **$Y_A = (g^{X_A}) \bmod n$**
5. User B computes his **public key** i.e. **$Y_B = (g^{X_B}) \bmod n$**
6. Exchange their public keys
7. User A computes key called **shared secret key**. i.e. **$k = (Y_B^{X_A}) \bmod n$**
8. User B computes key called **shared secret key**. i.e. **$k = (Y_A^{X_B}) \bmod n$**
9. Both user communicate each other using one of the **symmetric encryption technique**.

They use shared secret key as the encryption key for selected algorithm.



Advantages

1. The Diffie-Hellman algorithm is mostly used for key exchange.
2. The sender and receiver don't need any prior knowledge of each other.
3. Once the keys are exchanged, the communication of data can be done through an insecure channel.

Disadvantages

1. Can not be used to encrypt message
2. As there is no authentication involved, it is vulnerable to man-in-the-middle attack.
3. It can not be used for digital signatures.

Examples

1. Solve if $p = 17$ and $q = 7$ using Diffie Hellman algorithm. Assume $A = 5$, and $B = 3$

a is primitive root of p if

$$\left. \begin{array}{l} a \bmod p \\ a^2 \bmod p \\ \vdots \\ a^{q-1} \bmod p \end{array} \right\} \text{ values is } \{1, 2, 3, \dots, p-1\}$$

⇒ Example 1: -
 $p = 17, q = 7$

$X_A = 5$	$X_B = 3$
public key	
$Y_A = q^{X_A} \bmod p$	$Y_B = 7^3 \bmod 17$
$Y_A = 7^5 \bmod 17$	$Y_B = 3$
$= 11$	

Exchange the key.
calculate private secret key.

For A	For B
$K = 3^5 \bmod 17$	$K = X_A^{X_B} \bmod p$
$K = Y_B^{X_A} \bmod p$	$K = 11^3 \bmod 17$
$K = 5$	$= 5$
\therefore shared secret key is 5	

2. Find the key exchanged between Alok and Bobby considering following data:

$$n = 11, g = 5, X = 2, Y = 3$$

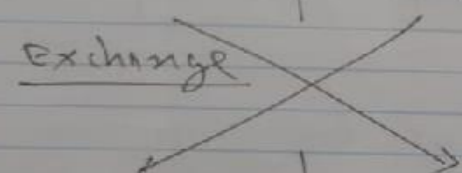
Find values of A, B and secret key K

It means, we need to calculate public key and secret key.

Public key

$A = g^X \mod n$ $= 5^2 \mod 11$ $= 3$	$B = g^Y \mod n$ $= 5^3 \mod 11$ $= 4$
--	--

Exchange



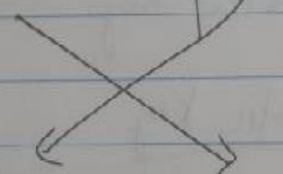
shared secret key

$K = B^X \mod n$ $= 4^2 \mod 11$ $= 5$	$K = A^Y \mod n$ $= 3^3 \mod 11$ $= 5$
--	--

3. If generator $g = 2$ and n or $p = 11$ using Diffie Hellman algorithm, solve the following:
- Show that 2 is primitive root of 11
 - If A has private key 9 what is A's public key ?
 - If B has private key 3 what is B's public key ?
 - Calculate shared secret key

(ii) Public key

$A = 2^9 \mod 11$ $= 6$	$B = 2^3 \mod 11$ $= 8$
----------------------------	----------------------------



Secret Key

$k = 8^9 \mod 11$ $= 7$	$k = 6^3 \mod 11$ $k = 7$
----------------------------	------------------------------

4. Users Alice and Bob use the Diffie-Hellman key exchange technique with a common prime $q = 83$ and a primitive root $\alpha = 5$.
 - a. If Alice has a private key $X_A = 6$, what is Alice's public key Y_A ?
 - b. If Bob has a private key $X_B = 10$, what is Bob's public key Y_B ?
 - c. What is the shared secret key?

5. Consider a Diffie-Hellman scheme with a common prime $q = 13$ and a primitive root $\alpha = 7$.
 - a. Show that 7 is a primitive root of 13.
 - b. If Alice has a public key $Y_A = 5$, what is Alice's private key X_A ?
 - c. If Bob has a public key $Y_B = 12$, what is the secret key K shared with Alice?

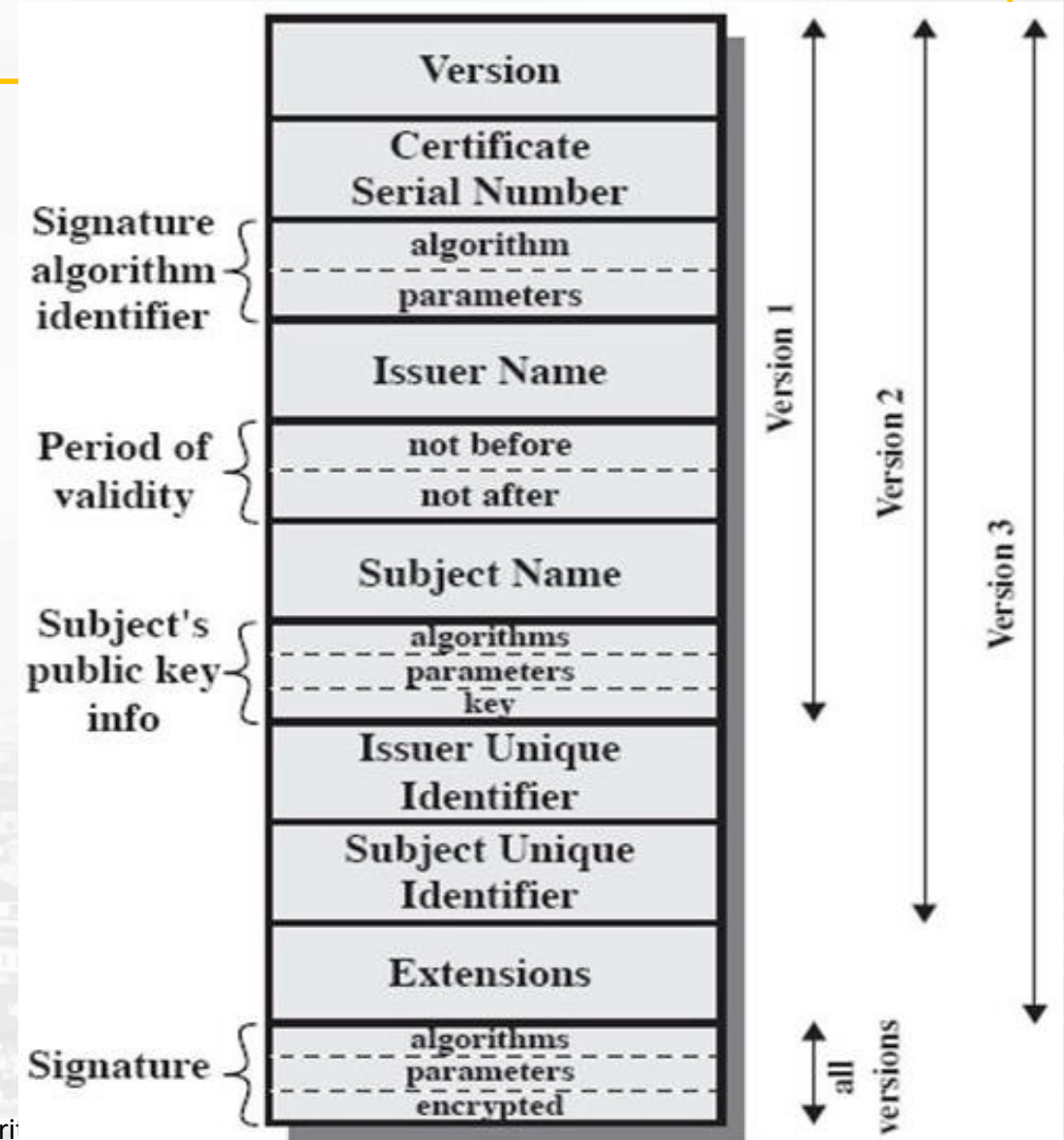
6. Consider a Diffie-Hellman scheme with a common prime $q=11$ and a primitive root $\alpha=2$.
 - a. If user A has public key $Y_A = 9$, what is A's private key X_A ?
 - b. If user B has public key $Y_B = 3$, what is B's private key X_B , what is the shared secret key k , shared with A ?

