Digital Signature and Cryptographic Hash Function

Dr. Risala T Khan

Professor

IIT, JU

Topics to be Discussed

- Digital Signature
  - To be familiar with the general idea behind digital signature
  - To define security services provided by a digital signature
  - To describe some applications of digital signatures
- Cryptographic Hash Function
  - To introduce general ideas behind hash function
  - To discuss the usage and application of hash function
  - To know the desirable properties of a hash function

What is Digital Signature?

- Digital signature is a digital code that can be attached to an electronically transmitted message that uniquely identifies the sender and provides the integrity of the message.
  - It was first proposed in 1976 by Whitfield Diffie of Stanford University.
- Typically the signature is formed by taking the hash of the message (called message digest) and encrypting the digest with the creator's private key.
  - The encrypted message digest is known as a digital signature.
  - The signature is then added at the end of each message that is to be sent to the recipient.
  - The recipient decrypts the signature using sender's public key and verifies that the message digest is correct and the message has come from the genuine sender. If the transmitted message is changed, the digital signature is invalidated.
- Like a written signature on a document, the purpose of a digital signature is to guarantee that the individual sending the message really is who he or she claims to be.

Process of Creating Digital Signature

- The process of creating a digital signature is outlined below:

Sender Site

1. Sender generates a message.
2. He/she then creates a "digest" of the message using cryptographic hash function.
3. Sender encrypts the message digest with his/her private key for authentication. This encrypted message digest is called digital signature.
4. Sender attaches the digital signature to the end of the message that is to be sent. The message attached with digital signature is known as digitally signed message.
5. The sender encrypts the digitally signed message with the recipient's public key and sends it to the recipient.

Receiver Site

6. After receiving, the recipient decrypts the entire message with his/her private key.
7. The recipient detaches the message and digital signature.
8. He/she creates a "digest" of the received message using the same hash function the sender used.
9. The recipient decrypts the digital signature and finds the "digest" that the sender created.
10. The recipient then compares the two digests. If they are equal, the message is granted, otherwise it will be rejected.

The processes are illustrated in the figure on the next slide..

Digital Signature Process

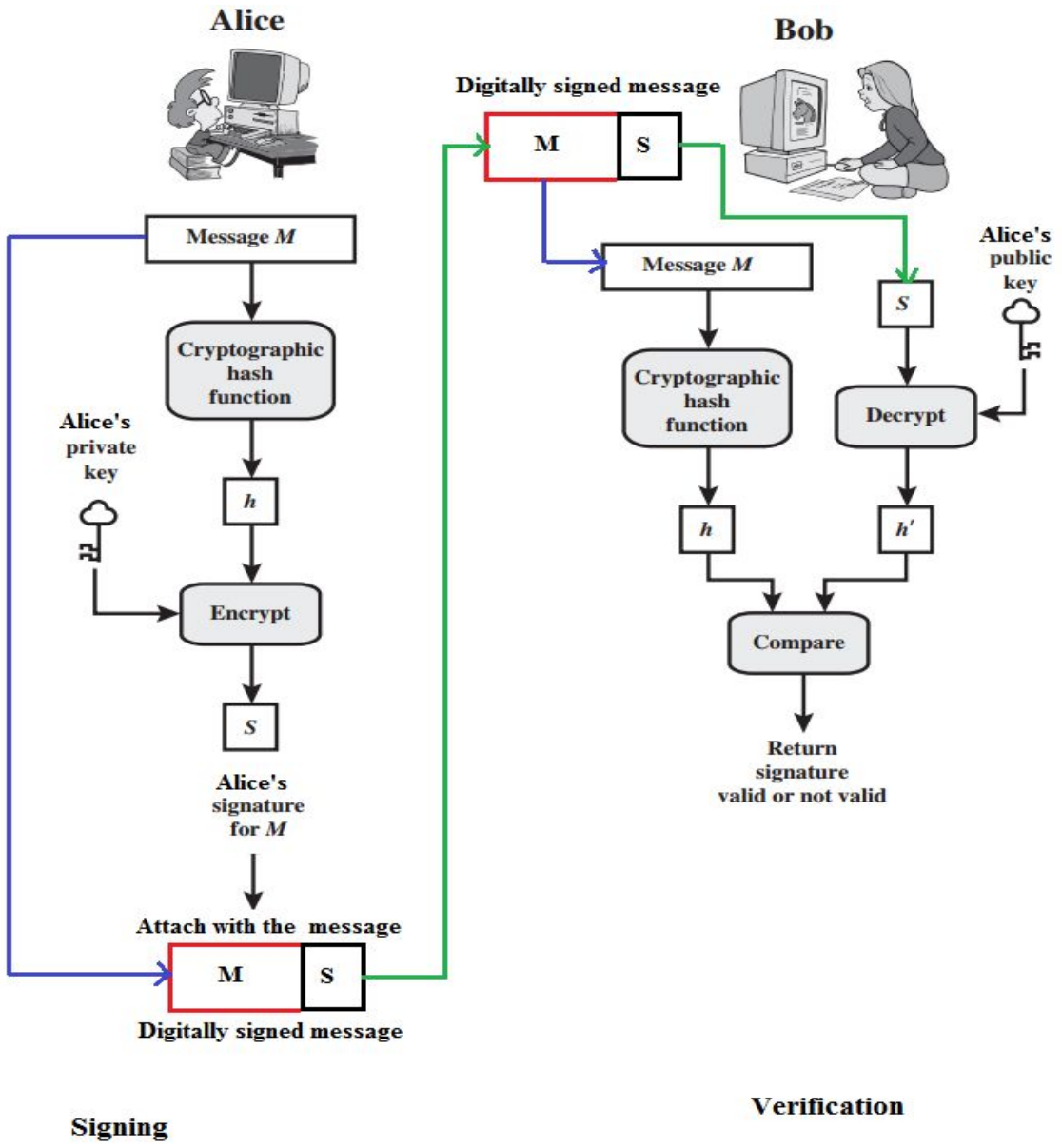


Figure: Illustration of digital signature process:

Digital Signature Vs. Conventional Signature

Key-point	Conventional Signature	Digital Signature
Inclusion	**a conventional signature is included in the document**; it is part of the document. E.g., when we write a check, the signature is on the check; it is not a separate document.	But, when we sign a document digitally, we send the signature as a separate document.; **a digital signature is a separate document. The sender sends two documents- the message and the signature.** The recipient receives both documents and verifies that the signature belongs to the supposed sender. If this is proven, the message is kept; otherwise, it is rejected.
Verification method	A conventional signature on a document is verified by comparing the signature on it with the signature on file.	For a digital signature, 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.

Digital Signature Vs. Conventional Signature

Key-point	Conventional Signature	Digital Signature
Relationship	For a conventional signature, **there is normally a one-to-many relationship between a signature and documents.** A person uses the same signature to sign many documents.	For a digital signature, **there is a one-to-one relationship between a signature and a message.** Each message has its own signature. The signature on one message can not be used in another message. For example, if Bob receives two messages, one after another, from Alice, he can not use the signature of the first message to verify the second. Each message needs a new signature.
Duplicity	In conventional signature, **a copy of the signed document can be distinguished from the original one on file**.	In digital signature, there is no such distinction unless there is a factor of time (such as a timestamp) on the document. For example, suppose Alice sends a document instructing Bob to pay Eve. If the intercepts the documents and the signature, she can replay it later to get money again from Bob.

Services Provided by Digital Signature

A digital signature serves three important purposes:

1. Provides authentication of the sender
2. Verifies data integrity
3. Provides non-repudiation

Services Provided by Digital Signature

Message Authentication:

- A digital signature's main function is to verify that a message or document, in fact, comes from the claimed sender. That is, to provide authentication is the main function of digital signature.

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. Therefore, digital signature provides the integrity of the message.

Non-Repudiation:

- Attaching a digital signature with message prevents repudiation. This ensures that the sender should not be able to later deny that he/she sent a message. Non-repudiation prevents sender and vendor in a transaction or communication activity from later falsely denying that the transaction occurred.

N.B. As contrast to encryption scheme, digital signature does not provides the confidentiality of the message.

Massage Authentication Code (MAC)

- A **Message Authentication Code (MAC)** is a short piece of information used to authenticate a message and ensure its integrity. Here's a brief overview:

1. **Purpose**: A MAC verifies that the message came from the stated sender (authenticity) and that it hasn't been altered (integrity).
2. **How It Works**:
    1. **Key Generation**: A secret key is generated and shared between the sender and the receiver.
    2. **MAC Generation**: The sender uses the secret key and a MAC algorithm to produce a MAC tag from the message.
    3. **Transmission**: The message and the MAC tag are sent to the receiver.
    4. **Verification**: The receiver uses the same secret key and MAC algorithm to generate a MAC tag from the received message and compares it with the received MAC tag. If they match, the message is verified as authentic and unaltered

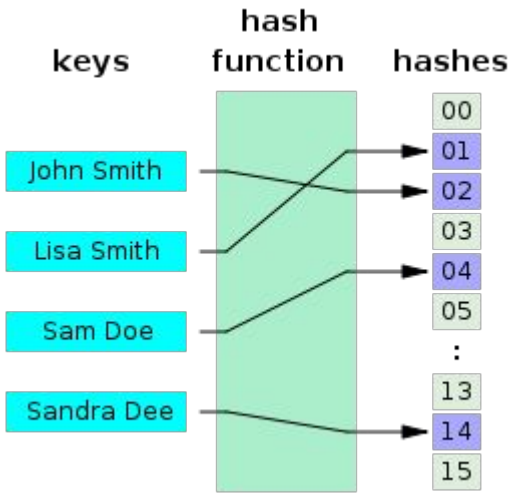
Message Authentication Code (MAC) Vs. Digital Signature

Is a message digest the same as a message authentication code?

- The MAC value protects both a message's integrity as well as its authenticity by allowing verifiers (who also possess the same secret key) to detect any changes to the message content.
- A message authentication code is different than a digital signature.
  - MAC values are both generated and verified using the same secret key.
    - While using MAC, sender and receiver of a message must agree on keys before initiating communications. As is the case with private key encryption.
  - A message authentication code does not provide the property of non-repudiation offered by digital signature.

Hash Function

- A hash function is a formula or an algorithm that-
  - takes large data sets of variable length as input, and
  - returns smaller data sets of fixed length as output.
- Since, the output is smaller than the input data, a hash function compresses an n-bit message string to create an m-bit string where n is normally greater than m.
- The values returned by a hash function are called hash values, hash codes, hash sums, checksums or simply hashes.
- Hash function creates hash value in such a way that it is extremely unlikely that some other text will produce the same hash value.
- A hash table (also called hash map) is used to implement an associative array that can map keys to values. A hash table uses a hash function to compute an index into an array of buckets or slots, from which the correct value can be found.



Cryptographic Hash Function

- A cryptographic hash function is a hash function that takes an arbitrary block of data as input and returns a fixed-size bit string as output. The returned value is called the cryptographic hash value.
- Cryptographic hash function creates hash value in such a way that any (accidental or intentional) change to the data will change the hash value. Therefore, it is extremely unlikely that some other text will produce the same hash value.
- The data to be encoded are often called the message, and the hash value is sometimes called the message digest or simply digest.

Cryptographic Hash Function

- In cryptographic hash function, even a small changes in the input would cause a large change in the output.
- Figure below shows how the slight changes input (here in the word "over") drastically change the resulting output.

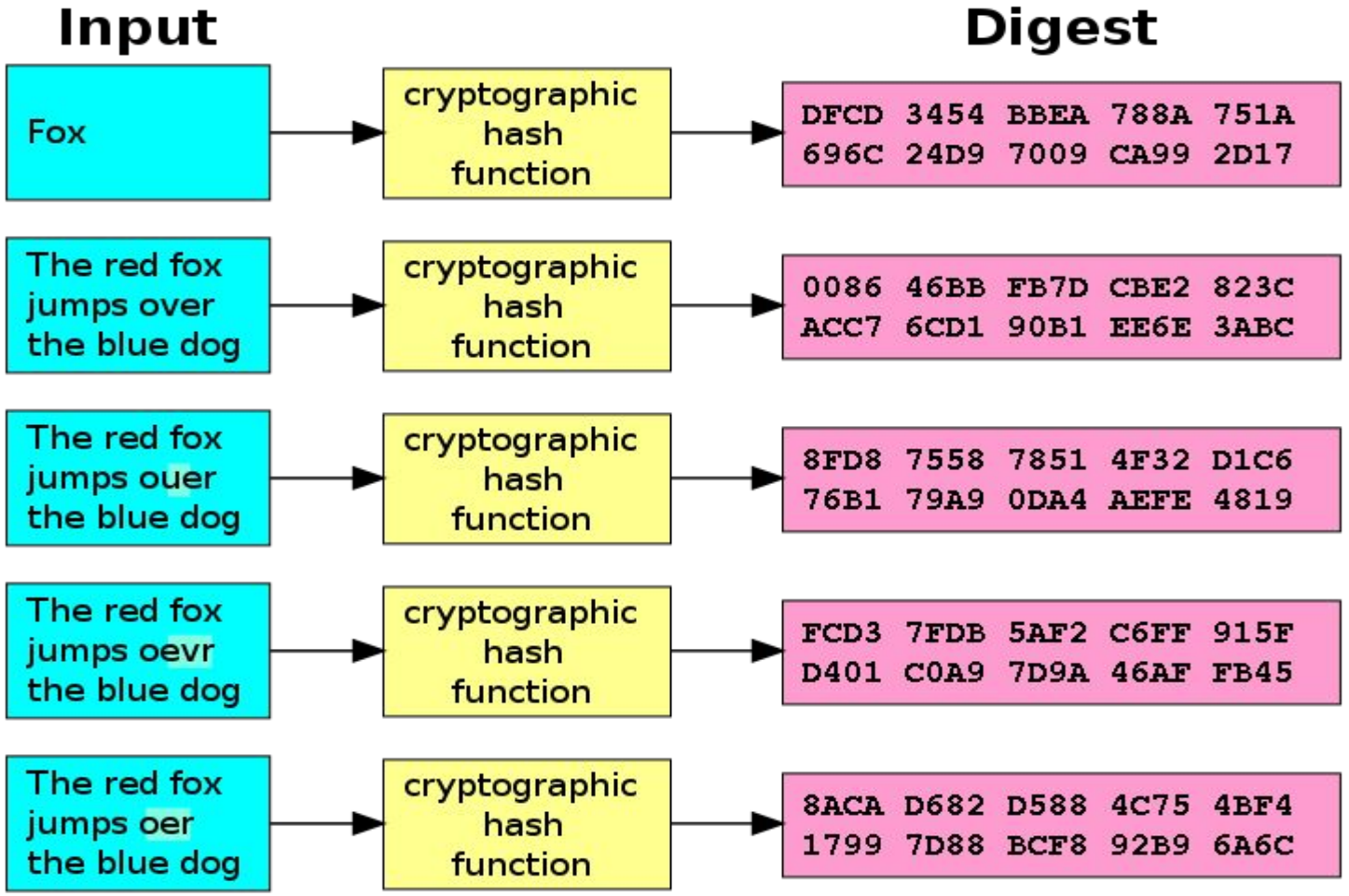


Illustration: Cryptographic Hash Function

- An illustration of the potential use of a cryptographic hash is as follows:
  - Alice poses a tough math problem to Bob and claims she has solved it.
  - Bob would like to try it himself, but would yet like to be sure that Alice is not bluffing. Therefore, Alice writes down her solution, computes its hash and tells Bob the hash value (whilst keeping the solution secret). Then, when Bob comes up with the solution himself a few days later, Alice can prove that she had the solution earlier by revealing it and having Bob hash it and check that it matches the hash value given to him before.

Use of Hash Function

- Cryptographic hash functions have many information security applications, such as in-
  - digital signatures
  - message authentication codes (MACs)
  - other forms of authentication
- Hash functions are primarily used to generate fixed-length output data that acts as a shortened reference to the original data. This is useful when the output data is too cumbersome to use in its entirety.
  - For example, consider a list of person's names. Here, name of each person is of variable length. Searching for a person's name in the list is slow; time required to retrieve each name may also vary. But if each name could be hashed to a fixed length integer, then searching and retrieving each name will be performed in faster with constant time.
- Hash functions are also used to accelerate table lookup or data comparison tasks such as finding items in a database, detecting duplicated or similar records in a large file, finding similar stretches in DNA sequences, and so on.

Hash Functions Used in Cryptography

- The two commonly used hash functions are MD5 and SHA-1.
  - MD5:
    - MD stands for Message Digest.
    - Several MD hash algorithms designed by Ron Rivest are MD2, MD4 and MD5.
    - The last version MD5 is more secured than the previous versions.
    - It divides the message into blocks of 512 bits and creates a 128-bit digest.
  - SHA-1:
    - SHA stands for Secure Hash Algorithm.
    - This standard was developed by NIST (National Institute of Standards and Technology).
    - This standard is mostly based on MD5.
    - Several versions of SHA standard were realsed: SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512.
    - SHA-1 returns a string of 160 bits.
    - Both MD5 and SHA-1 hash functions are built with the Merkle-Damgard construction.