Digital Certificate

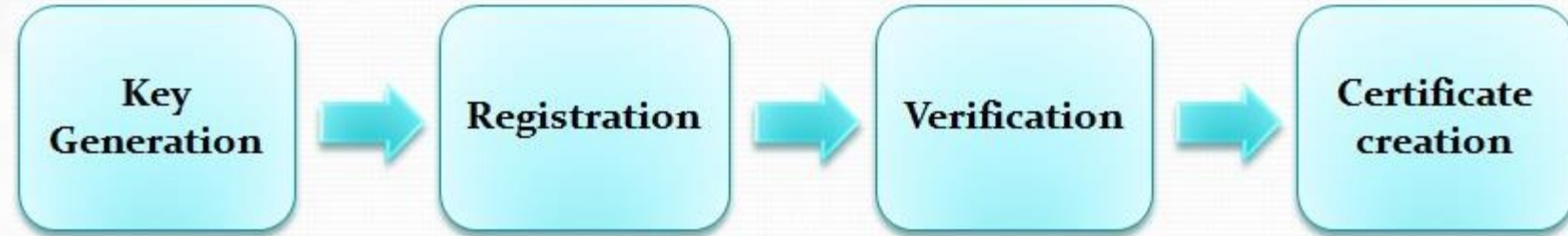
- ❖ A digital certificate is simply a small computer file
- ❖ Certificates are the framework for identification information, and bind identities with public keys. e.g. passport, driving license
- ❖ They provide a foundation for:
 - identification
 - authentication and
 - non-repudiation.

Structure of X.509 digital certificate

- ❖ Trusted organization (i.e. **Certificate Authority** (CA)) that issues certificates and maintains status information about certificates.
- ❖ The most popular CA's are **Verisign and Entrust**.
- ❖ A Standard called **X.509** define a structure of a digital certificate.
- ❖ CA issues new certificates, maintain old ones, and revoke the certificate that has become invalid for some sort of reasons, etc.
- ❖ The CA can delegate some of its tasks to this third-party called as a **Registration Authority (RA)**.



Certificate Creation Steps



Key generation:

- It starts with the creation of the subject's public and private keys using some software.
- The subject then sends the public key along with the other information like evidence about himself/herself to the **RA (Registration Authority)**.

Registration:

- Suppose the user has created the key pair, the user now sends the public key and the related registration information.
- The format for the certificate requests has been standardized and is called **certificate signing request (CSR)**. This is one of the public key cryptography standards (PKCS).

Verification:

- When the registration process is completed, the RA has to check the user's credentials such as the provided information is correct and acceptable or not.
- RA checking the **Proof Of Possession (POP)** of the private key.

Certificate creation:

- The RA accepts all the details of the user and send to the CA. The CA does its own verification (if required) and creates a digital certificate for the user.
- There are programs for creating certificates in the **X.509 standard** format. The CA delivers the certificate to the user and also keep a copy of the certificate for its own record.
- The CA's copy of the certificate is maintained in a **certificate directory**.

PHPki Certificate Authority

[Menu](#) [Public](#) [Policy](#) [Help](#) [About](#)

Certificate Request Form

Common Name
(i.e. User full name, Computer
host name)

Bill Doe

E-mail Address

bdoe@somewhere.com

Organization (Company/Agency)

ACME Hazardous Waste Disposal

Department/Unit

Administrative Services

Locality (City/County)

Centerville

State/Province

Kansas

Country

US

Certificate Password

Again

Certificate Life

1 Year ▼

Key Size

1024 bits ▼

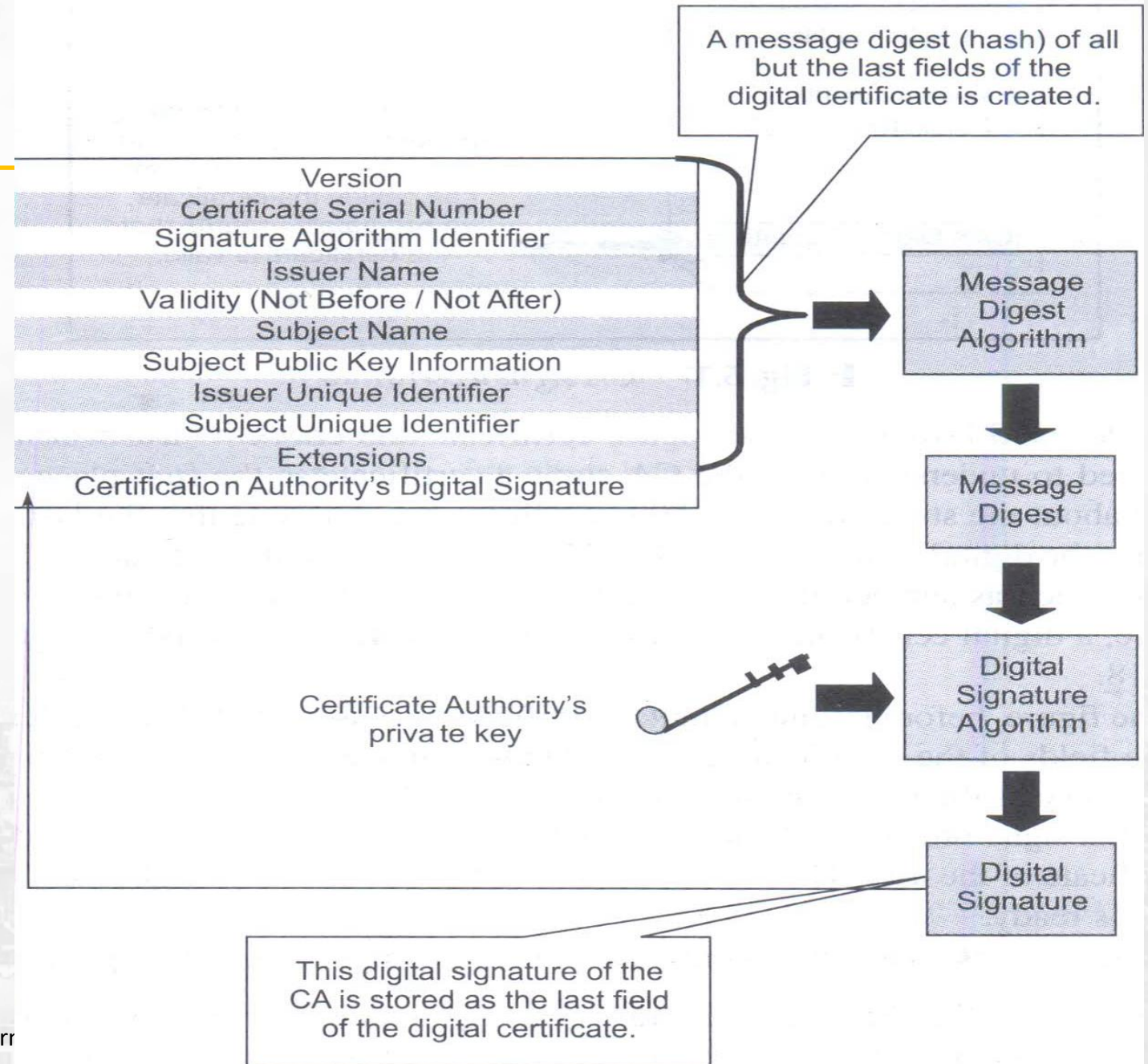
Certificate Use:

E-mail ▼

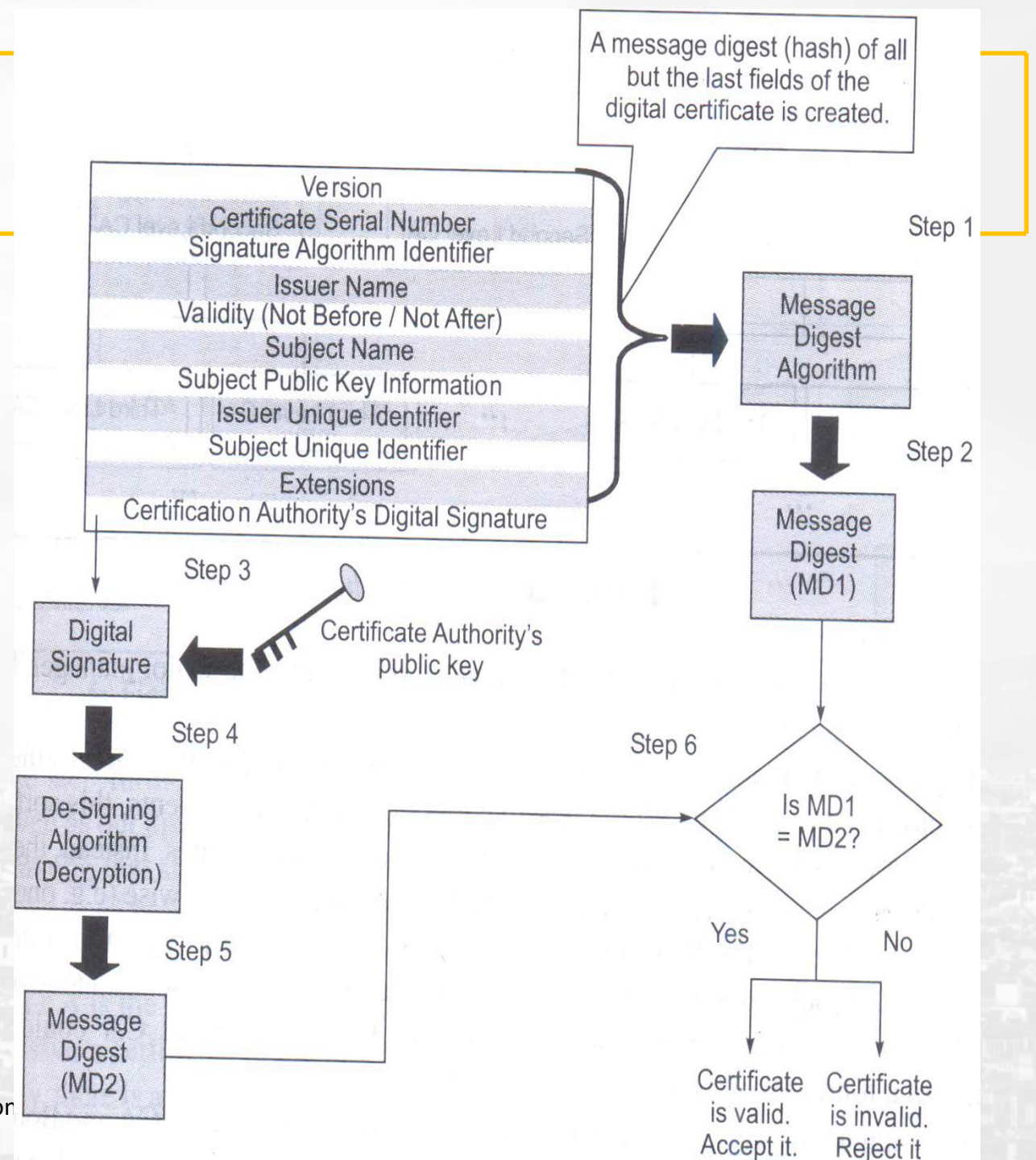
Submit Request

* All fields are required

Creation of the CA signature on a certificate

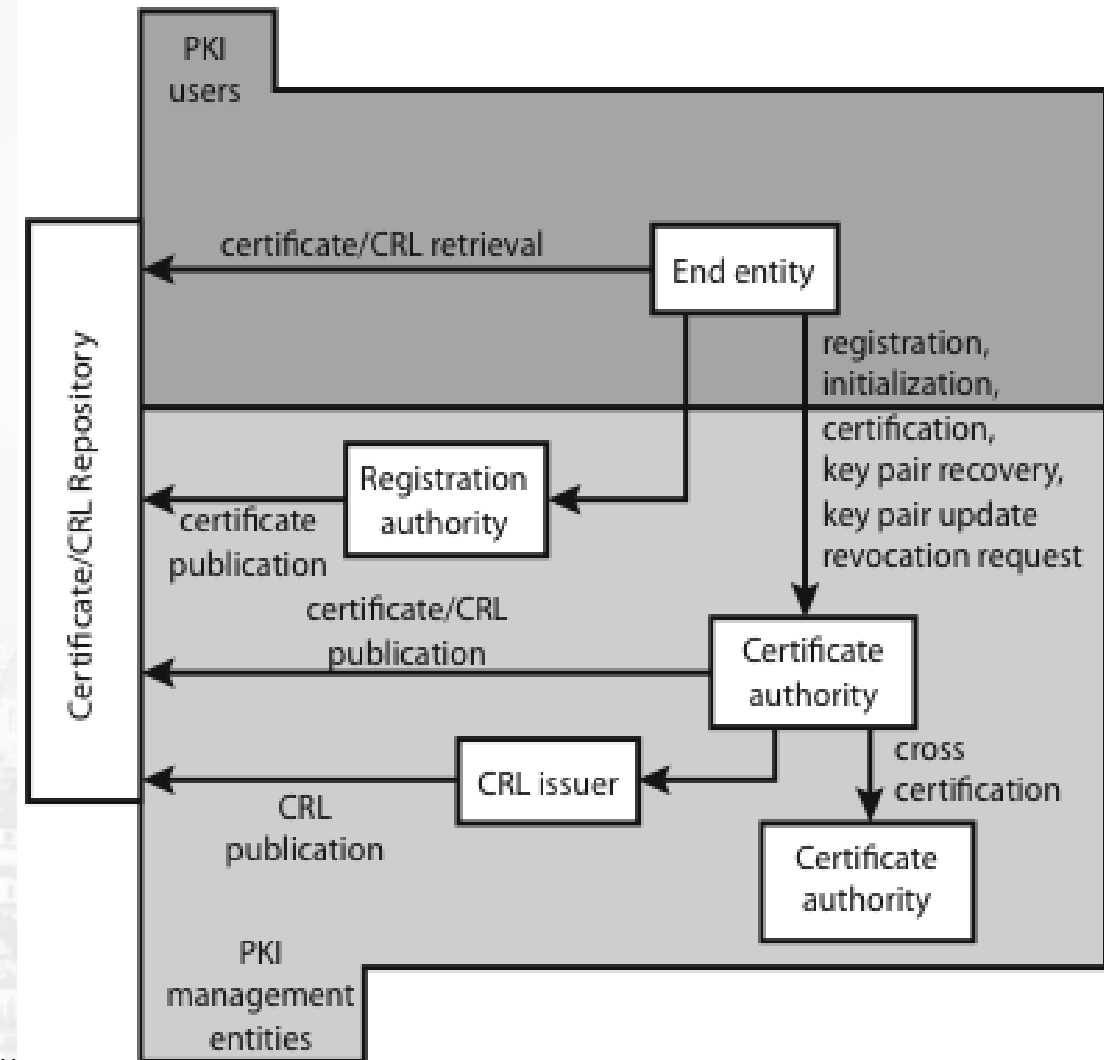


Verification of the CA signature on a certificate



Public Key Infrastructure (PKI)

- ❖ PKI is the set of hardware, software, policies, processes, and procedures required to create, manage, distribute, use, store, and revoke digital certificates using asymmetric cryptography.
- ❖ A certificate revocation list (or CRL) is "a list of digital certificates that have been revoked by the issuing certificate authority (CA) before their scheduled expiration date and should no longer be trusted



Authentication Protocols

❑ Remote User - Authentication Principles

- Entity authentication is a technique designed to let one party prove the identity of another party
- An entity can be a person, a process, a client, or a server
- The entity whose identity needs to be proved is called the claimant
- The party that tries to prove the identity of the claimant is called the verifier

❖ has two steps:

- **identification** - specify identifier
- **verification** - bind entity (person) and identifier

❖ Verification Categories:

- Something known e.g. password, PIN
- Something possessed e.g. key, token, smartcard
- Something inherent e.g. fingerprint, retina, voice, sign

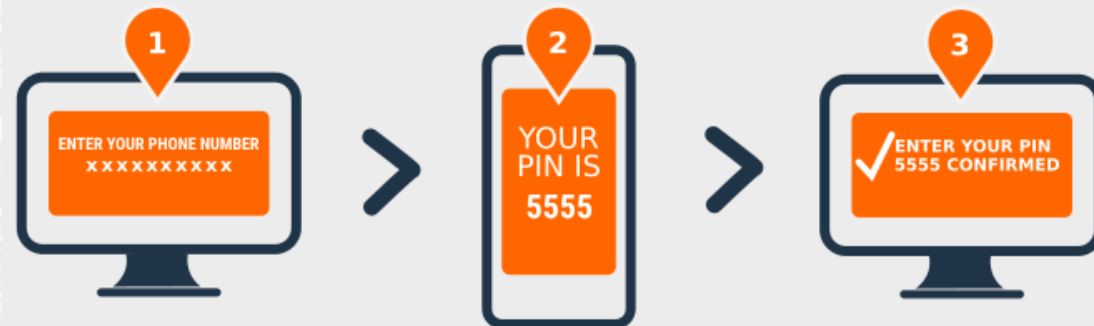
Authentication Protocol

One Way Authentication

Mutual Authentication

❑ One Way Authentication

- ❖ One-factor Authentication e.g. password
- ❖ Two-factor Authentication e.g. ATM card and PIN



❑ Mutual Authentication

- ❖ Protocols which enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys
- ❖ To achieve the goal of mutual authentication most of the protocols depends on an authentication server also called **Key Distribution Center (KDC)**.
- ❖ Authentication server also maintain a table containing a name and a **master key or secret key** of each client.