Application of Hash Function in Cryptography

Hash functions are used for:

- Verifying the integrity of message and file
- Verifying password for secure login
- fingerprints of keys
- authentication
- digital signatures

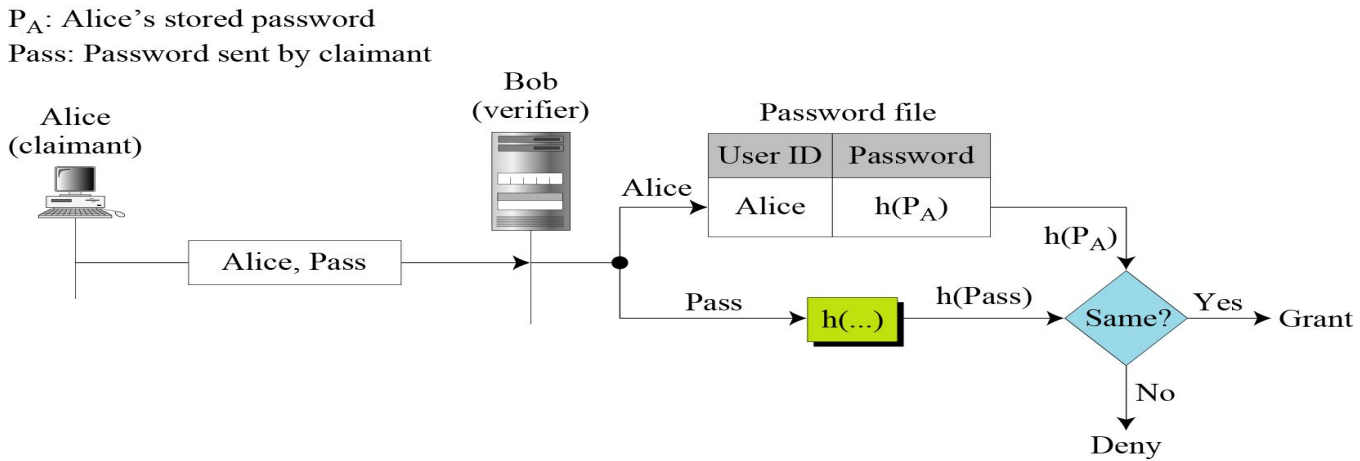
Verifying the integrity of files or messages:

- An important application of secure hashes is verification of message integrity. Determining whether any changes have been made to a message (or a file), for example, can be accomplished by comparing message digests calculated before, and after, transmission (or any other event).
- For this reason, most digital signature algorithms only confirm the authenticity of a hashed digest of the message to be "signed". Verifying the authenticity of a hashed digest of the message is considered proof that the message itself is authentic.

Application of Hash Function in Cryptography

Verifying password for secure login:

- A related application of hash function is password verification.
- Storing all user passwords as plaintext character can result in a massive security breach if the password file is compromised.
- One way to reduce this danger is to only store the hash digest of each password instead of the plaintext password in the table (a file) that is stored by user identification.
- Any user can read the contents of the file, but, because the hash function is a one-way function, it is almost impossible to guess the value of the password.
- When the password is created , the system hashes it and stores the hash in the password file.
- When the user sends her user ID and password, the system creates a hash of the password and then compare the hash value with the one stored in the file.
- If there is a match, the user is granted access; otherwise, access is denied.



Application of Hash Function in Cryptography

File or data identifier:

- A message digest can also serve as a means of reliably identifying a file;
- One of the main applications of a hash function is to allow the fast look-up of a data in a hash table. Being hash functions of a particular kind, cryptographic hash functions lend themselves well to this application too.

Authentication:

- Authentication is the assurance that the communicating entity is the one that it claims to be.
- Cryptographic hash function can be used for provide authentication.

Application of Hash Function in Cryptography

Digital Signature:

- Digital signature, first proposed in 1976 by Whitfield Diffie of Stanford University, is a digital code (encrypted message digest) that can be attached to an electronically transmitted message that uniquely identifies the sender.
- Like a written signature, the purpose of a digital signature is to guarantee that the individual sending the message really is who he or she claims to be. It is linked to the data in such a manner that if the data is changed, the digital signature is invalidated.
- When making a digital signature, cryptographic hash functions are generally used to construct the message digest.
- A digital signature servers three important purposes:
  - Verifies data integrity.
  - Provides authentication of the sender.
  - Provides non-repudiation

Properties of Cryptographic Hash Function

- **Deterministic**: A hash function must consistently produce the same output for the same input.
- **Fixed Output Size**: The output of a hash function should have a fixed size, regardless of the size of the input.
- **Efficiency**: The hash function should be able to process input quickly.
- **Uniformity**: The hash function should distribute the hash values uniformly across the output space to avoid clustering.
- **Pre-image Resistance**: It should be computationally infeasible to reverse the hash function, i.e., to find the original input given a hash value.
- **Collision Resistance**: It should be difficult to find two different inputs that produce the same hash value.
- **Avalanche Effect**: A small change in the input should produce a significantly different hash value.

Simple Hash Function

Some Popular Hash Function:

Here are some relatively simple hash functions that have been used:

- Division-remainder method
- Mid-square method
- Folding method

Division-remainder method:

- Using this method, choose a number m that is larger than the number n of keys in K (K is a set of keys). Generally, the number m is chosen to be a prime number. The hash function H is defined as:

  $H(k)= k \pmod m$ or, $H(k)= k \pmod m+1$

- Here k (mod m) denotes the remainder when k is divided by m.
- The second formula is used when we want the hash addresses to range from 1 to m rather than from 0 to m-1.

Simple Hash Function

Example: Division-remainder method:

Suppose a company with 68 employees assigned a 4-digit employee number to each employee which is used as the primary key. Apply the division method of hash function to each of the following employee number:
3205, 7148, 2345

Solution:

- Since, there are 68 employees in the company, two digit employee number is sufficient to represent them.
- Highest 2 digit number is 99 and 97 is the nearest 2 digit prime number of 99. So, we divide each of the 4 digit employee number by 97.

  H(3205)= 3205 (mod 97)= 04.

  H(7148)= 7148 (mod 97)= 67

  H(2345)= 2345 (mod 97)= 17
- In the case that the memory addresses begin with 01 rather than 00, we choose that the function H(k) = k(mod m)+1 to obtain. H(3205)= 3205 (mod 97)+1= 4+1=05

Simple Hash Function (Mid-Square Hashing)

- Mid-Square hashing is a hashing technique in which unique keys are generated.
- In this technique, a seed value is taken and it is squared.
- Then, some digits from the middle are extracted.
- These extracted digits form a number which is taken as the new seed.
- This technique can generate keys with high randomness if a big enough seed value is taken.
- However, it has a limitation.
- As the seed is squared, if a 6-digit number is taken, then the square will have 12-digits.
- This exceeds the range of int data type. So, overflow must be taken care of.

Simple Hash Function

Example: Mid-square method

Suppose a company with 68 employees assigned a 4-digit employee number to each employee which is used as the primary key. Apply the mid-square method of hash function to each of the following employee number:
3205, 7148, 2345

Solution:

K	3205	7148	2345
K^2	10272025	51093904	5499025
H(k) = I	72	93	99

- Observe that the 4th and 5th digits counting from right are chosen for the hash address.

Simple Hash Function

Folding method:

Folding Method in Hashing: It breaks up a key value into precise segments that are added to form a hash value

Algorithm:

- The folding method is used for creating hash functions starts with the item being divided into equal-sized pieces i.e., the last piece may not be of equal size.
- The outcome of adding these bits together is the hash value, **H(x) = (a + b + c) mod M**, where a, b, and c represent the preconditioned key broken down into three parts and **M** is the table size, and **mod** stands for modulo.
- In other words, the sum of three parts of the preconditioned key is divided by the table size. The remainder is the hash key.

- **Example 1:** The task is to fold the key **123456789** into a Hash Table of ten spaces (0 through 9).
  - It is given that the key, say **X** is 123456789 and the table size (i.e., **M** = 10).
  - Since it can break **X** into three parts in any order. Let's divide it evenly.
  - Therefore, a = 123, b = 456, c = 789.
  - Now, **H(x) = (a + b + c) mod M** i.e., H(123456789) =(123 + 456 + 789) mod 10 = 1368 mod 10 = **8**.
  - Hence, 123456789 is inserted into the table at address **8**.

SHA-512

- SHA-512, or Secure Hash Algorithm 512, is a hashing algorithm used to convert text of any length into a fixed-size string. Each output produces a SHA-512 length of 512 bits (64 bytes).
- This algorithm is commonly used for email addresses hashing, password hashing, and digital record verification. SHA-512 is also used in blockchain technology, with the most notable example being the BitShares network.

Key Features of SHA-512

- **Robustness and Resistance to various cryptographic attacks —** SHA-512 produces a fixed-size 512-bit hash value, providing a vast number of possible output combinations, which enhances its resistance to collision attacks.
- **Logical and bitwise operations —** Solutions such as modular addition and bitwise rotation ensure the unpredictability and irreversibility of the hash function. The algorithm is designed to handle input messages of up to 2^128 - 1 bits in length, making it suitable for a wide range of data sizes.
- **Iterative Structure:** SHA-512 employs an iterative process with multiple rounds of processing, each involving a set of specific mathematical functions. This iterative structure enhances the diffusion and avalanche effects, making it computationally infeasible to predict the hash output from small changes in the input.
- **Versatility:** While SHA-512 is commonly used for cryptographic purposes, it also finds applications in other fields, such as checksum verification and data integrity checks. Its versatility makes it a valuable tool for a wide range of information security and data management scenarios.
- **Resistance to Birthday Attacks:** The 512-bit output length significantly increases resistance to birthday attacks, a type of cryptographic attack that exploits the probability of two different inputs producing the same hash value. The large output space reduces the likelihood of such collisions, bolstering the security of the hash function.

SHA-512

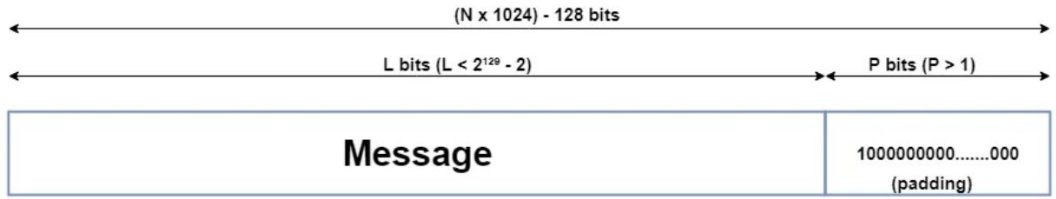
- SHA-512 does its work in a few stages. These stages go as follows:

1. Input formatting
2. Hash buffer initialization
3. Message Processing
4. Output

1. Input Formatting

- SHA-512 can’t actually hash a message input of any size, i.e. it has an input size limit.
- This limit is imposed by its very structure as you may see further on.
- The entire formatted message has basically three parts: the original message, padding bits, size of original message.
- And this should all have a combined size of a whole multiple of 1024 bits.
- This is because the formatted message will be processed as blocks of 1024 bits each, so each bock should have 1024 bits to work with.

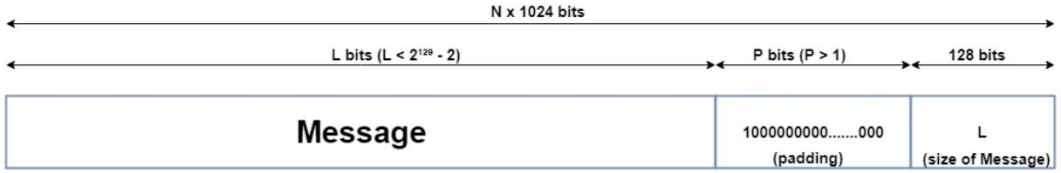
- The input message is taken and some padding bits are appended to it in order to get it to the desired length.
- The bits that are used for padding are simply ‘0’ bits with a leading ‘1’ (100000…000).
- Also, according to the algorithm, padding *needs* to be done, even if it is by one bit.
- So a single padding bit would only be a ‘1’.
- The total size should be equal to 128 bits short of a multiple of 1024 since the goal is to have the formatted message size as a multiple of 1024 bits (N x 1024).



Message with padding

Padding Size

- After this, the size of the original message given to the algorithm is appended.
- This size value needs to be represented in 128 bits and is the only reason that the SHA-512 has a limitation for its input message.
- Since the size of the original message needs to be represented in 128 bits and the largest number that can be represented using 128 bits is (2^{128}-1), the message size can be at most (2^{128}-1) bits; and also taking into consideration the necessary single padding bit, the maximum size for the original message would then be (2^{128}-2).
- Even though this limit exists, it doesn't actually cause a problem since the actual limit is so high (2^{128}-2 = 340,282,366,920,938,463,463,374,607,431,768,211,454 bits).
- This is the ultimate formatted input for SHA-512



Message with padding and size

Formatted Message

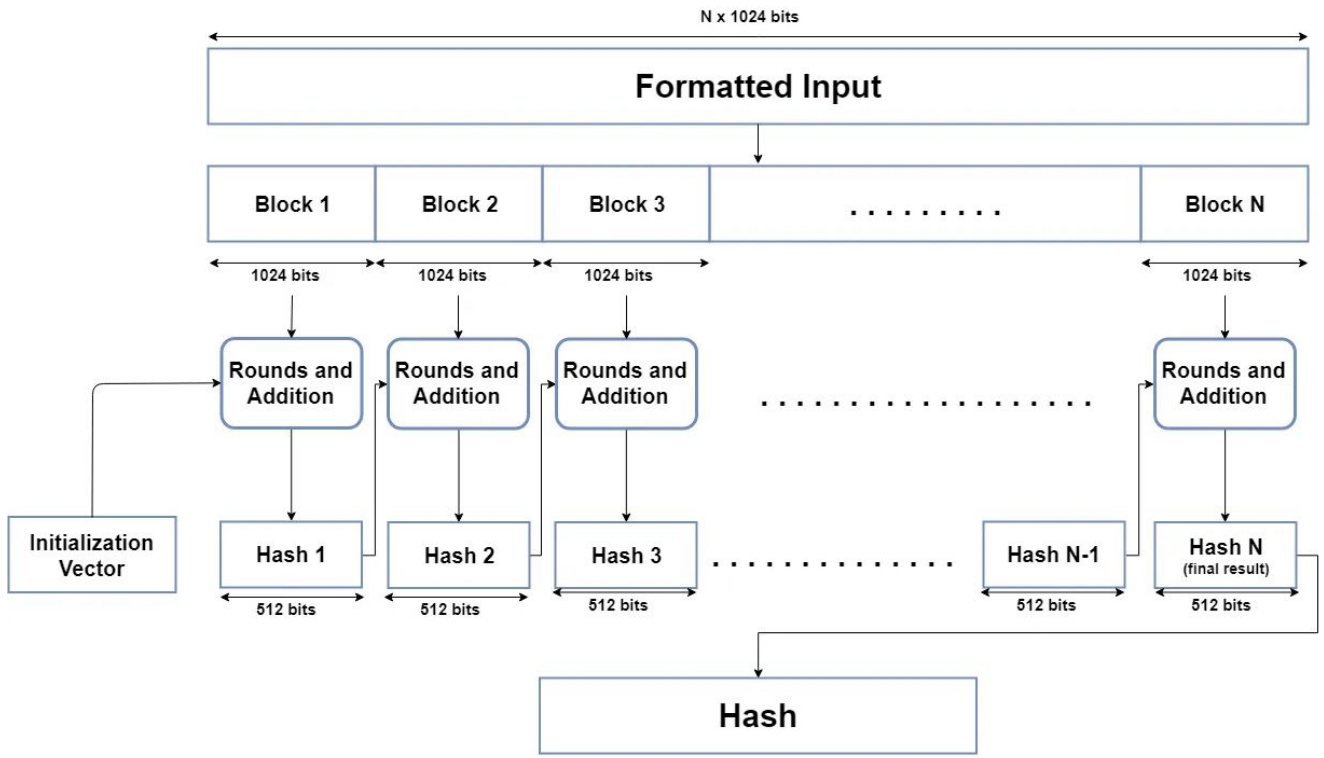
- The algorithm works in a way where it processes each block of 1024 bits from the message using the result from the previous block.
- Now, this poses a problem for the first 1024 bit block which can't use the result from any previous processing. This problem can be solved by using a default value to be used for the first block in order to start off the process. (Have a look at the second-last diagram).
- Since each intermediate result needs to be used in processing the next block, it needs to be stored somewhere for later use
- This would be done by the *hash buffer*, this would also then hold the final hash digest of the entire processing phase of SHA-512 as the last of these 'intermediate' results.
- So, the default values used for starting off the chain processing of each 1024 bit block are also stored into the hash buffer at the start of processing.
- The actual value used is of little consequence, but for those interested, the values used are obtained by taking the first 64 bits of the fractional parts of the square roots of the first 8 prime numbers (2,3,5,7,11,13,17,19). These values are called the Initial Vectors (IV).
- Why 8 prime numbers instead of 9? Because the hash buffer actually consists of 8 subparts (registers) for storing them.

Initialization Vector

a = 0x6A09E667F3BCC908	b = 0xBB67AE8584CAA73B
c = 0x3C6EF372FE94F82B	d = 0xA54FF53A5F1D36F1
e = 0x510E527FADE682D1	f = 0x9B05688C2B3E6C1F
g = 0x1F83D9ABFB41BD6B	h = 0x5BE0CD19137E2179

Hash buffer and Initialization Vector values

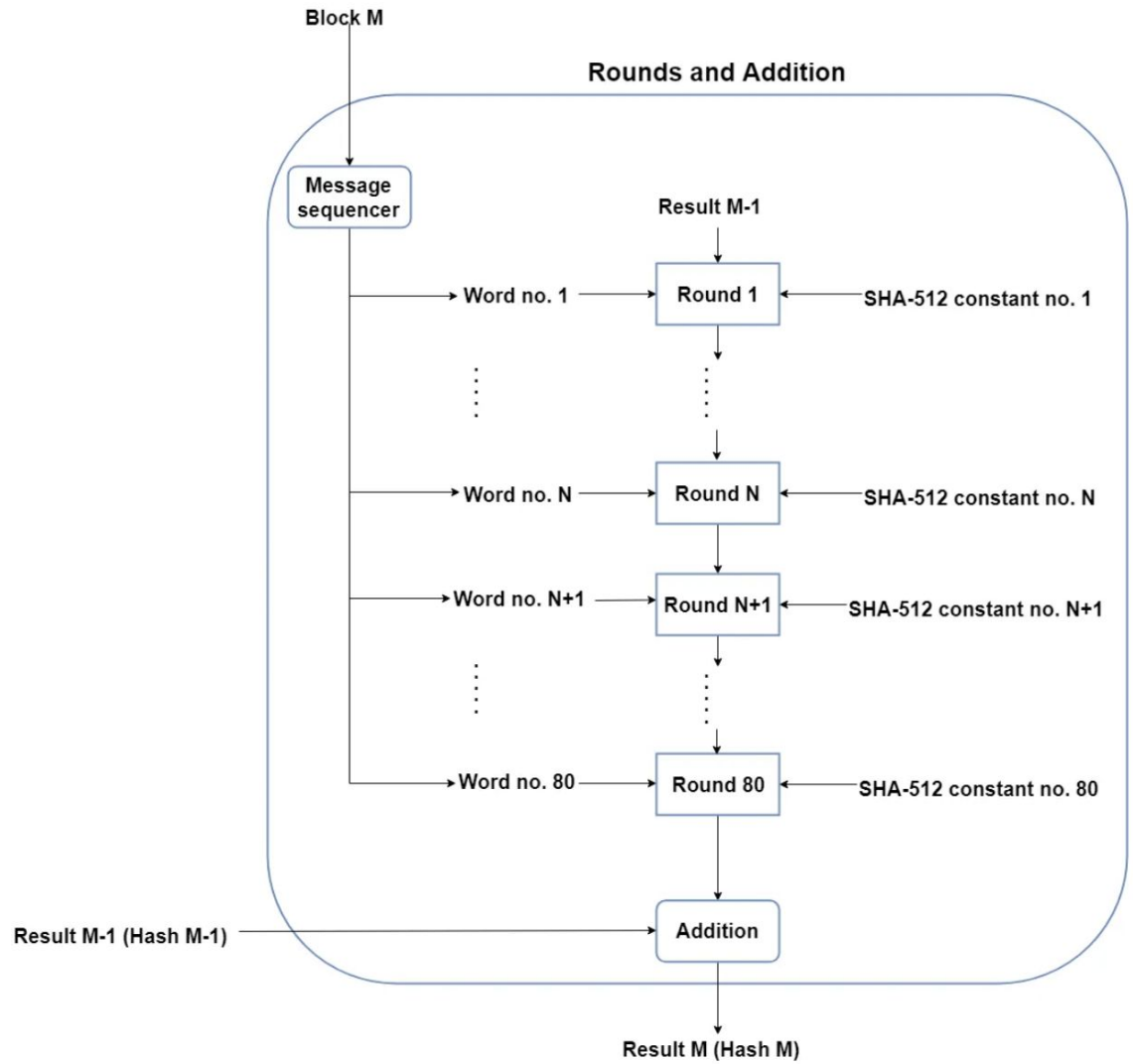
- Message processing is done upon the formatted input by taking one block of 1024 bits at a time.
- The actual processing takes place by using two things: The 1024 bit block, and the result from the previous processing.
- This part of the SHA-512 algorithm consists of several ‘Rounds’ and an addition operation.
- So, the Message block (1024 bit) is expanded out into ‘Words’ using a ‘message sequencer’.
- Eighty Words to be precise, each of them having a size of 64 bits.



William Stallings, Cryptography and Network Security — Principles and Practise (Seventh Edition) referred for diagram

Rounds

- The main part of the message processing phase may be considered to be the Rounds.
- Each round takes 3 things: one Word, the output of the previous Round, and a SHA-512 constant.
- The first Round doesn't have a previous Round whose output it can use, so it uses the final output from the previous message processing phase for the previous block of 1024 bits.
- For the first Round of the first block (1024 bits) of the formatted input, the Initial Vector (IV) is used.
- SHA-512 constants are predetermined values, each of whom is used for each Round in the message processing phase.
- Again, these aren't very important, but for those interested, they are the first 64 bits from the fractional part of the cube roots of the first 80 prime numbers.Why 80? Because there are 80 Rounds and each of them needs one of these constants.
- Once the Round function takes these 3 things, it processes them and gives an output of 512 bits.
- This is repeated for 80 Rounds. After the 80th Round, its output is simply added to the result of the previous message processing phase to get the final result for this iteration of message processing.


Block M
Rounds and Addition
Message sequencer
Result M-1
Word no. 1
Round 1
SHA-512 constant no. 1
Word no. N
Round N
SHA-512 constant no. N
Word no. N+1
Round N+1
SHA-512 constant no. N+1
Word no. 80
Round 80
SHA-512 constant no. 80
Result M-1 (Hash M-1)
Addition
Result M (Hash M)

4. Output

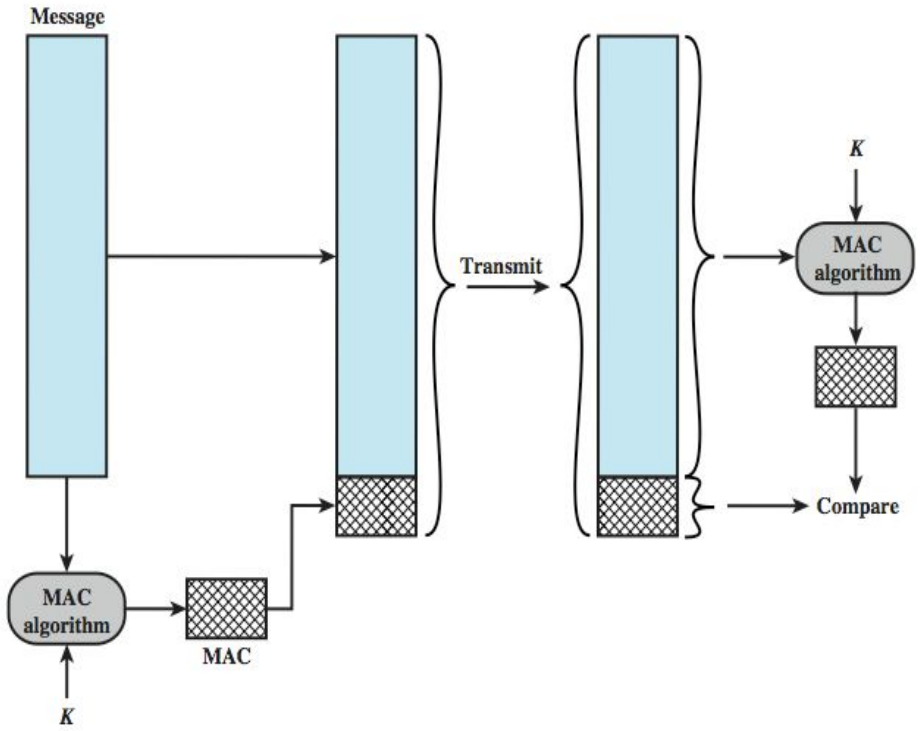
- After every block of 1024 bits goes through the message processing phase, i.e. the last iteration of the phase, we get the final 512 bit Hash value of our original message.
- So, the intermediate results are all used from each block for processing the next block.
- And when the final 1024 bit block has finished being processed, we have with us the final result of the SHA-512 algorithm for our original message.
- Thus, we obtain the final hash value from our original message.

Application of SHA-512

- Digital Signature Verification:
  - Digital signatures follow asymmetric encryption methodology to verify the authenticity of a document/file. Hash algorithms like SHA 256 go a long way in ensuring the verification of the signature.
- Password Hashing:
  - As discussed above, websites store user passwords in a hashed format for two benefits. It helps foster a sense of privacy, and it lessens the load on the central database since all the digests are of similar size.
- SSL Handshake:
  - The SSL handshake is a crucial segment of the web browsing sessions, and it's done using SHA functions. It consists of your web browsers and the web servers agreeing on encryption keys and hashing authentication to prepare a secure connection.
- Integrity Checks:
  - As discussed above, verifying file integrity has been using variants like SHA 256 algorithm and the MD5 algorithm. It helps maintain the full value functionality of files and makes sure they were not altered in transit.

Message Authentication Code (MAC) Vs. Hash Code

- MAC is a technique for message authentication which involves the use of a secret key to generate from a small block of data, known as a message authentication code, that is appended to the message.
- This technique assumes that two communicating parties, say Alice and Bob, share a common secret key K_{AB}. When Alice has a message to send to Bob, she calculates the message authentication code as a complex function of the message and the key: $MAC_M = F(K_{AB}, M)$.
  - The message plus code are transmitted to the intended recipient.
  - The recipient performs the same calculation on the received message, using the same secret key, to generate a new message authentication code.
  - The received code is compared to the calculated code.



Message Authentication Code (MAC) Vs. Hash Code

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

1. The receiver is assured that the message has not been altered.
   - If an attacker alters the message but does not alter the code, then the receiver's calculation of the code will differ from the received code.
   - Because the attacker is assumed not to know the secret key, the attacker cannot alter the code to correspond to the alterations in the message.

2. The receiver is assured that the message is from the alleged sender.
   - Because no one else knows the secret key, no one else could prepare a message with a proper code.

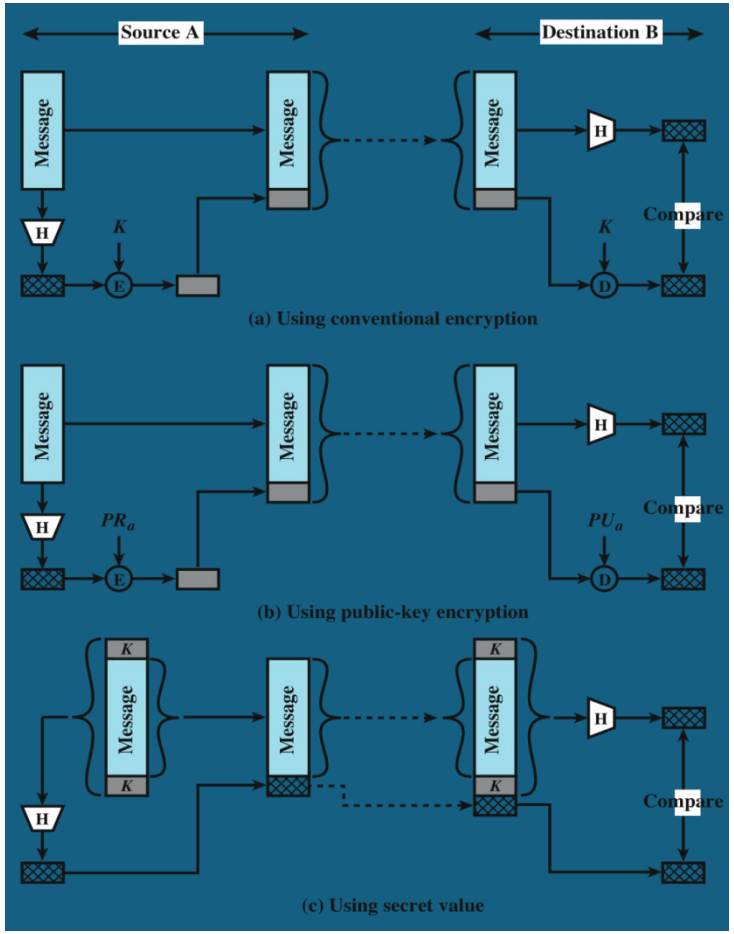
3. If the message includes a sequence number (such as is used with X.25, HDLC, and TCP), then the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

Message Authentication Code (MAC) Vs. Hash Code

- An alternative to the message authentication code is the one-way hash function.
- A hash function accepts a variable-size message M as input and produces a fixed-size message digest H(M) as output. The purpose of a hash function is to produce a "fingerprint" of a file, message, or other block of data.
- Typically, the message is padded out to an integer multiple of some fixed length (e.g., 1024 bits) and the padding includes the value of the length of the original message in bits.
- Unlike the MAC, a hash function does not take a secret key as input. To authenticate a message, the message digest is sent with the message in such a way that the message digest is authentic.

Message Authentication Code (MAC) Vs. Hash Code

- Figure below illustrates three ways in which the message can be authenticated using a hash code.



  - The message digest can be encrypted using symmetric key encryption (part a); if it is assumed that only the sender and receiver share the encryption key, then authenticity is assured.
  - The message digest can also be encrypted using public-key encryption (part b);
  - Part c illustrate a technique, known as a keyed hash MAC where authentication is done without using encryption. It assumes that two communicating parties, say A and B, share a common secret key K which is incorporated into the process of generating a hash code.
  - When A has a message to send to B, it calculates the hash function over the concatenation of the secret key and the message: MDM = H(KMK). It then sends [MMDM] to B. Because B possesses K, it can recompute H(KMK) and verify MDM. Because the secret key itself is not sent, it should not be possible for an attacker to modify an intercepted message. As long as the secret key remains secret, it should not be possible for an attacker to generate a false message.

Discussion Points

- Digital Signature
  - To be familiar with the general idea behind digital signature
  - To define security services provided by a digital signature
  - To describe some applications of digital signatures
- Cryptographic Hash Function
  - To introduce general ideas behind hash function
  - To discuss the usage and application of hash function
  - To know the desirable properties of a hash function

Identifying Basic Authentication & Authorization Concepts

Dr. Risala Tasin Khan

Overview

Strong authentication is the first line of defense in the battle to secure network resources. But authentication is not a single process; there are many different methods and mechanisms, some of which can even be combined to form more complex schemes. As a network professional, familiarizing yourself with basic authentication and authorization concepts can help you select, implement, and support the ones that are appropriate for your environment.

Authentication

- It is the process of ascertaining that somebody really is who he (or she) claims to be.
- Authentication = login + password (who you are).

Authorization

- It refers to rules that determine who is allowed to do what; that is, what level of access a particular authenticated user should have to secured resources controlled by the system.
- For example, Asif may be authorized to create and delete databases, while Rasel is only authorized to read.
  - Another example: a database management system might be designed so as to provide certain specified individuals with the ability to retrieve information from a database but not the ability to change data stored in the database, while giving other individuals the ability to change data.
  - Authorization systems provide answers to the questions:
    - Is user X authorized to access resource R?
    - Is user X authorized to perform operation P?
    - Is user X authorized to perform operation P on resource R?
  - Authorization = permissions (what you are allowed to do).

Some Common Terms

Password

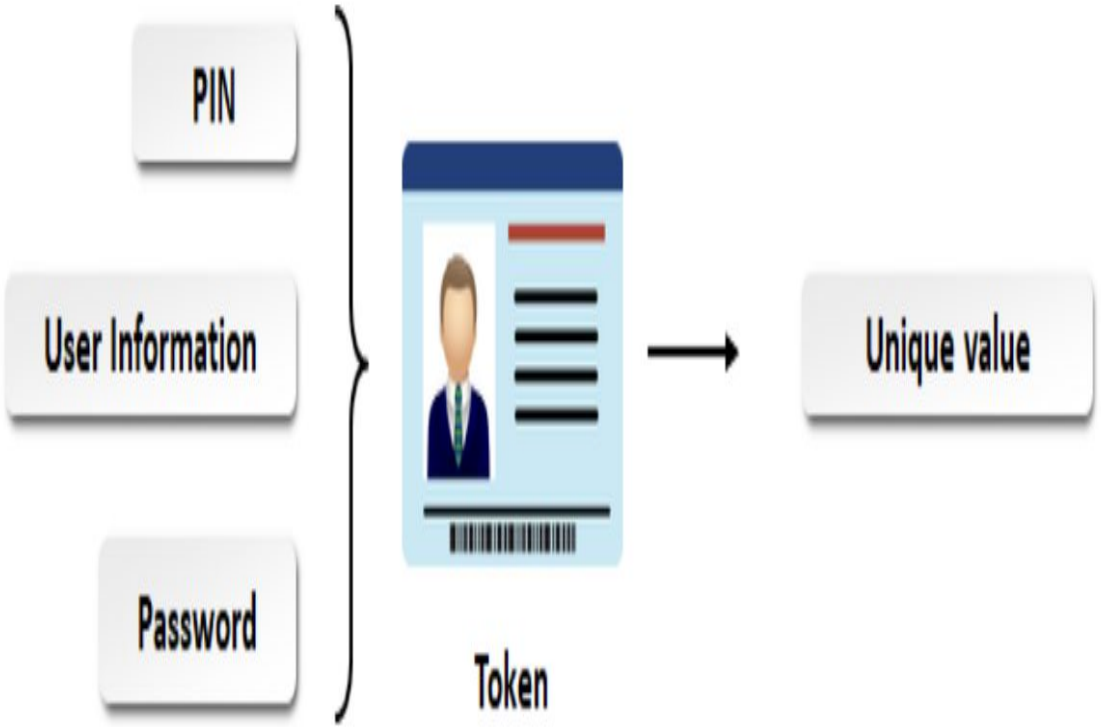
- The combination of user name and password is one of the most basic and widely used authentication scheme.
- In this type of authentication user’s credential is checked against credential stored in a database.
- If the user’s name and password match with the database the user is authenticated.
- If not, the user is denied access.
- This method may not be very secure as it doesn’t necessarily identify the correct user.

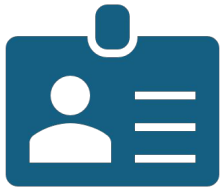
Token

Tokens are physical or virtual objects such as smart cards, ID badges, or data packets that store authentication information.

Tokens can store personal identification number(PIN), information about users or passwords.

Unique token value can be generated using special devices or software in response to a challenge from an authentication server or by using independent algorithm.





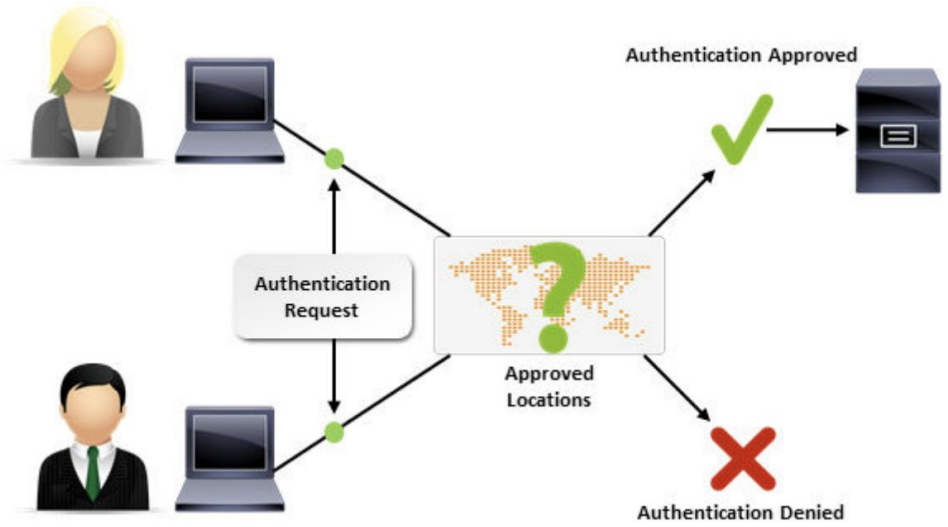
Smart Card

- Smart cards are a common example of token based authentication.
- A smart card is a plastic card containing an embedded computer chip that can store different types of electronic information.
- The content of a smart card can be read with a smart card reader.