Key exchanges issues:

- Confidentiality – **masquerade attack**
- timeliness – Replay attack

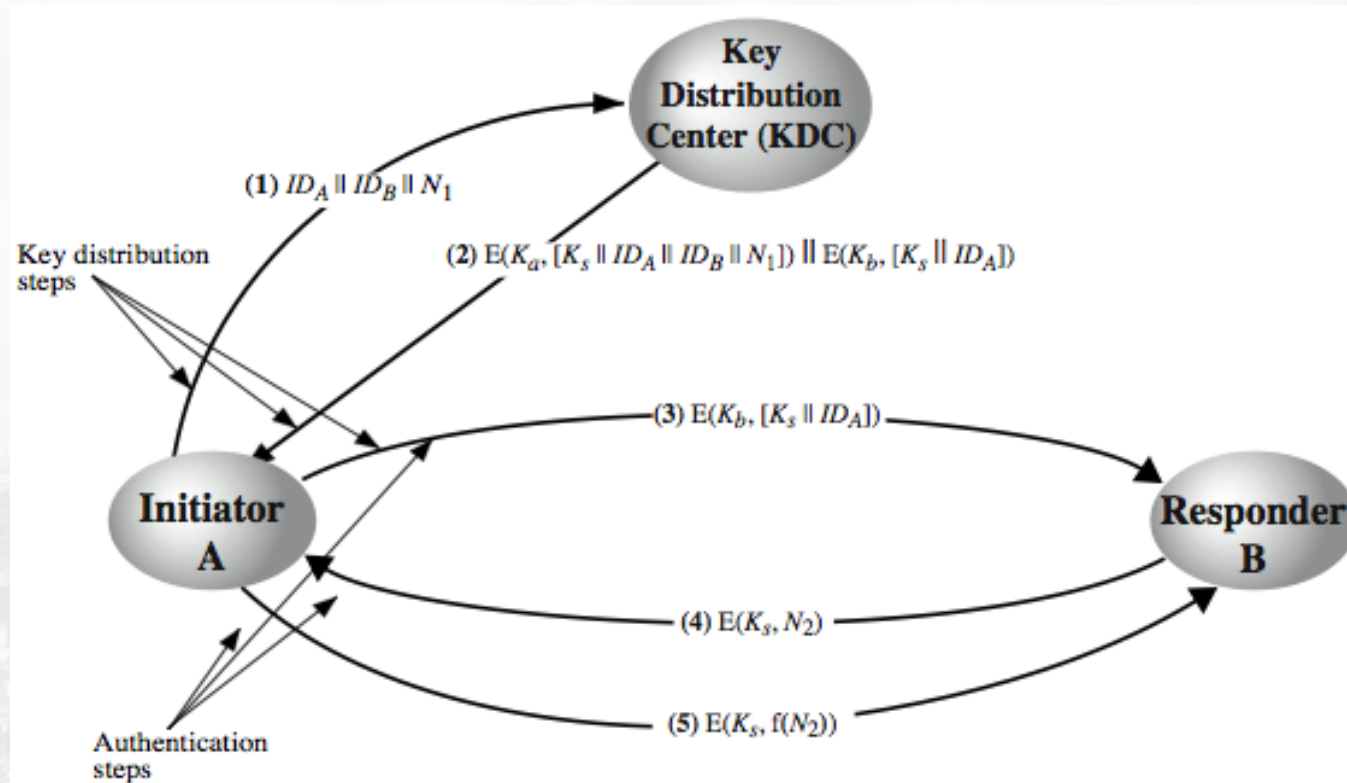
Replay Attack:

- use of **sequence numbers** (generally impractical)
- **timestamps** (needs synchronized clocks)
- **challenge/response** (using unique nonce)

Remote User-Authentication Using Symmetric Encryption

❑ Mutual Authentication

❖ Published in 1978 by **Needham-Schroeder** authentication protocol



- ❖ 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

protocol overview is:

1. A \rightarrow KDC: $ID_A \parallel ID_B \parallel N_1$
 2. KDC \rightarrow A: $E(K_a, [K_s \parallel ID_B \parallel N_1 \parallel E(K_b, [K_s \parallel ID_A])])$
 3. A \rightarrow B: $E(K_b, [K_s \parallel ID_A])$
 4. B \rightarrow A: $E(K_s, [N_2])$
 5. A \rightarrow B: $E(K_s, [f(N_2)])$
- } Prevent Replay attack

- ❖ 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
- ❖ modifications to address this require:
 - timestamps in steps 2 & 3 (Denning 81)
 - using an extra nonce (Neuman 93)

Denning 81:

1. A \rightarrow KDC: $ID_A \parallel ID_B$
2. KDC \rightarrow A: $E(K_a, [K_s \parallel ID_B \parallel T \parallel E(K_b, [K_s \parallel ID_A \parallel T])])$
3. A \rightarrow B: $E(K_b, [K_s \parallel ID_A \parallel T])$
4. B \rightarrow A: $E(K_s, [N_I])$
5. A \rightarrow B: $E(K_s, [f(N_I)])$

❖ When sender's clock is ahead of the intended recipient's clock – Problem occurs??

Neuman 93:

1. A \rightarrow B: $ID_A \parallel N_a$
2. B \rightarrow KDC : $ID_B \parallel N_b \parallel E(K_b, [ID_A \parallel N_a \parallel T_b])$
3. KDC \rightarrow A: $E(K_a, [ID_B \parallel N_a \parallel K_s \parallel T_b]) \parallel E(K_b, [ID_A \parallel K_s \parallel T_b]) \parallel N_b$
4. A \rightarrow B: $E(K_b, [ID_A \parallel K_s \parallel T_b]) \parallel E(K_s, N_b)$

❑ One Way Authentication

- ❖ To avoid the requiring recipient (B) on line at time sender (A)
 - drop steps 4 & 5
- ❖ protocol becomes:

1. A->KDC:	$ID_A \parallel ID_B \parallel N_I$
2. KDC -> A:	$E(K_a, [K_s \parallel ID_B \parallel N_I \parallel E(K_b, [K_s \parallel ID_A])])$
3. A -> B:	$E(K_b, [K_s \parallel ID_A]) \parallel E(K_s, M)$

- ❖ provides encryption & some authentication
- ❖ does not protect from replay attack

Remote User-Authentication Using Asymmetric Encryption

❑ Mutual Authentication

- ❖ assumes both parties have other's public keys
 - may not be practical
- ❖ have Denning protocol using timestamps
- ❖ uses central authentication server (AS) to provide public-key certificates
- ❖ requires synchronized clocks

1. A->AS: $ID_A \parallel ID_B$

2. AS -> A: $E(PR_{as}, [ID_A \parallel PU_a \parallel T]) \parallel E(PR_{as}, [ID_B \parallel PU_b \parallel T])$

3. A -> B: $E(PRa, [ID_A \parallel PU_a \parallel T]) \parallel E(PR_{as}, [ID_B \parallel PU_b \parallel T]) \parallel E(PU_b, E(PRa, [Ks \parallel T]))$

- ❖ have Woo and Lam protocol using nonces
- ❖ care needed to ensure no protocol flaws

❑ One Way Authentication

- ❖ have public-key approaches for email
 - encryption of message for confidentiality, authentication, or both
 - must now public keys
 - using costly public-key alg on long message
- ❖ for confidentiality encrypt message with one-time secret key, public-key encrypted
- ❖ for authentication use a digital signature
 - may need to protect by encrypting signature
- ❖ use digital certificate to supply public key

A → B: $E(PU_b, K_s) \parallel E(K_s, M)$

A → B: $M \parallel E(PR_a, H(M))$

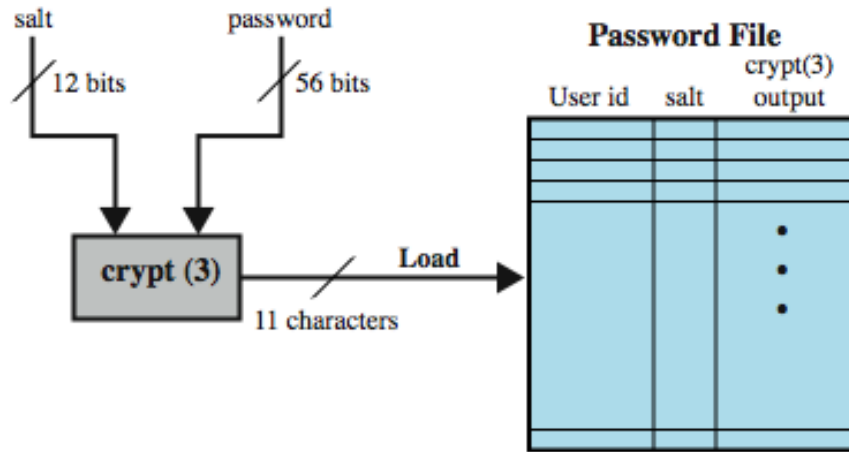
Password vulnerabilities

- The password serves to authenticate the ID of the individual logging on to the system.
- Password are themselves to vulnerable to attacks.
- ❖ Offline dictionary attack
- ❖ Specific account attack (user john)
- ❖ Popular password attack (against a wide range of IDs)
- ❖ Password guessing against single user (w/ previous knowledge about the user)
- ❖ Workstation hijacking
- ❖ Exploiting user mistakes
- ❖ Exploiting multiple password use
- ❖ Electronic monitoring

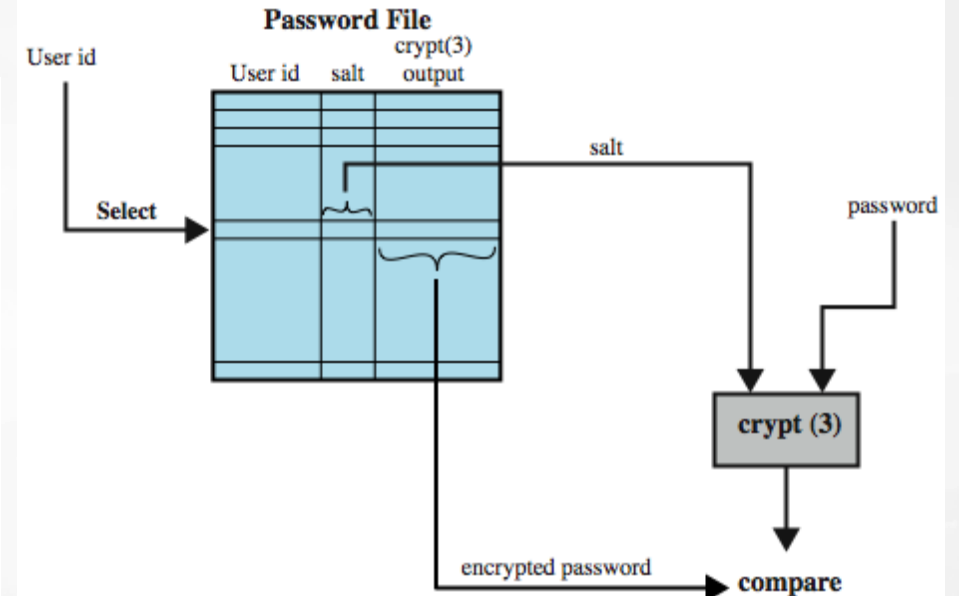
Countermeasures for password vulnerability

- ❖ Stop unauthorized access to password file
- ❖ Intrusion detection measures
- ❖ Account lockout mechanisms
- ❖ Policies against using common passwords but rather hard to guess passwords
- ❖ Training & enforcement of policies
- ❖ Automatic workstation logout
- ❖ Encrypted network links

Use of hashed passwords



(a) Loading a new password



(b) Verifying a password

Why a salt value?

- Prevents duplicate passwords from being visible in the password file
- Increases the difficulty of offline dictionary attacks
- Nearly impossible to tell if a person used the same password on multiple systems

Kerberos Security System

- Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users.
- Kerberos relies exclusively on symmetric encryption, making no use of public-key encryption.

Most popular versions of Kerberos:

- Version 4
- Version 5

Kerberos Requirement:

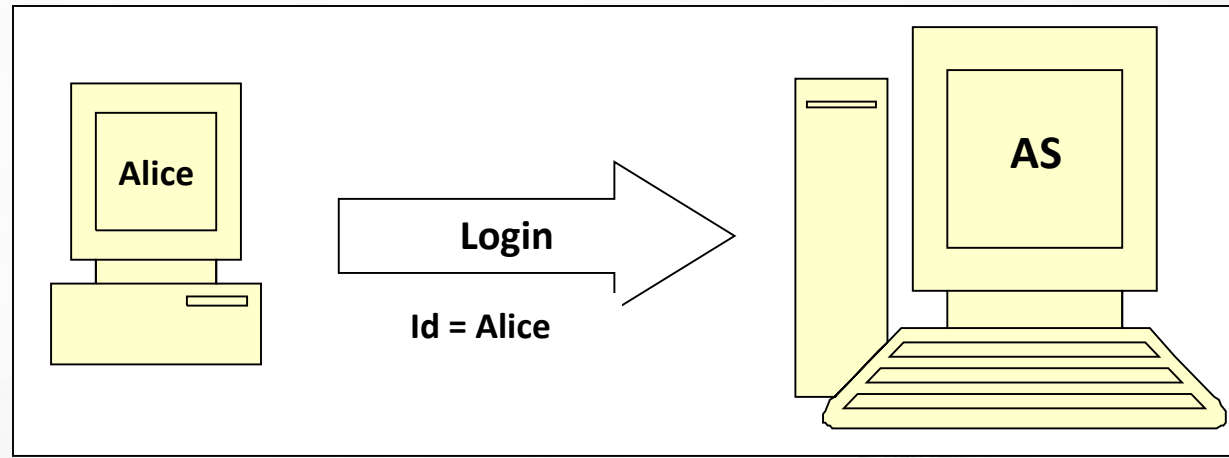
- Secure
- Reliable
- Transparent
- Scalable

The basis for Kerberos is another protocol called as Needham-Shroeder.