Biomatrices

- Biomatrices are authentication scheme based on the identification of individuals by their physical characteristics.
- This can involve a fingerprint scanner, a retinal scanner, a hand geometry scanner, and voice recognition and facial recognition software.

Geolocation



- With more and more mobile devices connecting to networks, geolocation provides an extra level of authentication.
- Users who are attempting to authenticate from an approved location can be granted network access.
- Internet and computer geolocation can be performed by associating a geographic location with an internet protocol (IP) address, RFID, embedded hardware or software number, Wi-Fi position system, GPS coordinates, or other information.
- Geolocation usually works by looking us a host's IP address in a geolocation database and retrieving the registrant's country, region, city name and other information.
- When a physical location is identified that can be compared to a list of locations that are approved for (or restricted for) network access and approve to the resources can be granted accordingly.
- Conversely, if a network attack originates from a particular country packets originating from IP addresses physically located to that country could be automatically dropped during that attack period.

TYPES OF AUTHENTICATION

Data Origin or Message Authentication

Data-origin or Message authentication:

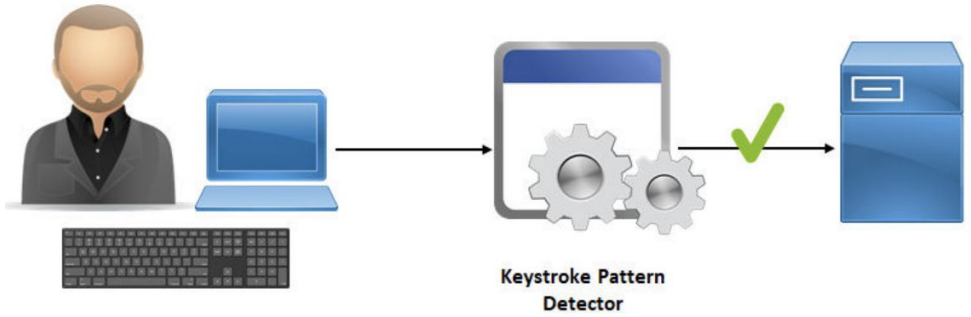
- Message or data authentication is a procedure that allows communicating parties to verify that received or stored messages are authentic.
- A message, file, document, or other collection of data is said to be authentic when it is genuine and came from its alleged source.
- The two important aspects of message authentication are to verify that the contents of the message have not been altered and that the source is authentic.

Entity Authentication

- Entity authentication is a technique designed to let one party prove the identity of another party. For example, a student who needs to access her university resources needs to be authenticated during the logging process.
- 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.

Keystroke Authentication

- Keystroke authentication is a type of authentication that relies on detailed information that describes exactly when a keyboard key is pressed and released as someone types information into a computer or other electronic device.
- Each user has certain techniques, rhythms and patterns when it comes to typing on a keyboard and these can be recorded and measured to compare against future keystrokes.
- Keystroke authentication requires the use of a keystroke logger and other measurements such as when a key is pressed and released, the interval between a key release and the next key being pressed and so on.
- Some consider keystroke authentication as an extension of biomatrics.



Multi-factor Authentication

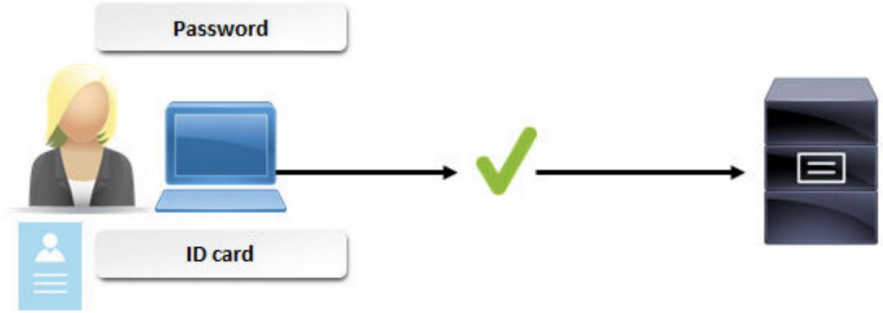


Figure 1-18: Multi-factor authentication.

- Multi-factor authentication is any authentication scheme that requires validation of two or more authentication factors.
- It can be any combination of who you are, what you have, what you know, where you are and what you do.
- Requiring a physical ID card along with a secret password is an example of multi factor authentication.
- Another example a user requires a validation code from his mobile txt message for entering into his email after giving password.
- Multi-factor authentication requires the factors to be different, not just the specific objects or methods.

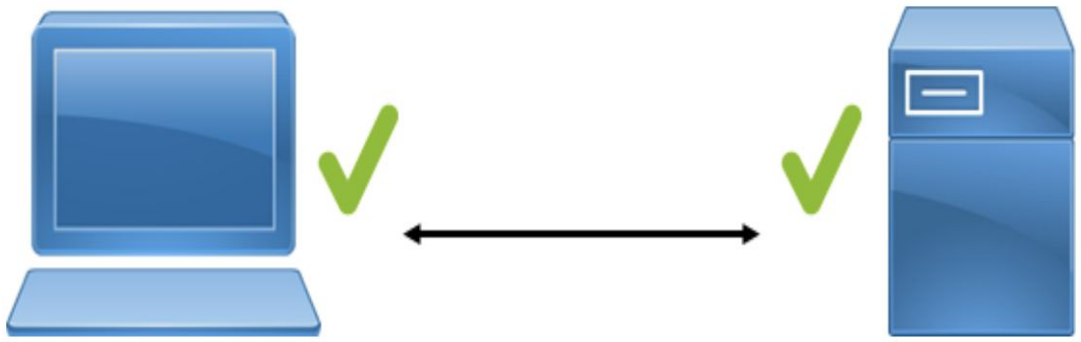
Mutual Authentication

Mutual authentication is a security mechanism that requires that each party in a communication verifies each other's identity.

A service or a resource verifies the client's credential and the client verifies the resource's credential.

Mutual authentication prevents a client from inadvertently submitting confidential information to a non-secure server.

Mutual authentication helps in avoiding man-in-the-middle attack and session hijacking attack.



Message Authentication

- A message can be authenticated either by-

1) Message Authentication Code (MAC)
2) Hash Function

Message Authentication using MAC

- **MAC** is a technique for message authentication.
  - A message authentication code (MAC) is a short piece of information that is appended to the message to be sent for authentication.
  - A MAC algorithm accepts a **secret key** and a **message** to be authenticated.
    - It outputs a MAC, which is sometimes called a **tag**.
- The MAC value protects both a message's integrity as well as its authenticity by allowing verifiers (who also possess the same secret key) to detect any changes to the message content.

Message Authentication using MAC

- This technique assumes that two communicating parties, say Alice and Bob, share a common secret key K_{AB}. When Alice has a message to send to Bob, she calculates the message authentication code as a complex function of the message and the key: $MAC_M = F(K_{AB}, M)$.
- The process of creating MAC value and verifying the integrity of message is outlined below:

Sender Site

1. Sender generates a message.
2. He/she then creates a "MAC or Tag" of the message using MAC algorithm (Note that the MAC algorithm accepts two input- a shared secret key between the communicating parties and the message that is to be sent).
3. Sender attaches the MAC code to the end of the message that is to be sent.
4. The sender sends the attached message to the recipient.
5. The sender encrypts the attached message with the recipient's public key and sends it to the recipient.

Receiver Site

6. After receiving, the recipient decrypts the entire message with his/her private key.
7. The recipient detaches the message and MAC code.
8. He/she creates a "MAC or Tag" of the received message using the same MAC algorithm the sender used.
9. The recipient then compares the two MACs. If they are equal, the message is granted, otherwise it will be rejected.

Message Authentication using MAC

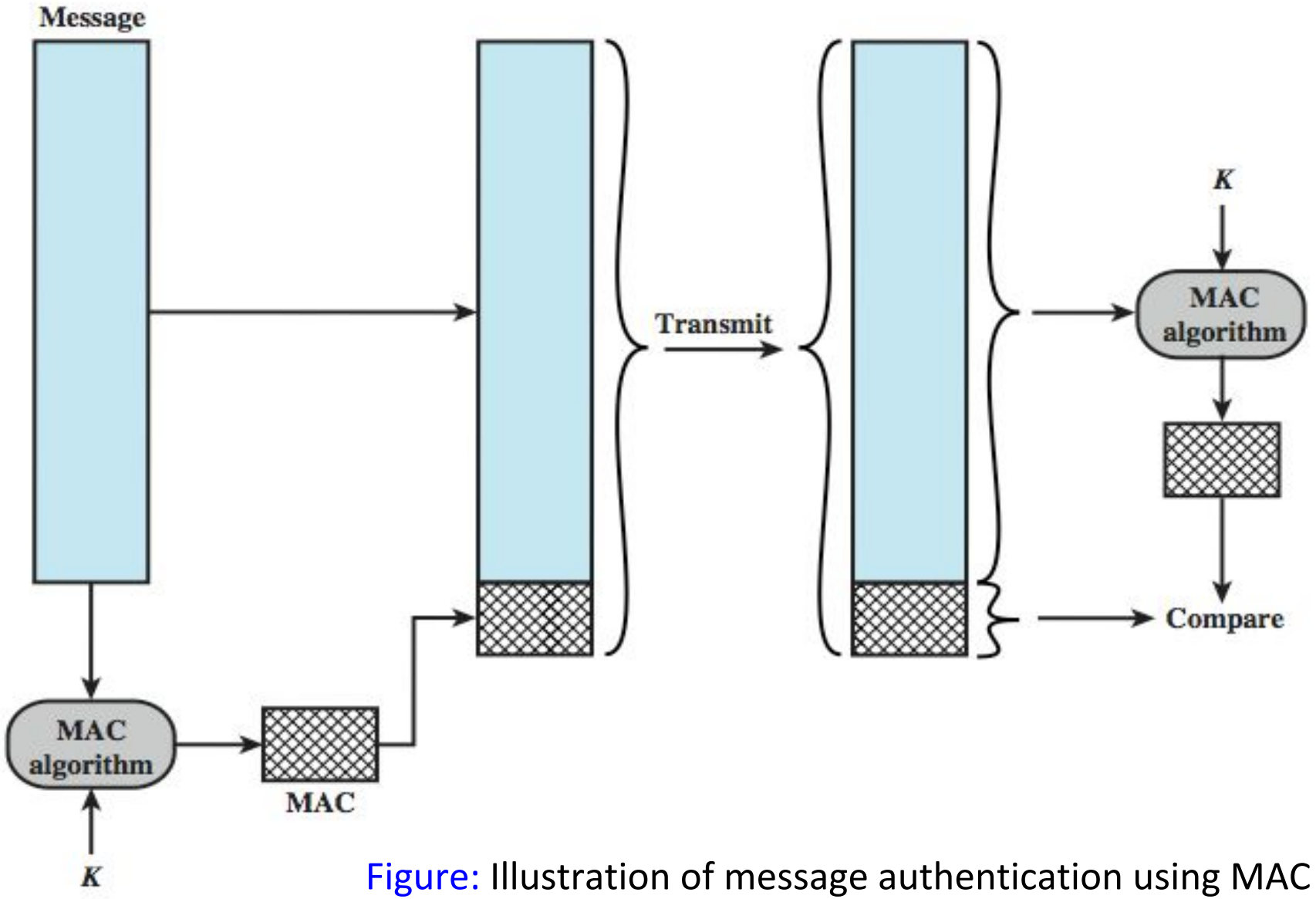


Figure: Illustration of message authentication using MAC

Message Authentication Code (MAC)

- If we assume that only the receiver and the sender know the identity of the secret key, and if the received code matches the calculated code, then-
  1. The receiver is assured that the message has not been altered.
     - If an attacker alters the message but does not alter the code, then the receiver's calculation of the code will differ from the received code.
     - Because the attacker is assumed not to know the secret key, the attacker cannot alter the code to correspond to the alterations in the message.
  1. The receiver is assured that the message is from the alleged sender.
     - Because no one else knows the secret key, no one else could prepare a message with a proper code.
       - If the message includes a sequence number, then the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

Message Authentication using Hash Function

- To authenticate a message, an alternative approach to the **MAC** is the **one-way hash function.**
- A hash function is a formula or an algorithm that-
  - takes large data sets of variable length as input, and
  - returns smaller data sets of fixed length as output.
    - Since, the output is smaller than the input data, a hash function **compresses** an **n-bit message** string to create an **m-bit string** where **n** is normally greater than **m**.
    - The values returned by a hash function are called **hash values, hash codes, hash sums, checksums or message digest (MD).**
- Unlike the MAC (that takes a secret key and message as input and produces a tag as output), a hash function does not take a secret key as input.
  - It only receives message as input and produces message digest as output.
    - The message digest is created in such a way that it is extremely unlikely that some other text will produce the same hash value.

Message Authentication using Hash Function

- The process of creating a digital signature and authenticating a message using hash function is outlined below:

Sender Site

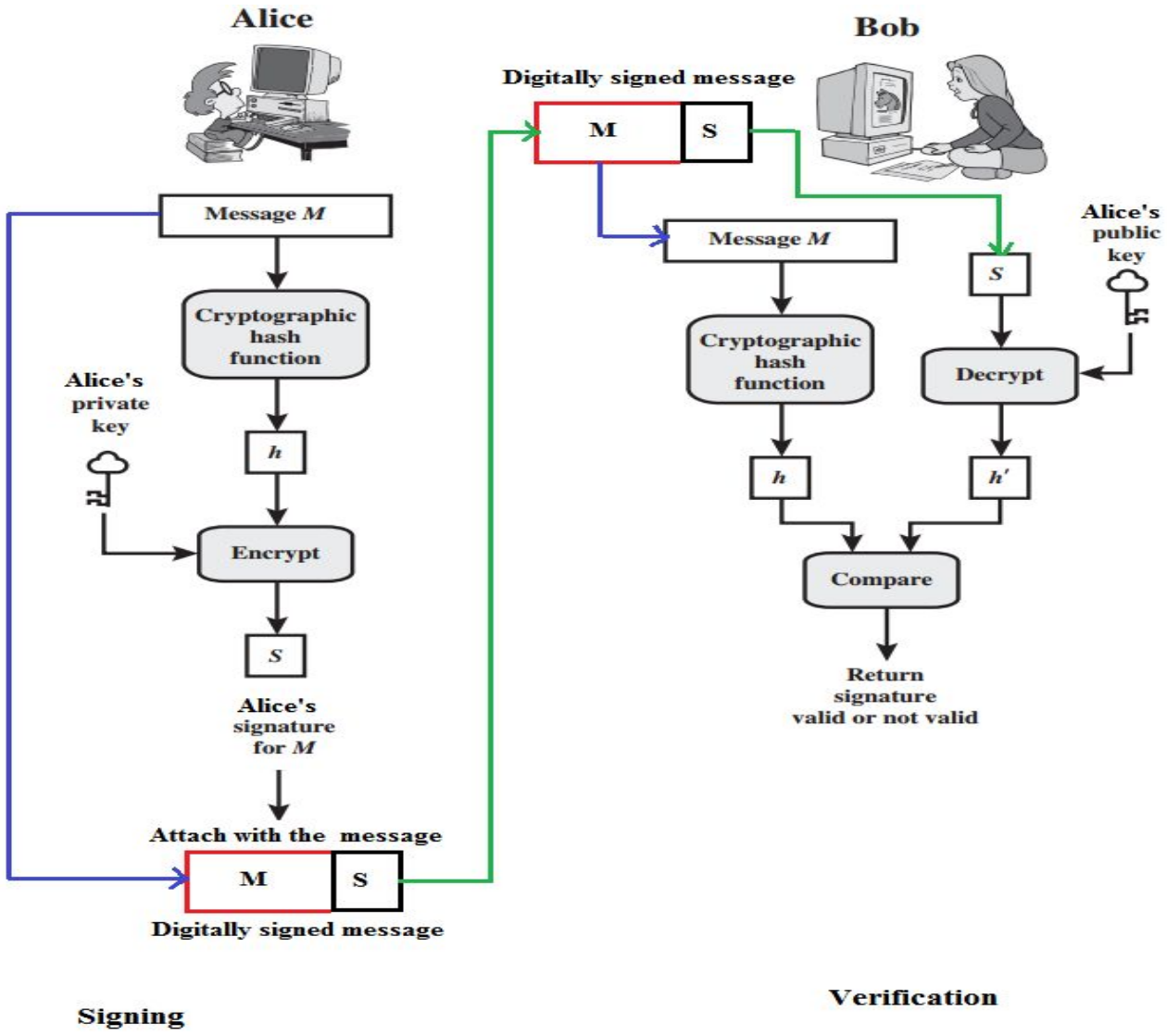
1. Sender generates a message.
2. He/she then creates a "digest" of the message using cryptographic hash function.
3. Sender encrypts the message digest with his/her private key for authentication. This encrypted message digest is called digital signature.
4. Sender attaches the digital signature to the end of the message that is to be sent. The message attached with digital signature is known as digitally signed message.
5. The sender encrypts the digitally signed message with the recipient's public key and sends it to the recipient.

Receiver Site

6. After receiving, the recipient decrypts the entire message with his/her private key.
7. The recipient detaches the message and digital signature.
8. He/she creates a "digest" of the received message using the same hash function the sender used.
9. The recipient decrypts the digital signature and finds the "digest" that the sender created.
10. The recipient then compares the two digests. If they are equal, the message is granted, otherwise it will be rejected.

Message Authentication using Hash Function

The processes are illustrated in the figure below.



Slide-16

Figure: Illustration of message authentication using hash function

Verification Categories/ Authentication Factors:

- In entity authentication, the claimant must identify herself to the verifier.
- This can be done with one of three kinds of witnesses or factors:

❑ Something known/ knowledge factor:

- This is a secret known only by the claimant that can be checked by the verifier.
  - e.g. Password, PIN, secret key, and private key.