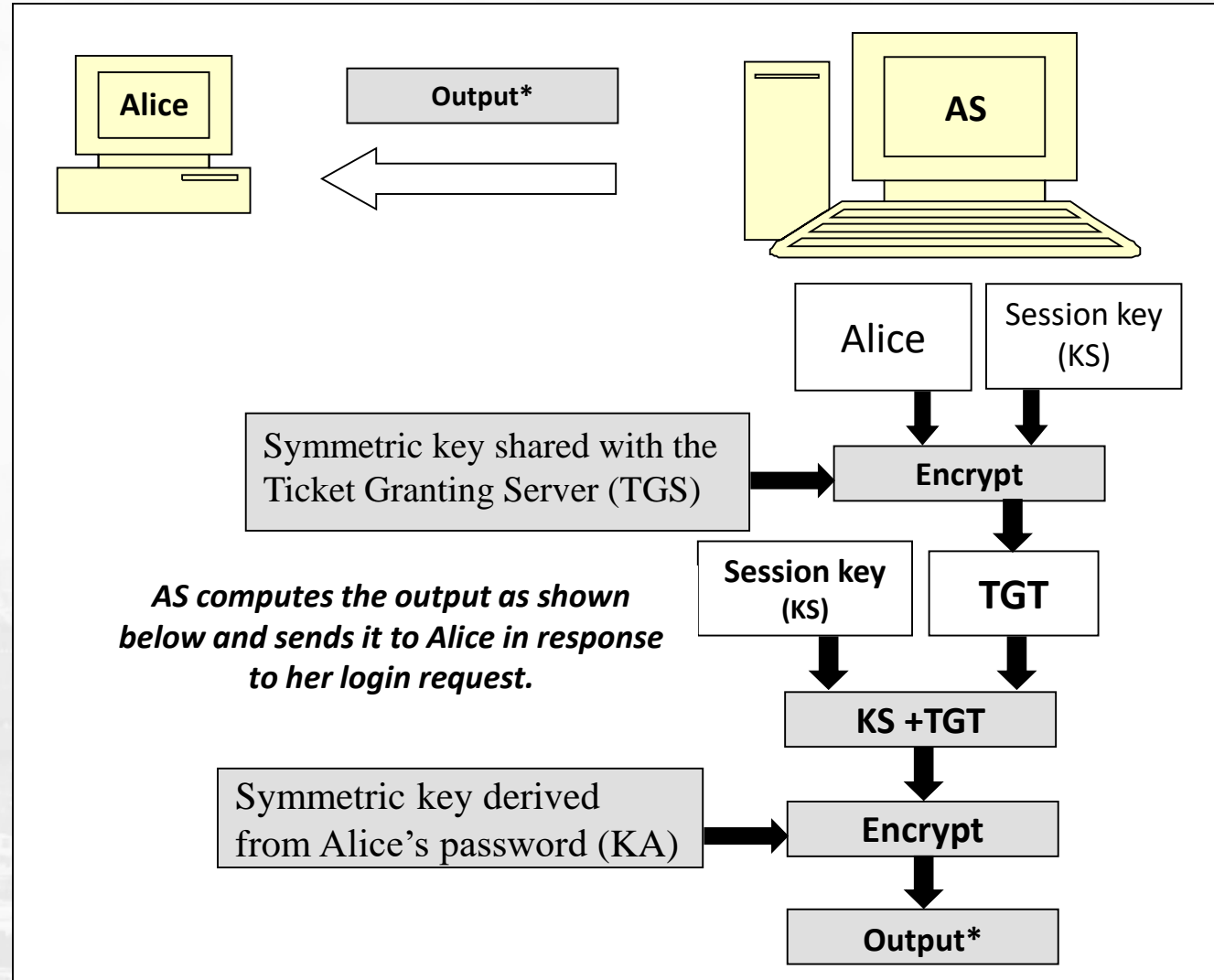
How Kerberos Works???

- **Alice:** The client workstation
- **Authentication Server (AS):** Verifies (authenticates) the user during login
- **Ticket Granting Server (TGS):** Issues **tickets** to certify *proof of identity*
- **Bob:** The server offering services such as network printing, file sharing or an application program
- The job of **AS** is to authenticate every user at the login time and shares a unique secret password with every user.
- The job of **TGS** is to certify to the servers in the network.

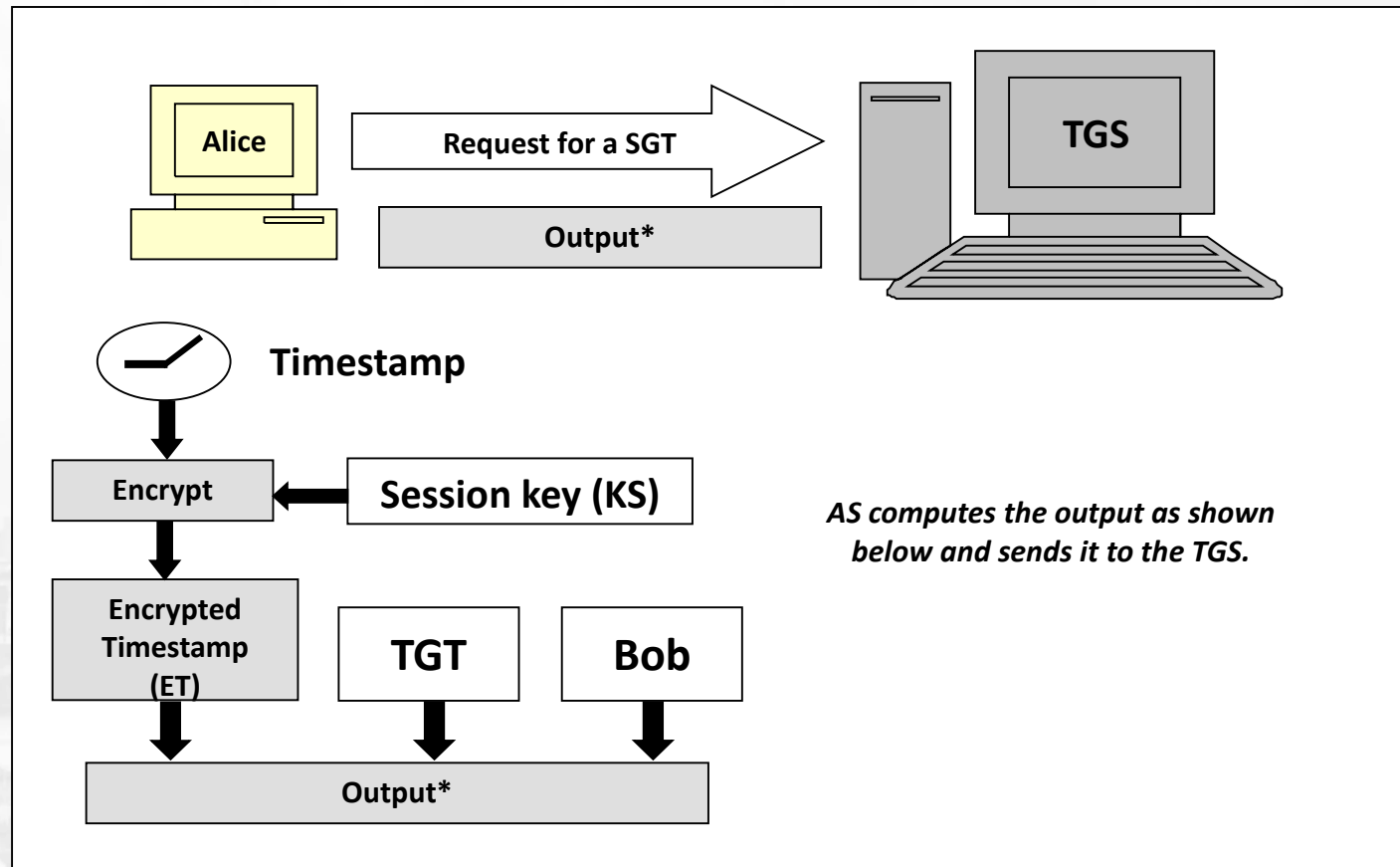
❖ Step 1: Login



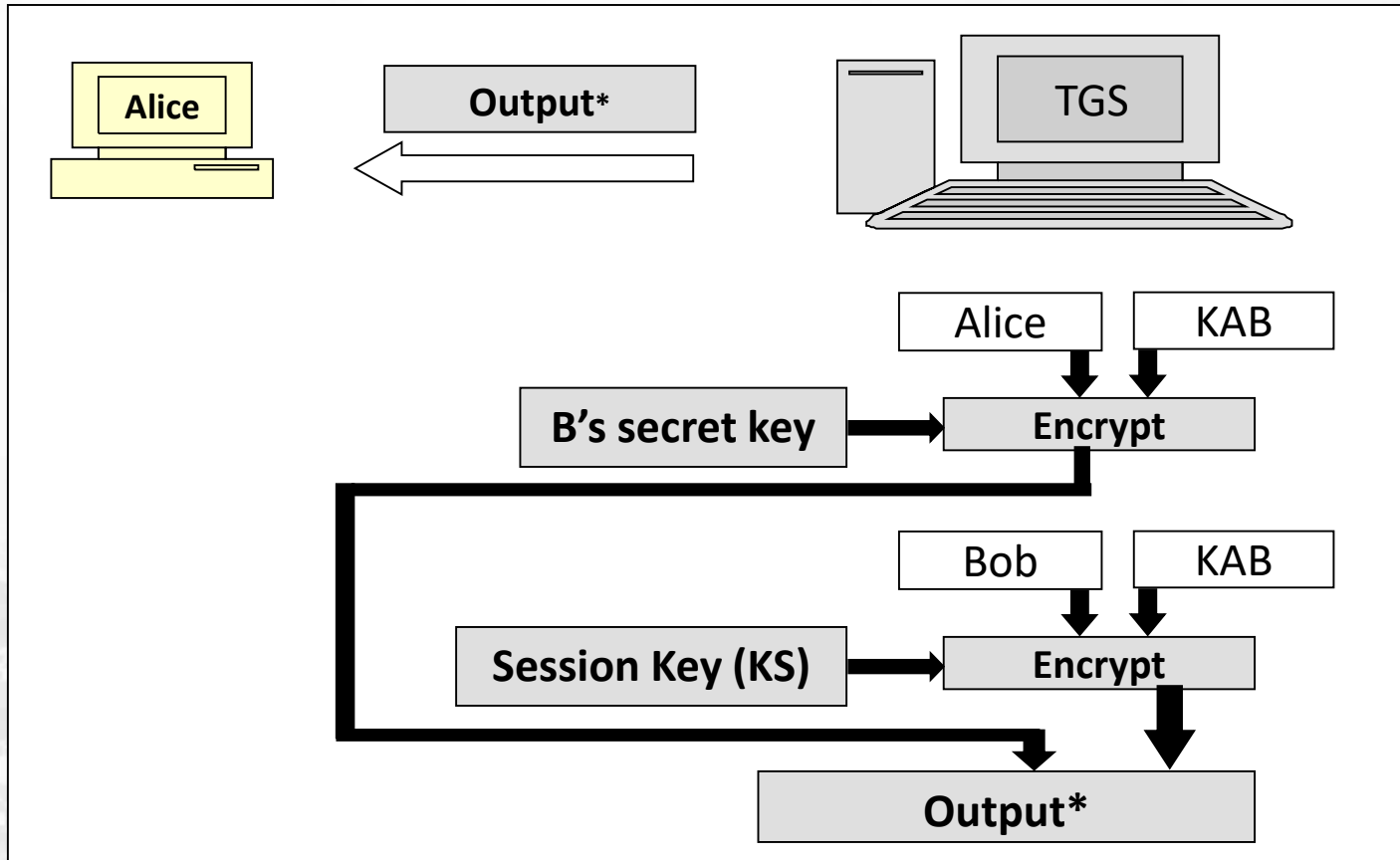
❖ Step 2: Encrypted session key and TGT



❖ Step 3: Obtaining the service granting ticket (SGT)