❑ Something possessed/ Possession factor:

- This is something that can prove the claimant's identity.
  - e.g. passport, driver's license, ID card, credit card, smart card.

❑ Something inherent/ Inherence factor:

- This is an inherent characteristics of the claimant.
  - e.g. Conventional signature, fingerprints, voice, facial characteristics, retinal pattern, and handwriting

Types of Authentication based on Factors

Single-Factor Authentication (SFA):

- Single-factor authentication is a process for securing access to a given system, such as a network or website, that identifies the party requesting access through only one category of credentials.
  - The most common example of SFA is password-based authentication.

Types of Authentication based on Factors

Two-Factor Authentication (2FA):

- A 2FA system strengthens security by requiring the user to provide dual means of identification from separate authentication categories- one of which is typically a physical token, such as a card, and the other of which is typically something memorized, such as a security code.
  - An automated teller machine (ATM) typically requires two-factor verification.
    - To prove that users are who they claim to be, the system requires two items: an ATM smartcard and the personal identification number (PIN).
    - In the case of a lost ATM card, the user's accounts are still safe; anyone who finds the card cannot withdraw money as they do not know the PIN.
    - The same is true if the attacker has only knowledge of the PIN and does not have the card. or a voiceprint.

Types of Authentication based on Factors

Three-factor Authentication (3FA):

- 3FA adds another factor for further difficulty in falsifying authentication. Typically a biometric trait measurement is added for the inherence factor.
  - Some security procedures now require three-factor authentication, which involves possession of a physical token and a password, used in conjunction with biometric data, such as finger scanning or a voiceprint.

Verification by Something Known: Passwords

- The simplest and oldest method of entity authentication is the password-based authentication, where the password is something that the claimant knows.
- A password is used when a user needs to access a system to use the system's resources (login).
  - User has a secret password
  - System checks password to authenticate user
- Each user of the system has a user identification that is public, and a password that is private.
- **Issues:**
  - How is password stored?
  - How does system check password?
  - How easy is it to guess a password?
  - Difficult to keep password file secret, so best if it is hard to guess password even if you have the password file.

Verification by Something Known: Passwords

Kinds of Password Authentication:

- Password authentication can be divided into two schemes:
  - **1st Scheme: Fixed Password**
    - This password is fixed and used always for every communication.
  - **2nd Scheme: One-time Password**
    - One form of attack on networked computing systems is eavesdropping on network connections to obtain authentication information such as the login IDs and passwords of legitimate users.
    - Once this information is captured, it can be used at a later time to gain access to the system. One-time password systems are designed to counter this type of attack, called a "replay attack".
    - The authentication system uses a secret pass-phrase to generate a sequence of one-time (single use) passwords.

Verification by Something Known: Passwords

1st Scheme: Fixed Password

- A fixed password is a password that is used over and over again for every access.
- This scheme has several approaches.
  - Approach-1: User ID and password file
  - Approach-2: Hashing the password
  - Approach-3: Salting the password
  - Approach-4: Combination of something known and something possessed

2nd Scheme: One-time Password

- A one-time password is a password that is used only once.
- This kind of password makes eavesdropping and salting useless.
- This scheme also has several approaches.
  - Approach-1: List of passwords
  - Approach-2: Sequentially updated password
  - Approach-3: Sequentially updated password with hash function

Verification by Something Known: Fixed Passwords

First Approach- User ID and Password File:

- In the first approach, the system keeps a table (a file) that is stored by user identification.
- To access the system resources, the user sends her user ID and password in plaintext format to the system.
- The system uses the user ID to find the corresponding password in the table.
- If the password sent by the user matches the password in the table, access is granted; otherwise, it is denied.
- Figure below shows this approach.

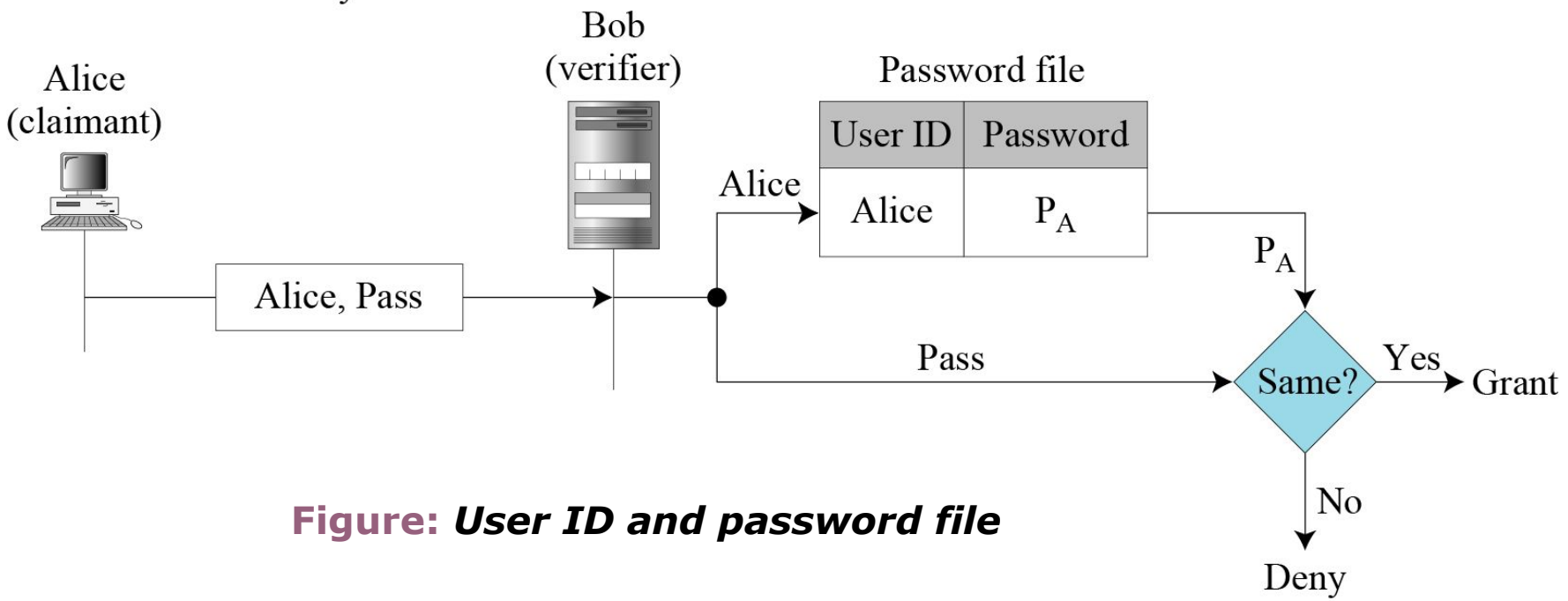


Figure: ***User ID and password file***

Possible Attacks on Fixed Passwords

- Using fixed password is subject to several kinds of attacks.

❑ Eavesdropping:

- Eve can watch Alice when she types her password.
  - ❖ Most system do not show the characters a user types.
- Eavesdropping can take a more sophisticated form. Eve can listen to the line and intercept the message, thereby capturing the password for her own use.

Possible Attacks on Fixed Passwords

❑ Stealing a Password:

- This occurs when Eve tries to steal Alice's password physically.

  - This can be prevented if Alice does not write down the password and instead she just commits it to memory.

    - ❖ For this reason, the password should be very simple or else related to something familiar to Alice.
    - ❖ But this makes the password vulnerable to other types of attacks.

Possible Attacks on Fixed Passwords

❑ Accessing a Password File:

- Eve can hack into the system and get access to the ID/password file.
- Eve can read the file and find Alice's password or even change it.
- To prevent this type of attack, the file can be read/write protected.
- However, most systems need this type of file to be readable by the public.

Possible Attacks on Fixed Passwords

❑ Guessing:

- Using a guessing attack, Eve can log into the system and try to guess Alice's password by trying different combinations of characters.
  - ❖ The password is particularly vulnerable if the user is allowed to choose a short password.
  - ❖ It is also vulnerable if Alice has chosen something trivial, such as her birthday, her child's name, or the name of her favorite actor.
- To prevent guessing, a long random password is recommended, something that is not very obvious.
  - ❖ **However, the use of such a random password may also create a problem**. Because she could easily forget such a password, Alice might store a copy of it somewhere, which makes the password subject to stealing.

Verification by Something Known: Fixed Passwords

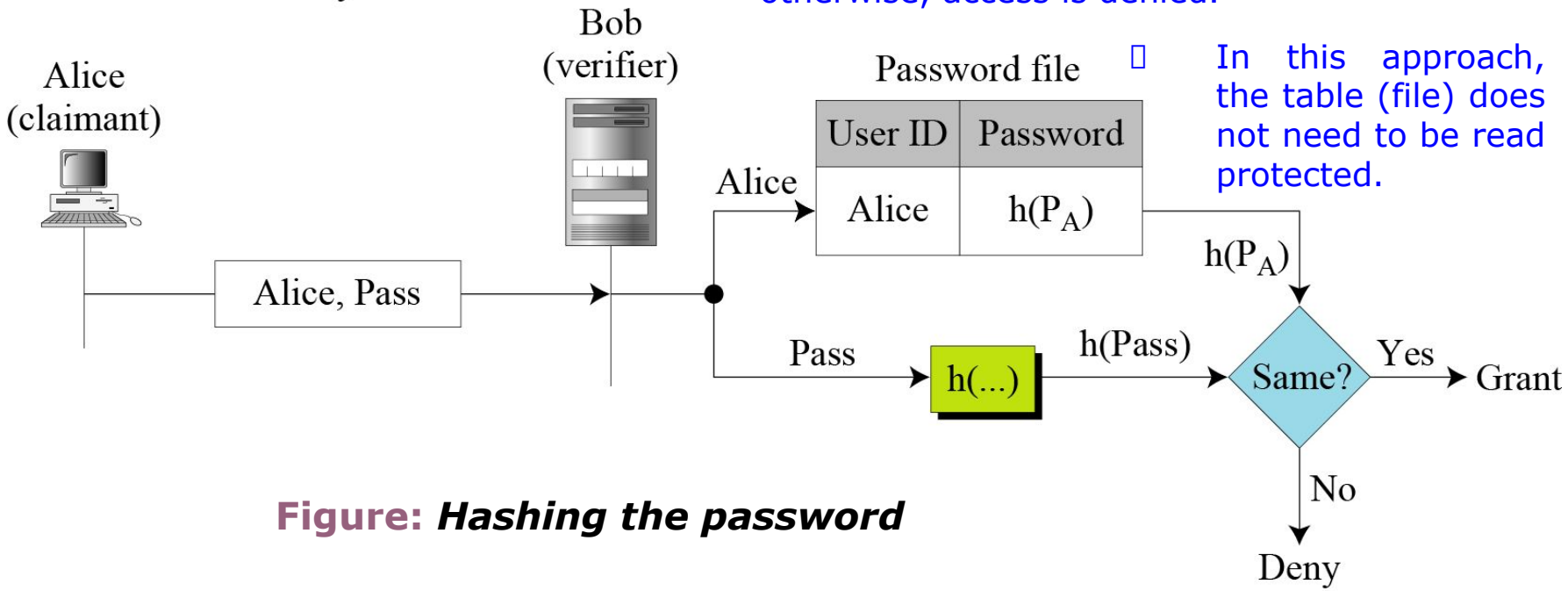
Second Approach- Hashing the Password:

- In the second approach, the system stores the hash of the password instead of the plaintext password in the table (a file) that is stored by user identification.
- Any user can read the contents of the file, but, because the hash function is a one-way function, it is almost impossible to guess the value of the password.
- When the password is created , the system hashes it and stores the hash in the password file.
- When the user sends her user ID and password, the system creates a hash of the password and then compare the hash value with the one stored in the file.

P_A: Alice's stored password

Pass: Password sent by claimant

- If there is a match, the user is granted access; otherwise, access is denied.



- In this approach, the table (file) does not need to be read protected.

Figure: ***Hashing the password***

Verification by Something Known: Fixed Passwords

Third Approach- Salting the Password:

- In the third approach, a random string, called the salt, is concatenated to the password when the password string is created.
- The salted password is then hashed.
- The ID, the salt, and the hash are then stored in the table (file).
- When a user asks for access, the system extracts the salt, concatenates it with the received password, makes a hash out of the result, and compares it with the hash stored in the file.

P_A: Alice's password
S_A: Alice's salt
Pass: Password sent by claimant

- If there is a match, access is granted; otherwise, it is denied.

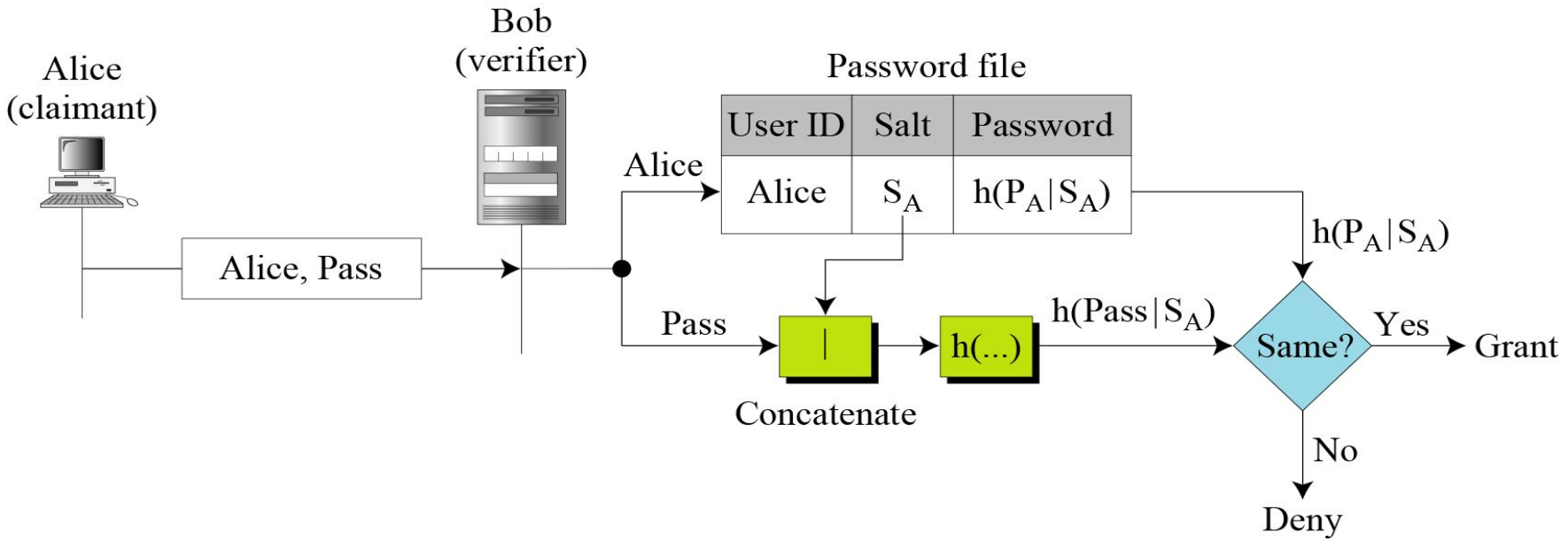


Figure: ***Salting the password***

Verification by Something Known: Fixed Passwords

Fourth Approach- Combination of something known and something possessed:

- In the fourth approach, two identification techniques are combined.
- A good example of this type of authentication is the use of an ATM card with a PIN (personal identification number).
- Here, the ATM card belongs to the category “something possessed” and the PIN belongs to the category “something known”.
- The PIN is a password that enhances the security of the card.
- If the card is stolen, it can not be used unless the PIN is known.
- The PIN number is very short, so it is easily remembered by the owner. Hence this makes it vulnerable to the guessing type of attack.

Verification by Something Known: One-time Password

First Approach- List of Passwords:

- In the first approach, the user and the system agree upon a list of passwords.
- Each password on the list can be used only once.
- There are some drawbacks to this approach:
  - The system and the user must keep a long list of passwords.
  - If the user does not use the password in sequence, the system needs to perform a long search to find the match.
- This approach makes eavesdropping and reuse of the password useless.
- The password is valid only once and can not be used again.

Verification by Something Known: One-time Password

Second Approach- Sequentially Updated Password:

- In the second approach, the user and the system agree to sequentially update the password.
- They agree on an original password, P_1, which is valid only for the first access.
- During the first access, the user generates a new password, P_2, and encrypts this password with P_1 as the key.
- P_2 is the password for the second access.
- During the second access, the user generates a new password, P_3, and encrypts this password with P_2 as the key.
- P_3 is the password for the third access.
- And so on..
- If Eve can guess the first password (P_1), she can find all of the subsequent passwords.

Verification by Something Known: One-time Password

Third Approach- Sequentially Updated Password with Hash Function:

- This approach is devised by Leslie Lamport where the user and the system create a sequentially update the password using a hash function.
- In this approach, the user and the system agree upon an original password, P_0, and a counter, n.
- The system calculates $h^n(P_0)$, where h^n means applying a hash function n times. In other words,

$$h^n(x) = h(h^{n-1}(x)) \quad h^{n-1}(x) = h(h^{n-2}(x)) \quad \ldots \quad h^2(x) = h(h(x)) \quad h^1(x) = h(x)$$

- The system stores the identity of Alice, the value of n, and the value of $h^n(P_0)$.
- When the system receives the response of the user in the third message, it applies the hash function to the value received to see if it matches the value stored in the entry. If there is a match, access is granted; otherwise, it is denied. The system then decrements the value of n in the entry and replaces the old value of the password $h^n(P_0)$ *with the new value* $h^{n-1}(P_0)$.
- When the user tries to access the system for the second time, the value of the counter it receives is n-1. The third message from the user is now $h^{n-2}(P_0)$. *When the* system receives the message, it applies the hash function to get $h^{n-1}(P_0)$, *which can be compared with the updated entry.*

Verification by Something Known: One-time Password

Third Approach- Sequentially Updated Password with Hash Function (continue...):

- The value of n in the entry is decremented each time there is an access. When the value of n becomes 0, the user can no longer access the system; everything must be set up again. For this reason, the value of n is normally chosen as a large number such as 1000.
- Figure below shows how the user accesses the system for the first time.