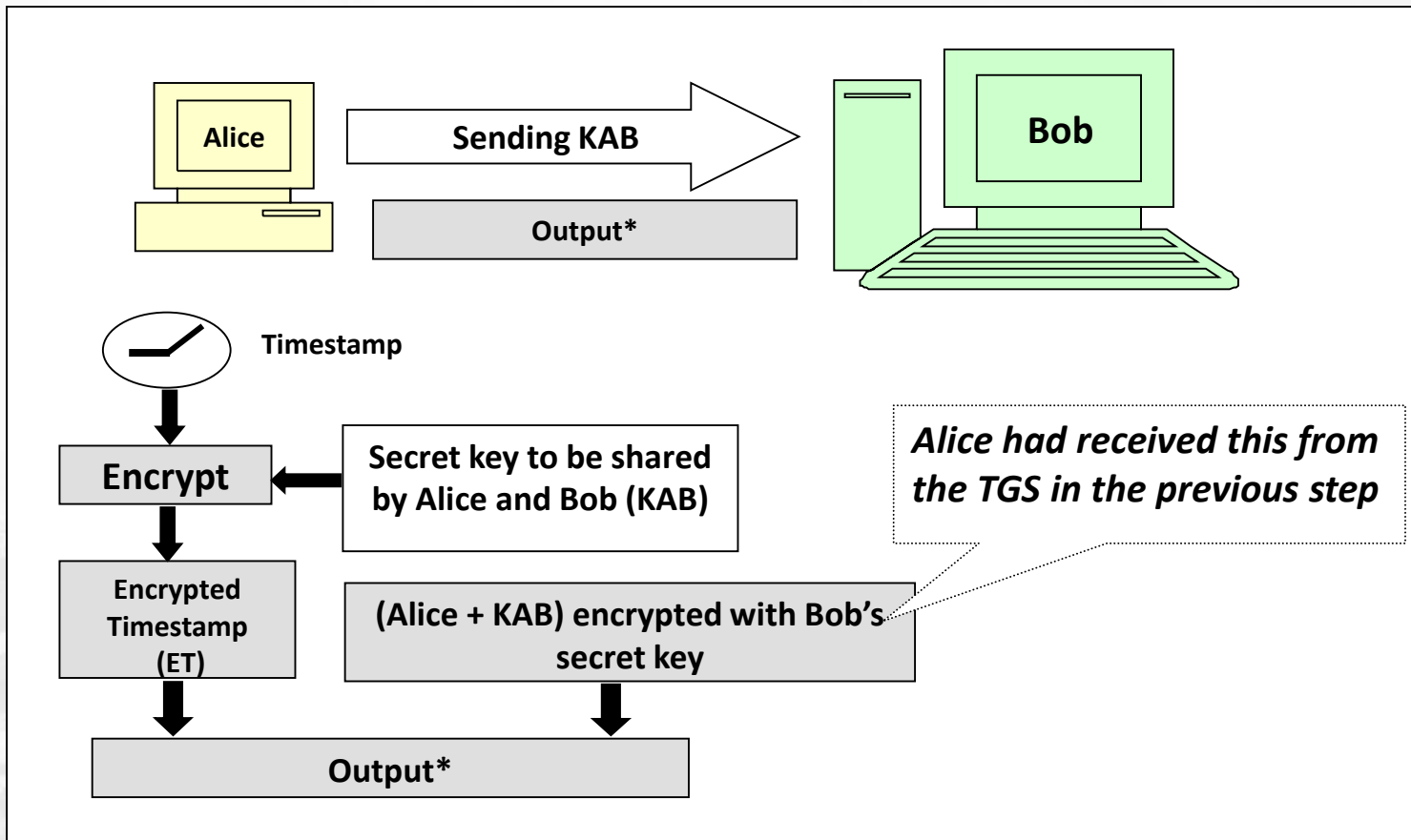


❖ Step 4: TGS Response to Alice

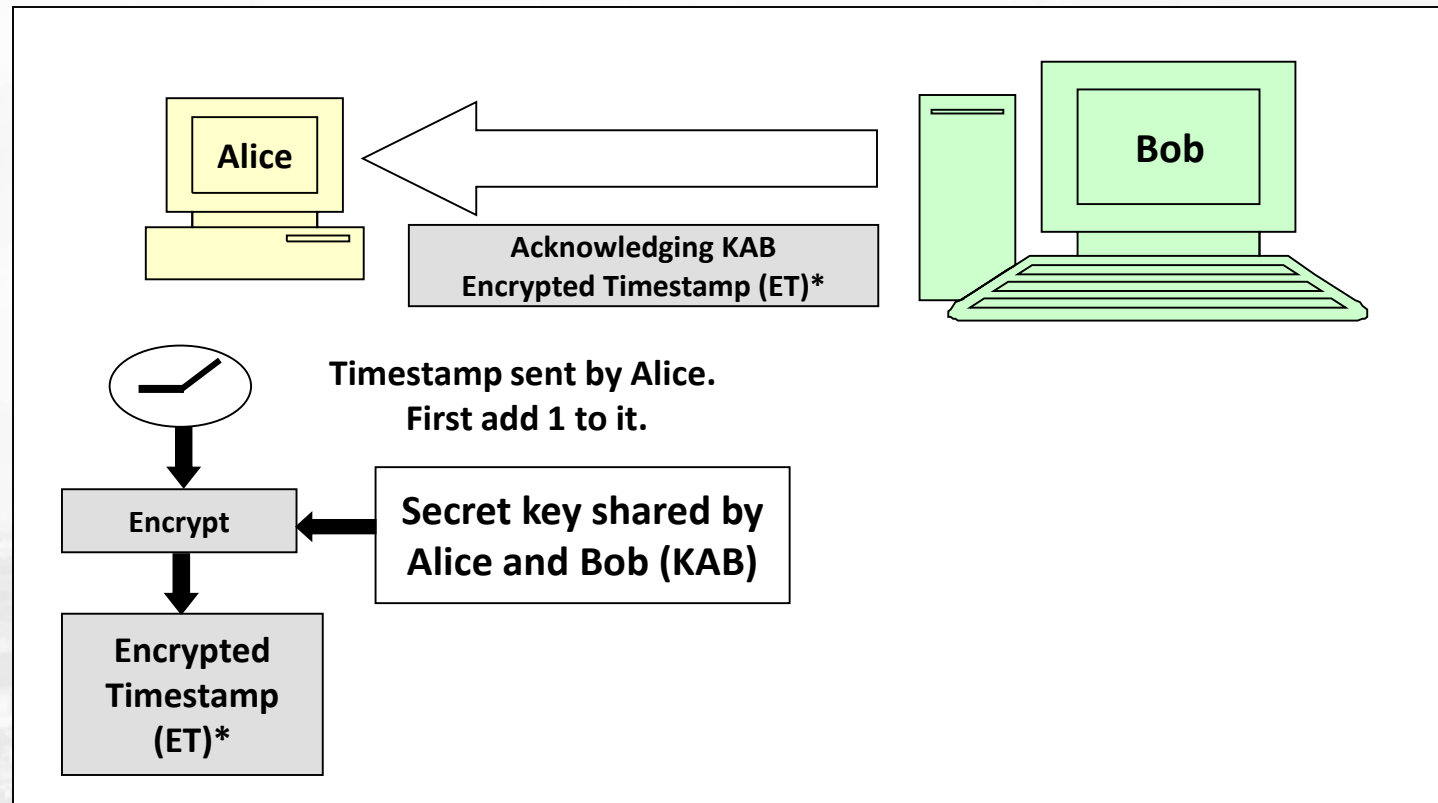


KAB - secret key of Alice and Bob

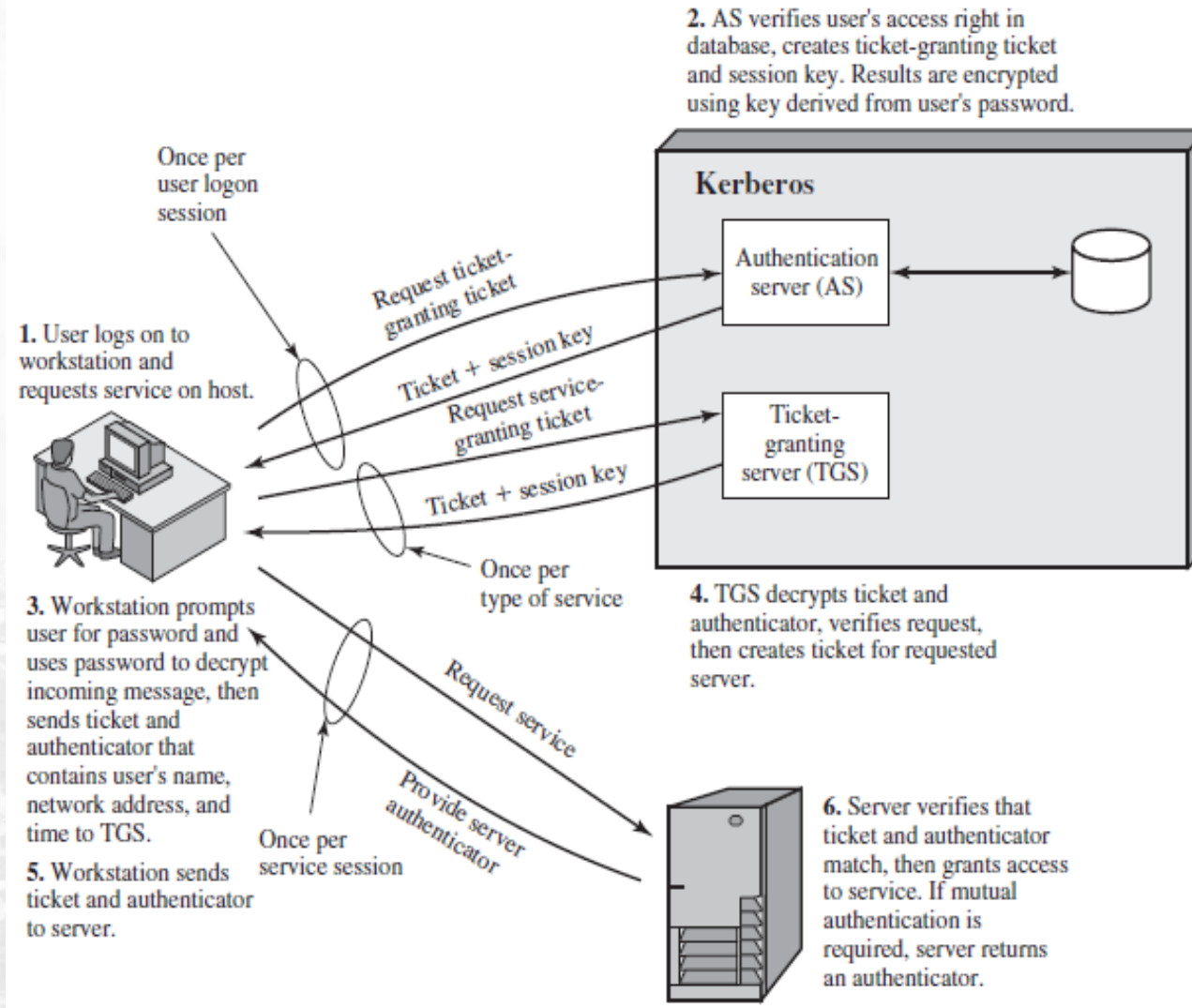
❖ Step 5 : User contacts Bob for accessing the server



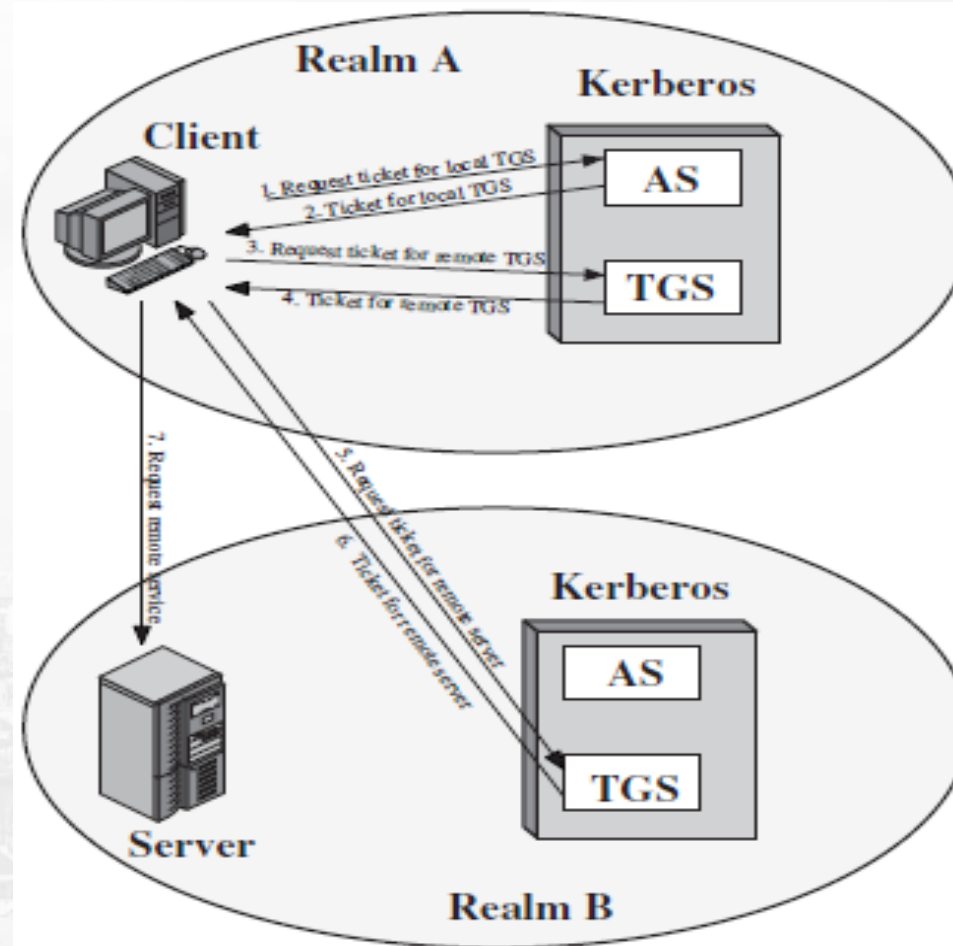
❖ Step 6: Acknowledges the receipt of KAB



Overview of Kerberos



Request for Service in Another Realm



- Kerberos version 4 uses DES to provide authentication. Version 5 allows flexibility for choice of cryptographic algorithm.
- Version 4 depends on IP addresses as identifiers. Version 5 allows the use of other types.

Thank You !!!!!!!