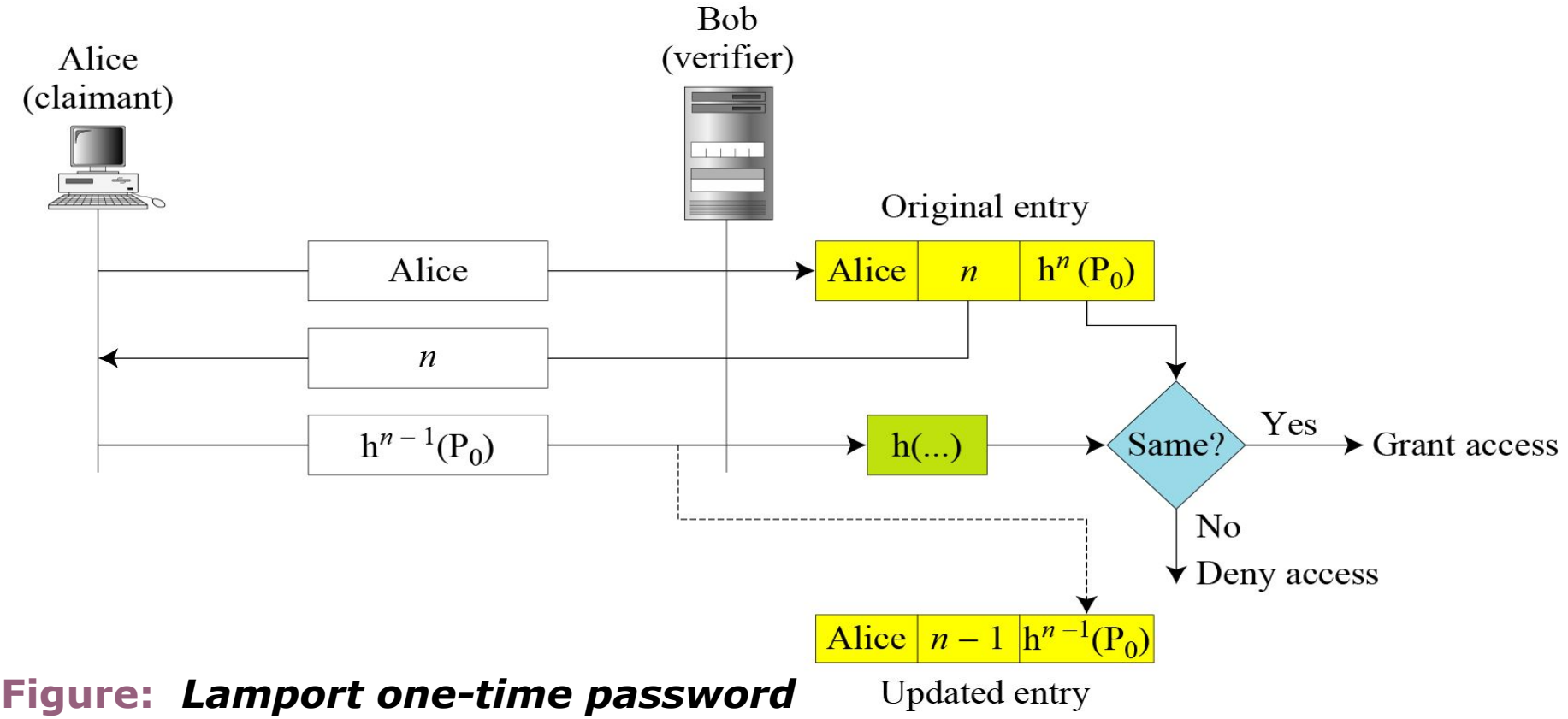


Figure: ***Lamport one-time password***

Authentication by Inherence Factor

- In computer system, authentication can be done traditionally based on:
  - something that one has e.g. passport, driving license, ID card, credit card, smart card.
  - or something one knows e.g. PIN, password, passphrase.
- Things like passport, credit cards tend to get stolen or lost and passwords are often forgotten or disclosed.
- To achieve more reliable verification or identification, we should use something that really characterizes the given person.
- With the help of biometrics, it is possible to confirm or establish an individual's identity based on "who he is", rather than by "what he possesses" (e.g., an ID card) or "what he knows" (e.g., a password).
- Biometric authentication has grown in popularity as a way to provide personal identification. It is highly reliable, because physical human characteristics are much more difficult to forge than security codes, passwords and hardware keys.
- Biometrics offer automated methods of verification or identification based on the measurable physiological or behavioral characteristics such as a fingerprint or a voice sample that is unique. These characteristics should not be duplicable, but it is unfortunately often possible to create a copy that is accepted by the biometric system as a true sample.

What is Biometrics?

- Biometrics is the science and technology of **measuring and analyzing biological data for recognizing a person**.
  - It is an **automated measurement** of **physiological and/or behavioral characteristics** of a human being such as a fingerprint or a voice sample that can be used **to identify an individual or verify the claimed identity of an individual**.
- Let us spread the definition into its three major components, shown above in different colors.
- These components will determine what is and what is not a biometric and also its different types and functionalities.

1st Component: Automated Measurement

- It means no human intervention or involvement is required. Biometrics are automated in as much as the processes involved in sample acquisition, feature extraction, record retrieval, and algorithm-based matching are computerized or machine-based.
- Comparison takes place in Real-Time. The record retrieval and comparison against another measurement must take place in Real-Time
  - DNA sampling is not a Biometric measurement, because today it still requires human intervention and it is not done in real time.

What is Biometrics?

2nd Component: Physiological and/or Behavioral Characteristics

- The second component of the definition determines the two main biometric categories:
  1. Behavioral:
     - The behavioral characteristics measure the movement of a user, when users walk, speak, type on a keyboard or sign their name.
  2. Physiological:
     - The physiological characteristics would be the physical human traits like fingerprints, hand shape, eyes and face, veins, etc.

What is Biometrics?

3rd Component: Identify an Individual or Verify the Claimed Identity of an Individual

- The last component of the definition categorizes the two types of biometric functionalities.

1. Identification Systems:
   - Identification (also called search) occurs when the identity of a user is priori unknown. In this case the user's biometric data is matched against all the records in the database as the user can be anywhere in the database or he/she actually does not have to be there at all.
   - An example of an Identification system using biometrics would be: You approach an ATM with no card, no claimed identity, no PIN. The ATM scans your iris and determines who you are and gives you the access to your money.

2. Verification Systems:
   - Verification occurs when the user claims to be already enrolled in the system (presents an ID card or login name); in this case the verification biometric data obtained from the user is compared to the user's data already stored in the database.
   - An example of a Verification System using biometrics would be: You approach an ATM and swipe a card or enter an account number. The ATM scans your iris and uses it as a password to authenticate you are the rightful owner of the card and therefore give you access to your money.

Why Biometric Application?/ Advantages

❑ Biometric authentication is highly reliable

- Biometric authentication is highly reliable, **because** physical human characteristics are unique and much more difficult to forge than the traditional use of security codes or PINs, passwords and hardware keys.

❑ It provides increased security

- Tokens such as smart card, magnetic stripe cards, ID cards, physical keys, can be lost, stolen, duplicated or left at home. Password can be forgotten, shared or observed. But biometric data cannot be guessed, stolen or shared among users, therefore providing increased security to a system.

❑ Biometrics relieves user from the burden of remembering password

- Now-a-days, people are asked to remember a multitude of passwords and Personal Identification Number (PINs) for computer accounts, banks, ATMs, E-Mail, wireless, phones, websites and so forth. But, in theory, biometrics relieves the user from the burden of having to remember a password, or worse multiple passwords for different systems within an organization.

Why Biometric Application?/ Advantages

- **Biometric provides fast, easy, accurate, reliable and less expensive authentication for a variety of application**
  - Biometrics holds the promise of fast, easy, accurate, reliable and less expensive authentication for a variety of application.

- **Some biometric authentication systems is more speedy**
  - Another advantage of biometric authentication systems may be their speed. The authentication of a habituated user using an iris-based identification system may take 2 (or 3) seconds while finding your key ring, locating the right key and using it may take some 5 (or 10) seconds.

- **In large-scale identification systems, biometric applications offer fraud detection and fraud deterrence**
  - In addition to authentication, biometric applications are employed in large-scale identification systems, where they offer two important benefits: **fraud detection** and **fraud deterrence**. For example, one person can claim multiple identities, using fraudulent documents, to receive benefits from a public program. Without the use of biometrics, it would be extremely difficult to discover that the person has multiple registrations, considering the large volume of data stored in the system. Biometrics can therefore contribute to fraud detection.

Disadvantages of Biometric Authentication

Biometric authentication system is better than that of other identification systems.

- So why do not we use biometrics everywhere instead of passwords or tokens?
- Nothing is perfect and biometric authentication methods also have their own shortcomings.

1. Even if no biometric system is really dangerous, users are occasionally afraid of something they do not know much about. In some countries people do not like to touch something that has already been touched many times (e.g., biometric sensor), while in some countries people do not like to be photographed or their faces are completely covered.

Disadvantages of Biometric Authentication

3. Biometric systems may violate user's privacy.
   - Biometric characteristics are sensitive data that may contain a lot of personal information. The **DNA** (being the typical example) contains (among others) the user's preposition to diseases. This may be a very interesting piece of information for an **insurance company**.
   - The **body odor can provide** information about user's recent activities.
   - It is also told that people with **asymmetric fingerprints** are more likely to be homosexually oriented, etc.
4. Although good for user authentication, biometrics cannot be used to authenticate computers or messages.
   - **Biometric characteristics are not secret** and therefore **they cannot be used to sign messages or encrypt documents.** If my fingerprint is not secret there is no sense in adding it to documents we have written. Anyone else could do the same.

Disadvantages of Biometric Authentication

5. Use of biometric systems may also imply loss of anonymity.
   - ❖ While one can have multiple identities when authentication methods are based on something the user knows or has, biometric systems can sometimes link all user actions to a single identity.

6. The performance of biometric systems is not ideal.
   - ❖ Biometric systems still need to be improved in the terms of accuracy and speed.
   - ❖ Biometric systems with the false rejection rate under 1% are still rare today. Although few biometric systems are fast and accurate enough to allow identification, most of current systems are suitable for the verification only, as the false acceptance rate is too high.

Disadvantages of Biometric Authentication

7. The fail to enroll rate brings up another important problem. Not all users can use any given biometric system. People without hands cannot use fingerprint or hand-based systems. Visually impaired people have difficulties using iris or retina based techniques. As not all users are able to use a specific biometric system, the authentication system must be extended to handle users falling into the FTE category. This can make the resulting system more complicated, less secure or more expensive.

8. Even enrolled users can have difficulties using a biometric system. The FTE rate says how many of the input samples are of insufficient quality. Data acquisition must be repeated if the quality of input sample is not sufficient for further processing and this would be annoying for users.

9. Lack of standards (or ignorance of standards) may also posses a serious problem. Two similar biometric systems from two different vendors are not likely to interoperate at present.

Biometric Devices

- Biometric device is a device that translates personal characteristics into a digital code that is compared with a digital code stored in the database.
- Some biometric devices include:
  - Fingerprint Recognition,
  - Facial Recognition
  - Hand Geometry,
  - Iris Scanning,
  - Retinal Scanning,
  - Voice Recognition and
  - Signature Verification

Types of Biometric Technologies

- There are two major classifications of biometric technologies depending on the number of distinctive characteristics each technology offer:
  - Those that do identification and verification, like
    - Finger scan
    - Iris scan
    - Retina scan
    - Facial scan (optical and infrared)
  - Those that do verification only, like
    - Hand Geometry
    - Voice Print
    - Keystroke Behavior
    - Signature
- Biometric technologies that do identification and verification will have more distinctive characteristics to work with, than the ones that only do verification.

Biometric Technology: Finger Scan/ Finger Print

- A fingerprint is an impression (ছাপ) of the friction ridges of all or any part of the finger.
- This is a technology that uses the unique fingerprint patterns present on the human finger to identify or verify the identity of the individual.
- Several acquisition techniques can be used:
  - optical scanning
  - capacitive scanning (silicon chip)
  - ultrasound scanning
- These characteristics or minutiae (as they are called), are
  - Crossover
  - Core
  - Bifurcations
  - Ridge ending
  - Island
  - Delta
  - Pore
- The finger print obtained from an Optical Fingerprint Reader is shown in figure here.

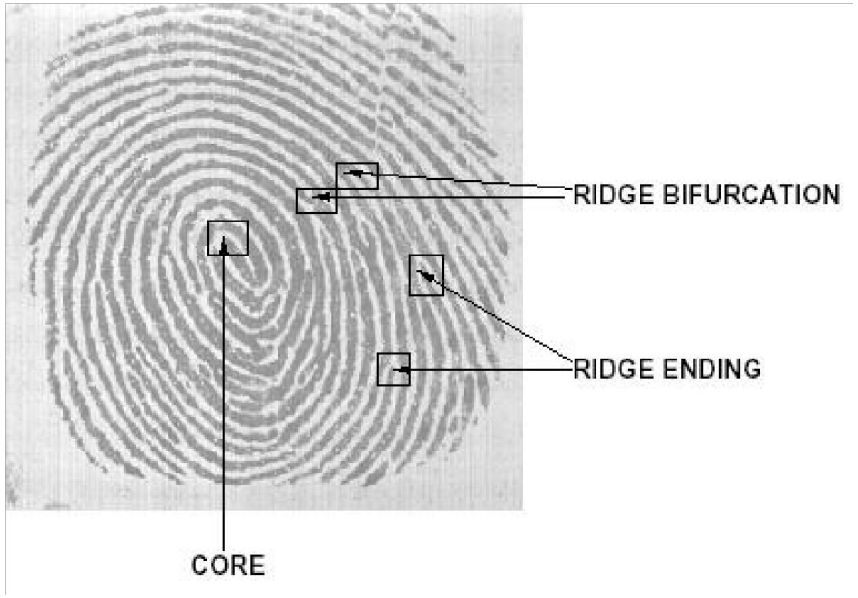


Figure: Fingerprint Bitmap

Biometric Technology: Finger Scan/ Finger Print

- Fingerprint samples like the one shown in the figure above, typically don't have all the minutiae types available. It is desirable but not always possible.
- The traditional fingerprint method uses the ink to get the finger print onto a piece of paper. This piece of paper is then scanned using a traditional scanner.
- Now in modern approach, live finger print readers are used. Today we may find many automated fingerprint identification systems (AFIS), because of the high quality scanners available.
- Fingerprints do not change over lifetime and that no two fingerprints are exactly alike.
- This technique is used mostly for forensic and background checks and is being used in both logical and physical security.
- Features of Fingerprint scanner:
  - Good accuracy
  - Used for both identification and verification
  - Low cost
  - Problem when skin is too dry or too wet
  - Problem with dirt

Biometric Technology: Iris Scan

- Iris is the colored area that surrounds the pupil.
- It forms during gestation and remains the same for the rest of one's life. It is well protected and extremely difficult to be modified.
- Iris patterns are unique for individuals and are obtained through video based image acquisition system in the distance of 10- 40 cm of camera.
- Iris scan measures the unique characteristics (or pattern) of the iris.
- So far, the technology has been successfully implemented in ATMs and is currently being promoted for desktop usage.
- This technique, just like finger scan, is being used in both logical and physical security.
- Once the gray scale image of the eye is obtained then the software tries to locate the iris within the image.
- Image of an iris is shown in the figure below.

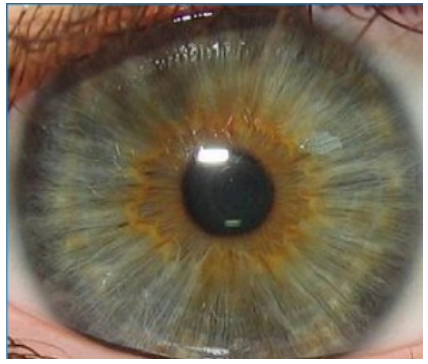
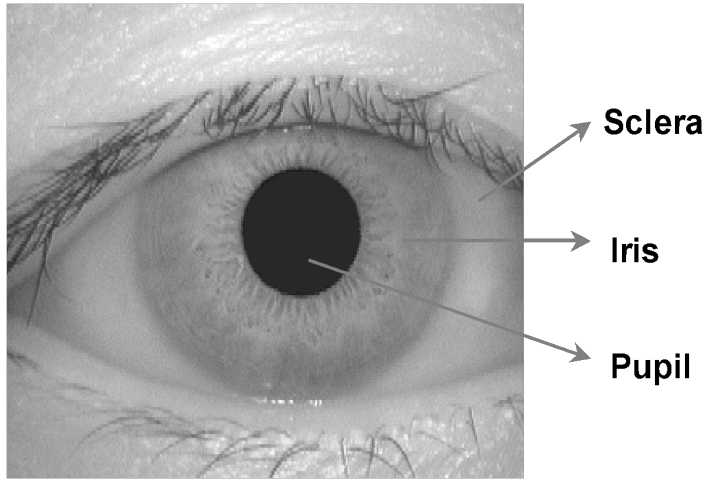
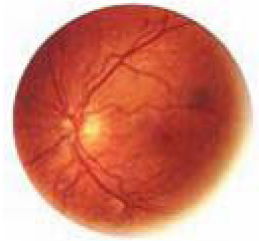


Figure: Iris Bitmap



Biometric Technology: Retina Scan

- It is based on the blood vessel pattern in the retina of the eye as the blood vessels at the back of the eye have a unique pattern, from eye to eye and person to person.
- Retina is not directly visible and so a coherent infrared light source is necessary to illuminate the retina. The infrared energy is absorbed faster by blood vessels in the retina than by the surrounding tissue.
- After capturing, the image of the retina blood vessel pattern is then analyzed.
- Retina scan requires significant more effort to use than Iris scan, and it is more challenging because the slightest movement causes rejection by the system. It also needs more sophisticated cameras than iris scan.
- Figure below shows the image of retina.



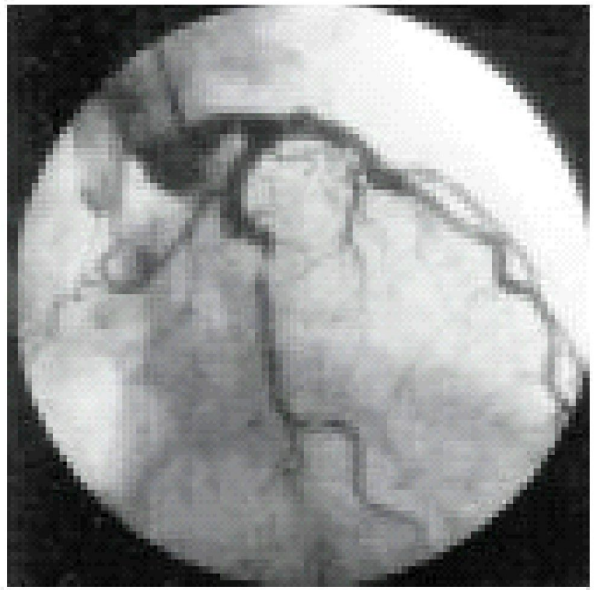


Figure: Retina Bitmap

Biometric Technology: Facial Scan

- A facial recognition technique is an application of computer for automatically identifying or verifying a person from a digital image or a video frame from a video source. It is the most natural means of biometric identification.

- Facial recognition is a form of computer vision that uses faces to attempt to identify a person or verify a person's claimed identity. Regardless of specific method used, the facial recognition is accomplished in a five step process.

1. Acquiring the image of an individual's face:
   - There are two ways to acquire an image-
     - Digitally scan an existing photograph
     - Acquire a live picture of a subject

2. Locate image of face:
   software is used to locate the faces in the image that has been obtained

3. Analysis of facial image:
   - software measures face according to is peaks and valleys (nodal points)
   - focuses on the inner region of the face known as the "golden triangle"
   - nodal points are used to make a face print

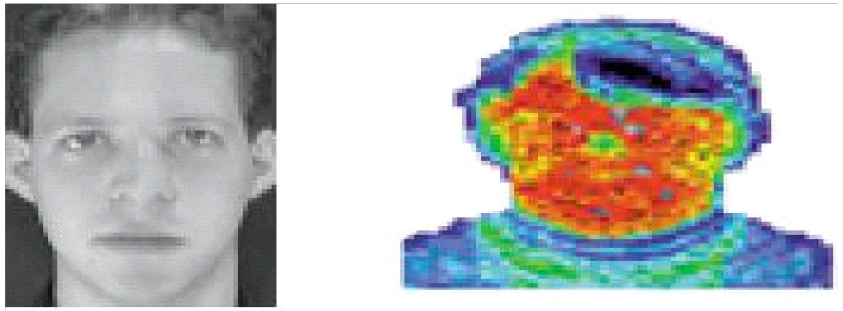
4. Comparison:
   - the face print created by the software is compared to all face prints the system has stored in its database.

5. Match or no match:
   - software decides whether or not any comparisons from step 4 are close enough to declare a possible match.

Biometric Technology: Facial Scan

- In the case of facial scan, it measures facial features like the-
  - Distance between the eyes.
  - Distance between the eyes and nose ridge.
  - Angle of a cheek.
  - Slope of the nose.
  - Thickness of the lips.
  - Facial Temperatures.

- It is the most common Biometric technique used to obtain a personal identification.
- This technique is used at all US embassies worldwide, and government agencies.
- Also used to guarantee uniqueness against an image databases usually to prevent identity theft.
- Many ATMs and casinos around the country, use this techniques to identify users.



face

facial thermogram

Biometric Technology: Hand Geometry/ Hand Scan

- This technology uses distinctive features of the hand, such as geometry of hand and fingers, for identity verification.
- It is based on the fact that nearly every person's hand is shaped differently and that the shape of a person's hand does not change after certain age.
- These techniques include the estimation of length, width, thickness and surface area of the hand. Various method are used to measure the hands.
- Recent uses include the I. N. S. pass System, which scans a hand of frequent travelers, so instead of presenting a passport for authentication these frequent travelers swipe a card and do a hand scan. It is both convenient to consumers and frees up human resources to attend for more higher risk passengers.
- The hand geometry can change due to age and health conditions.
- Figure below shown a hand geometry scanner.

Figure: Hand Geometry Scanner

Biometric Technology: Voice Scan

- This is a technology that uses the unique aspects of the individual's voice for identification or authentication purposes.
- This technique is text-dependent, which means that the system cannot verify any phrase spoken by the user, but rather a specific phrase associated with that user's account.
- Voice is also physiological trait because every person has different pitch, but voice recognition is mainly based on the study of the way a person speaks, commonly classified as behavioral.
- Your voice can vary with age, illness and emotions.
- Speaker recognition uses a microphone to record the voice.
- The most common use of voice scan biometric systems is where a telephone is already being used.
- For instance home arrest verification is a very common use. Any time of the day or night a computer calls the home of a person under home arrest, and that person has to answer the phone and speak a passphrase to be authenticated.

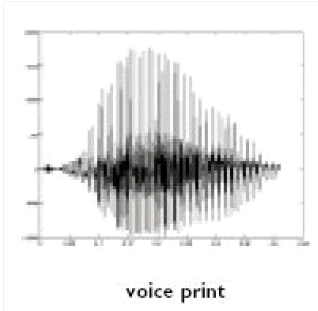


Figure: Voice print

Biometric Technology: Signature Scan

- This technology uses the human written signature for identity verification.
- In the case of signature scan, it measures the speed, pressure, direction, acceleration, stroke order, length of the strokes and image of a signature.
- So it is not only the signature image as it is commonly believed.
- If a signature from a user is already captured, this biometric technology adds an extra level of security with non-repudiation.
- The most obvious and important advantage of this technique is that a fraudster cannot glean any information on how to write the signature by simply looking at one that has been previously written.
- There are various kinds of devices used to capture the signature dynamics. These are either traditional tablets or special purpose devices.
- Figure below shows a signature taken using Tablet.

Figure: A Signature taken using Tablet.

Biometric Technology Still in Drawing Board

Vein scan :

- Another fairly new technology that uses the vein patterns on the back of the hand for identification and authentication.
- The technology has the potential of delivering high accuracy, in addition to the advantage of being non-intrusive to the user.
- Vein identification has been recently implemented in commercial products, such as VeinID.

Lip movement :

- camera captures images of how user lips move while user speaks a passphrase.

Body Odor:

- The body odor biometrics is based on the fact that virtually each human smell is unique. The smell is captured by sensors that are capable to obtain the odor from nonintrusive parts of the body such as the back of the hand. The use of body odor sensors brings up the privacy issue as the body odor carries a significant amount of sensitive personal information. It is possible to diagnose some diseases or activities in the last hours (like sex, for example) by analyzing the body odor.

Biometric Technology Still in Drawing Board

DNA:

- A relatively new technology that relies on the analysis of DNA sequences for identification and authentication.
- DNA sampling is rather intrusive at present and requires a form of tissue, blood or other bodily sample.
- This method of capture still has to be refined. So far the DNA analysis has not been sufficiently automatic to rank the DNA analysis as a biometric technology.
- The analysis of human DNA is now possible within 10 minutes. As soon as the technology advances so that DNA can be matched automatically in real time, it may become more significant. At present Biometric Systems DNA is very entrenched in crime detection and so will remain in the law enforcement area for the time being.
- The DNA technology raises many concerns over "privacy issues, invasiveness and data misuse and currently cannot be done fully automated.

Biometric Application

- Biometric authentication is highly reliable, because physical human characteristics are much more difficult to forge than security codes, passwords, hardware keys sensors, fast processing equipment and substantial memory capacity, so the system are costly.
- Biometric-based authentication applications include workstation and network access, single sign-on, application logon, data protection, remote access to resources, transaction security, and Web security.
- The promises of e-commerce and e-government can be achieved through the utilization of strong personal authentication procedures using biometric application.

Biometric Application

- Secure electronic banking, investing and other financial transactions, retail sales, law enforcement, and health and social services are already benefiting from these technologies.
- Biometric technologies are expected to play a key role in personal authentication for large scale enterprise network authentication environments, Point-of-Sale and for the protection of all types of digital content such as in Digital Rights Management and Health Care applications.
- Utilized alone or integrated with other technologies such as smart cards, encryption keys and digital signatures, biometrics is anticipated to pervade nearly all aspects of the economy and our daily lives.

Key Management & Certification

Dr. Risala Tasin Khan

Professor

IIT, JU

Problems with Symmetric Key and Public Key

Symmetric key problem:

- How do two entities establish shared secret key over network?

Solution:

- Trusted key distribution center (KDC) acting as intermediary between entities.

Public key problem:

- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?

Solution:

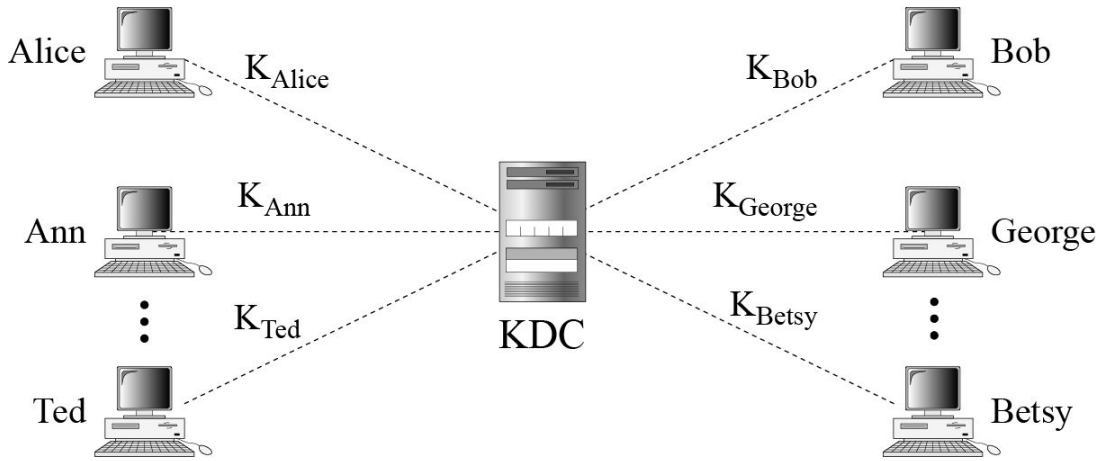
- Trusted certification authority (CA)

Key-Distribution Center: KDC

- In symmetric-key cryptography, a shared secret key is needed to be exchanged between two parties involved in a communication.
- If Alice and Bob want to communicate, they need a way to exchange a secret key between them; if Alice wants to communicate with one million people, how can she exchange one million keys with one million people? Using the Internet is definitely not a secure method. It is obvious that we need an efficient way to maintain and distribute secret keys.
- A practical solution to maintain and distribute secret keys is the use of a trusted third party, referred to as a key-distribution center (KDC).
- To reduce the number of keys, each person establishes a shared secret key with the KDC, as shown in the figure below.
- A secret key is established between the KDC and each member.

❖ Alice has a secret key K_{Alice} with the KDC; Bob has a secret key K_{Bob} with the KDC; and so on.

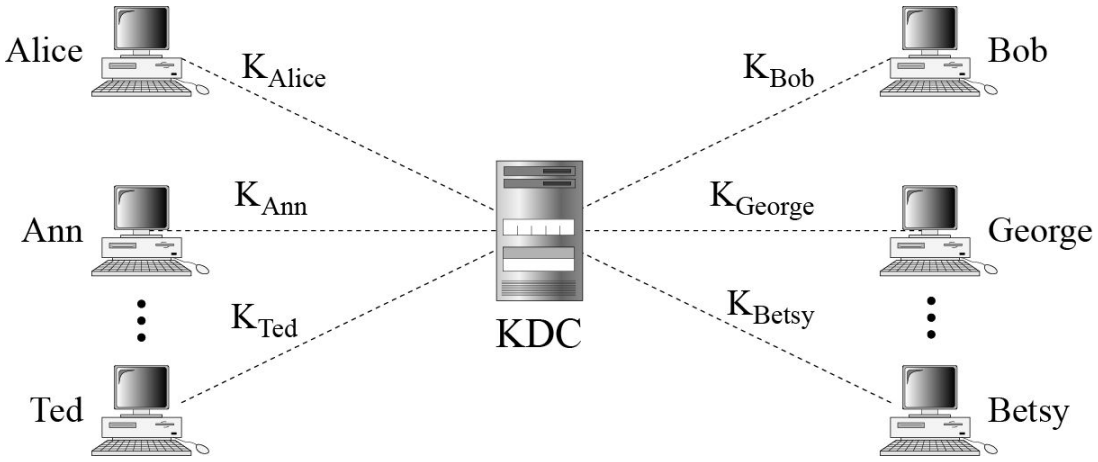
❖ They communicate with each other via the KDC.



Key Distribution Center-KDC (Cont..)

Q: How can Alice send a confidential message to Bob using the KDC?

- The process is as follows:
  1. Alice sends a request to the KDC stating that she needs a session (temporary) secret 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 parties.
- The established session key between Alice and Bob with the KDC is used to authenticate Alice and Bob to the KDC which prevents Eve from impersonating either of them.



How KDC Works

APPROACH-1

In a secure communication scenario between two parties like Bob and Alice, a Key Distribution Center (KDC) helps establish a shared, secure key between them, allowing for secure, authenticated communication. Here's how this process typically works:

1. **Registration with the KDC**:
   - Both Bob and Alice register with the KDC beforehand, each receiving a unique secret key shared only with the KDC (let's call these keys K_Bob and K_Alice, respectively).
   - These keys enable secure communication between the KDC and each user.
2. **Request for Communication**:
   - Suppose Bob wants to communicate securely with Alice.
   - Bob sends a request to the KDC, asking for permission to communicate with Alice.
   - This request is encrypted with Bob's secret key (K_Bob) to ensure its authenticity and integrity.
   - The KDC informs Alice about Bob's request.
   - If Alice agrees, a session key is created between the two parties.

3. KDC Generates a Session Key:

- The KDC generates a **session key** (K_session) that Bob and Alice will use to encrypt their messages to each other.
- The KDC creates two copies of this session key, one encrypted with K_Bob (for Bob) and the other encrypted with K_Alice (for Alice).

4. KDC Sends Session Key to Both Parties:

3. After receiving the message from Bob, the KDC creates a ticket which contains the identities of Alice and Bob, and the session key (K_session). The KDC then encrypts the ticket using Alice's key (K_Alice).

- The session key along with the encrypted ticket is again encrypted using Bob's key K_Bob. Then it is sent to Bob. Bob receives the message. Decrypts it, and extracts the session key.
  - ❖ Bob can not decrypt the ticket, which is only for Bob.
  - ❖ In this message, Bob is actually authenticated to the KDC, because only she can open the whole message using her secret key with the KDC.

5. Bob Sends the Ticket to Alice:

- Bob sends the ticket to Alice, who decrypts it using her secret key, K_Alice, to retrieve K_session.
- Now, both Bob and Alice have K_session, a shared secret key they can use for secure communication.
  - ❖ In this message, Bob is authenticated to the KDC, because only he can open the ticket.
  - ❖ Since Bob is authenticated to the KDC, he is also authenticated to Alice, who trusts the KDC. In the same way, Alice is also authenticated to Bob, Bob trusts the KDC and the KDC has sent Bob the ticket that includes the identity of Alice.

6. Secure Communication:

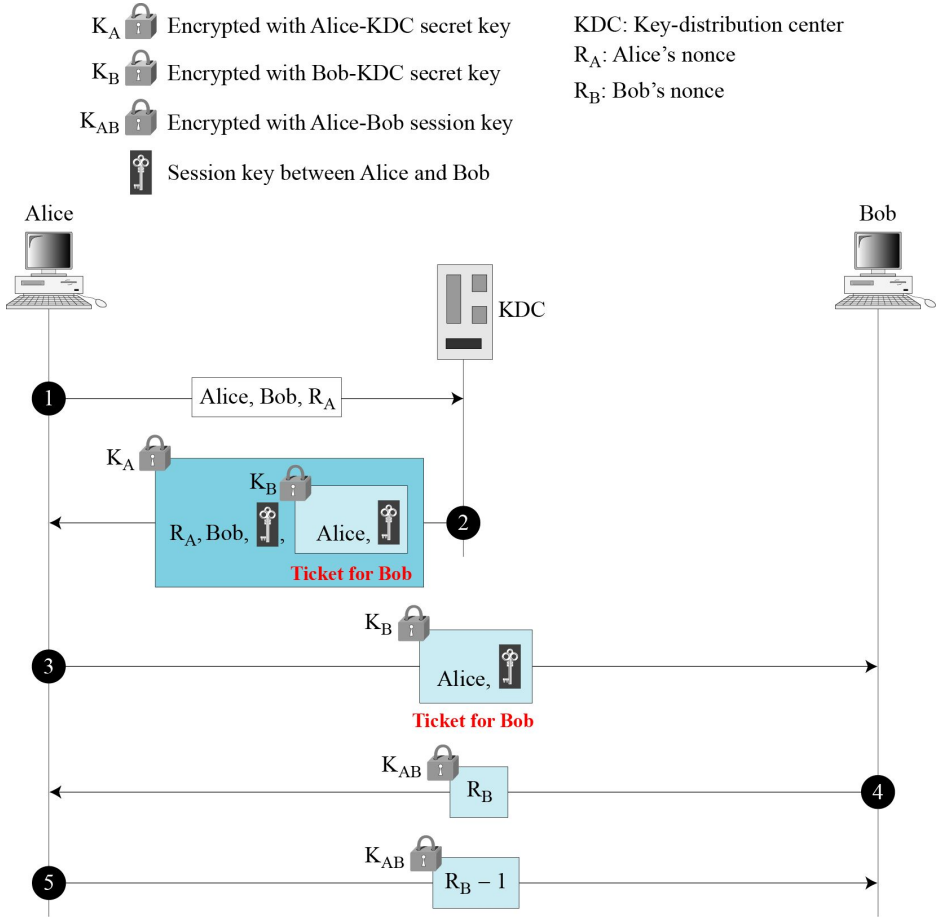
- Using K_session, Bob and Alice can now exchange encrypted messages that only they can decrypt, ensuring the confidentiality and integrity of their communication.

- By using a KDC to generate and distribute session keys, Bob and Alice can communicate securely without having to establish a direct key exchange themselves.
- This system also allows the KDC to act as a trusted authority, simplifying key management across the network.

Unfortunately, this protocol has a flaw. Eve can use the replay attack. That is, she can save the message in step 5 and replay it later.

APPROACH-2: Needham-Schroeder Protocol

- This protocol is a foundation for many other protocols that uses multiple challenge-response interactions between parties to achieve a flawless protocol.
- Needham and Schroeder uses two nonce: R_A and R_B.
- Figure below shows the five steps used in this protocol.



- The steps for creating session key using Needham-Schroeder protocol is summarized below:

1. Alice sends a message to the KDC that includes her nonce (R_A), her identity, and Bob's identity.
2. After receiving the message from Alice, the KDC creates a ticket which contains the identities of Alice, and the session key K_{AB} between Alice and Bob. The KDC then encrypts the ticket using Bob's key (K_B).

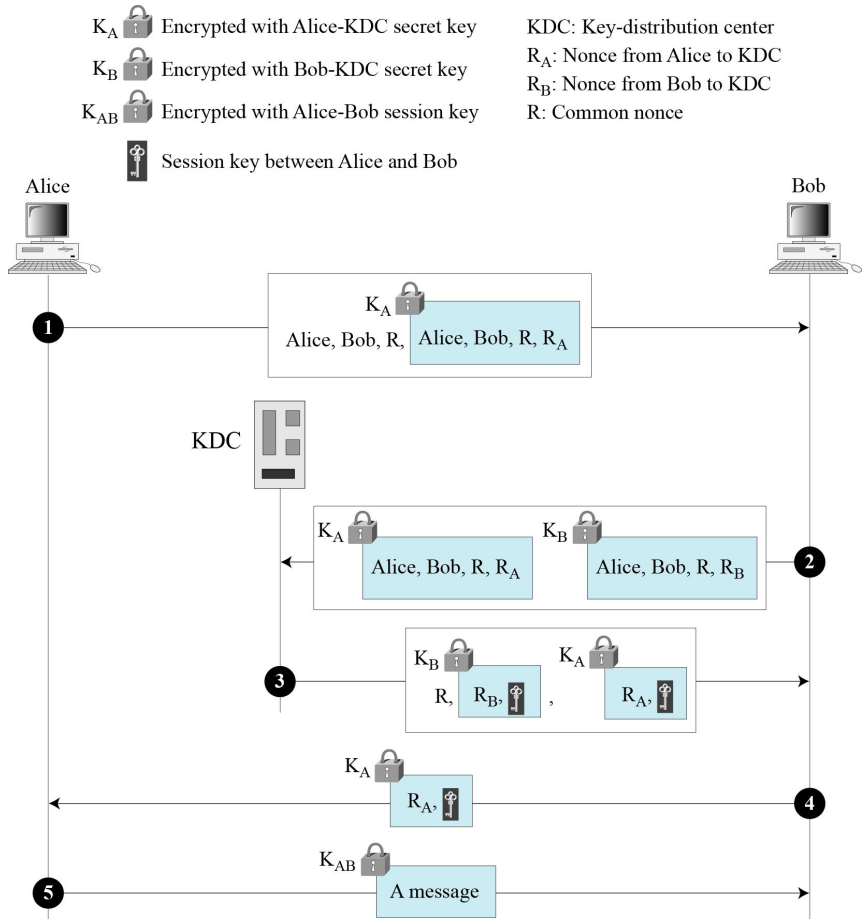
- Alice's nonce R_A, Bob's identity, the session key K_{AB}, and the encrypted ticket for Bob is again encrypted using Alice's key K_A. Then it is sent to Alice. Alice receives the message. Decrypts it, and extracts the session key.
  - Alice can not decrypt the ticket, which is only for Bob.
  - In this message, Alice is actually authenticated to the KDC, because only she can open the whole message using her secret key with the KDC.

1. After getting the session key, Alice sends the encrypted ticket to Bob. Bob decrypts the ticket using his key K_B and comes to know the session key.
2. Bob encrypts his nonce R_B with the session key K_{AB} and sends the encrypted nonce to Alice.
3. Alice decrypts Bob's encrypted nonce and extracts it.
4. She then decrease Bob's nonce by 1 (i.e. R_B-1), encrypts it using the session key and finally sends it back to Bob.

- *The response carries $R_B - 1$ instead of R_B.*

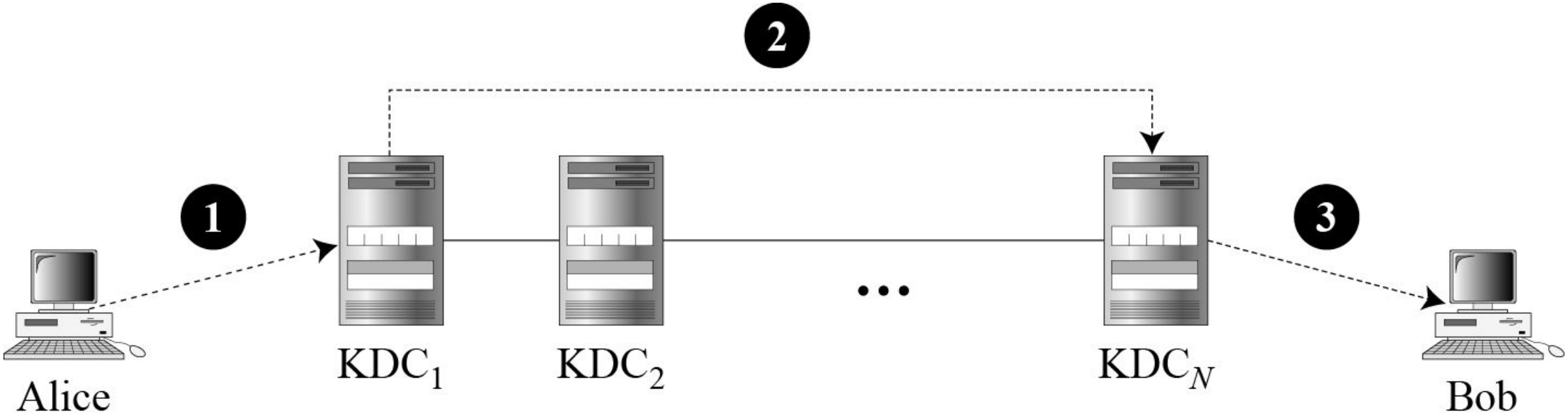
Approach-3: Otway-Rees Protocol

- The steps for creating session key using this approach is given below:
  1. Alice sends a message to Bob that includes a common nonce R, the identities of Alice and Bob, and an encrypted ticket for KDC that includes Alice's nonce R_A (a challenge for the KDC to use), a copy of the common nonce R, and the identities of Alice and Bob.
  2. Bob creates the same type of ticket, but with his own nonce R_B. Both tickets are sent to the KDC.
  3. The KDC creates a message that contains the common nonce R, a ticket for Alice and a ticket for Bob; the message is sent to Bob. The tickets contain the corresponding nonce R_A or R_B, and the **session key K_{AB}**.
  4. Bob sends Alice her ticket.
  5. Alice sends a short message encrypted with her session key K_{AB} to show that she has the session key.



Using Multiple KDCs

- When the number of people using a KDC increases, the system becomes unmanageable and a bottleneck can result.
- Using multiple KDCs can solve this problem.
- There are two approaches to use multiple KDCs:
  1. Flat multiple KDCs
  2. Hierarchical multiple KDCs

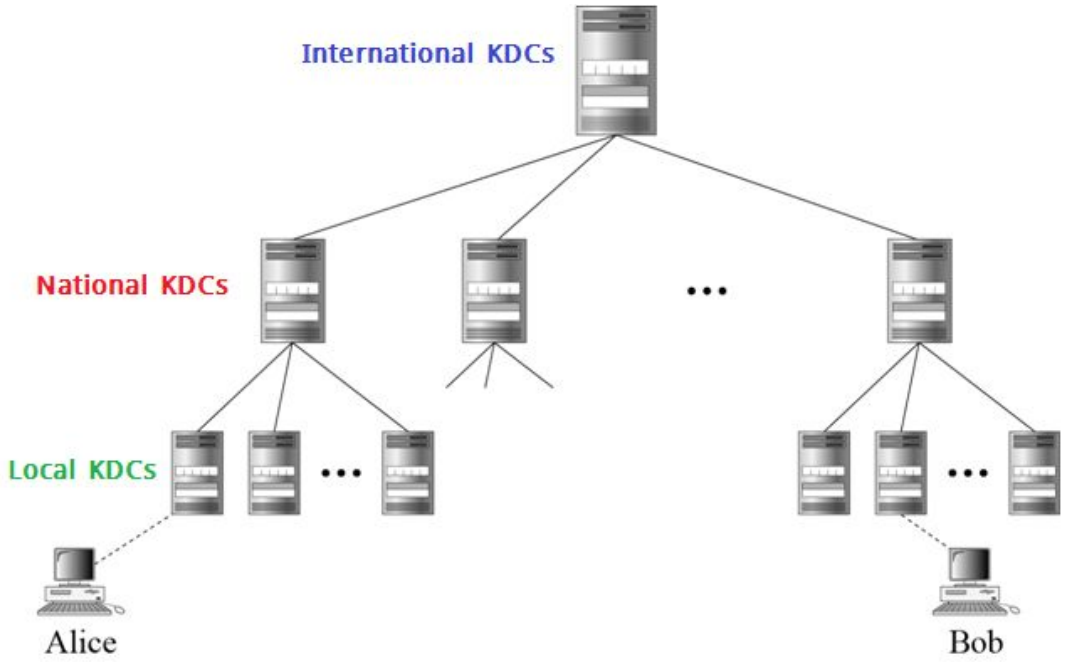


Flat Multiple KDCs

- In this approach, the world is divided into domains where each domain can have one or more KDCs.
  - If Alice want to send a confidential message to Bob, who belongs to another domain, Alice contacts her KDC, which in turn contacts the KDC in Bob's domain.
  - The two KDCs can create a secret key between Alice and Bob.

Hierarchical Multiple KDCs

- In hierarchical multiple KDC system, the concept of flat multiple KDCs can be extended with one or more KDCs at the top of the hierarchy.
- For example, there can be local, national and international KDCs.
- When Alice needs to communicate with Bob, who lives in another country, she needs her request to a local KDC, the local KDC relays the request to the national KDC, the national KDC relays the request to an international KDC.
- The request is then relayed all the way down to the local KDC where Bob lives.



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.
- It is named so after the three-headed dog in Greek mythology that guards the gates of Hades.
- Originally designed at MIT, it has gone through several versions.

Servers Involved in Kerberos

- Three servers are involved in the Kerberos protocol:
  1. 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 ($K_{A\text{-}TGS}$) to be used between Alice and the TGS, and
       - ✔ sends a ticket for the TGS.
  2. Ticket-Granting Server (TGS):
     - The TGS –
       - ✔ issues a ticket for the real server (Bob).
       - ✔ provides the session key ($K_{A\text{-}B}$) between Alice and Bob.
  3. Real Server:
     - The real server (Bob) provides services for the user (Alice).
     - Kerberos is designed for a client-server program, such as FTP, in which a user uses the client process to access the server process.
     - Kerberos is not used for person-to-person authentication.

Servers Involved in Kerberos:

- Figure below shows the relationship between these three servers.
- In our examples and figures, Bob is the real server and Alice is the user requesting service.

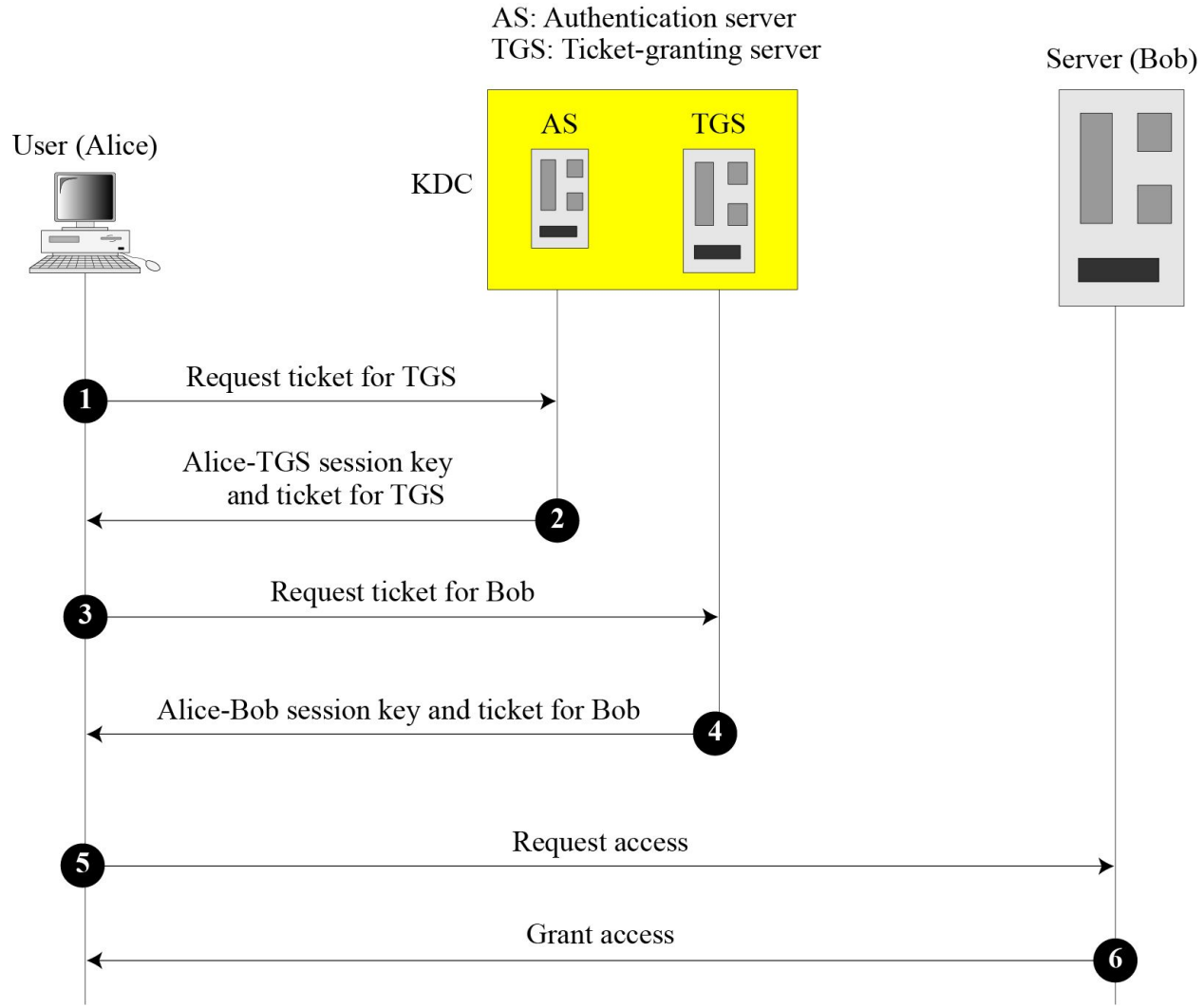
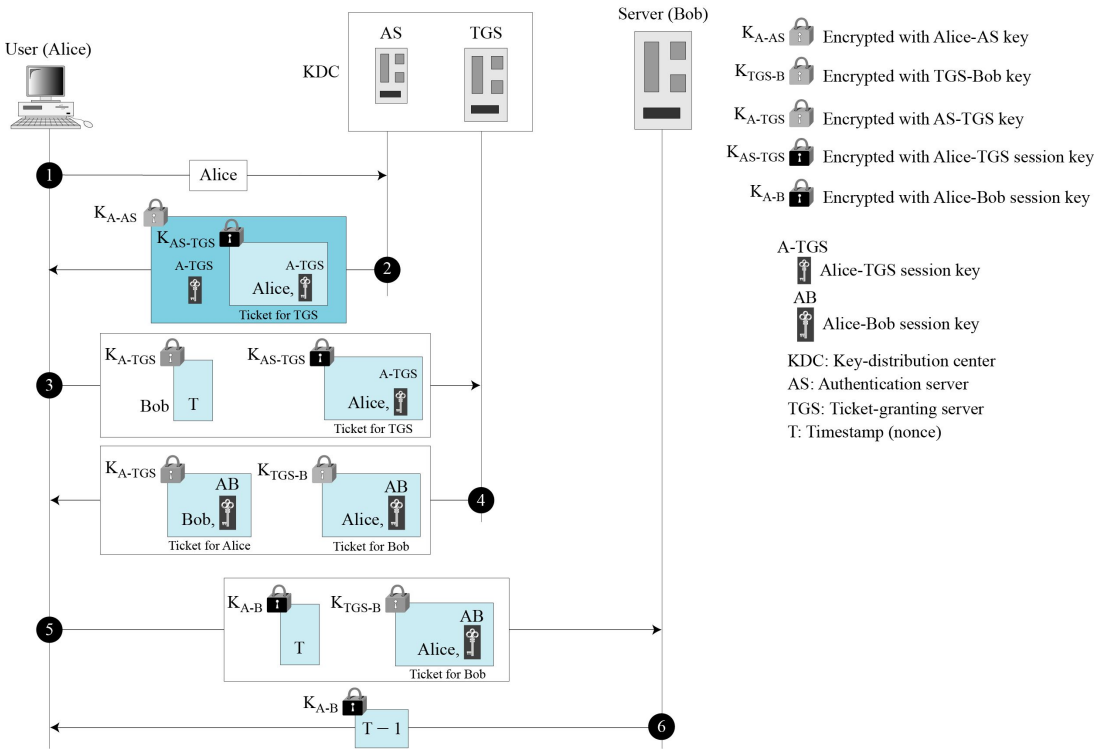


Figure: Kerberos servers

Operation of Kerberos

- In Kerberos, a client process (Alice) can access a process running on the real server (Bob) in six steps, which are summarized below:

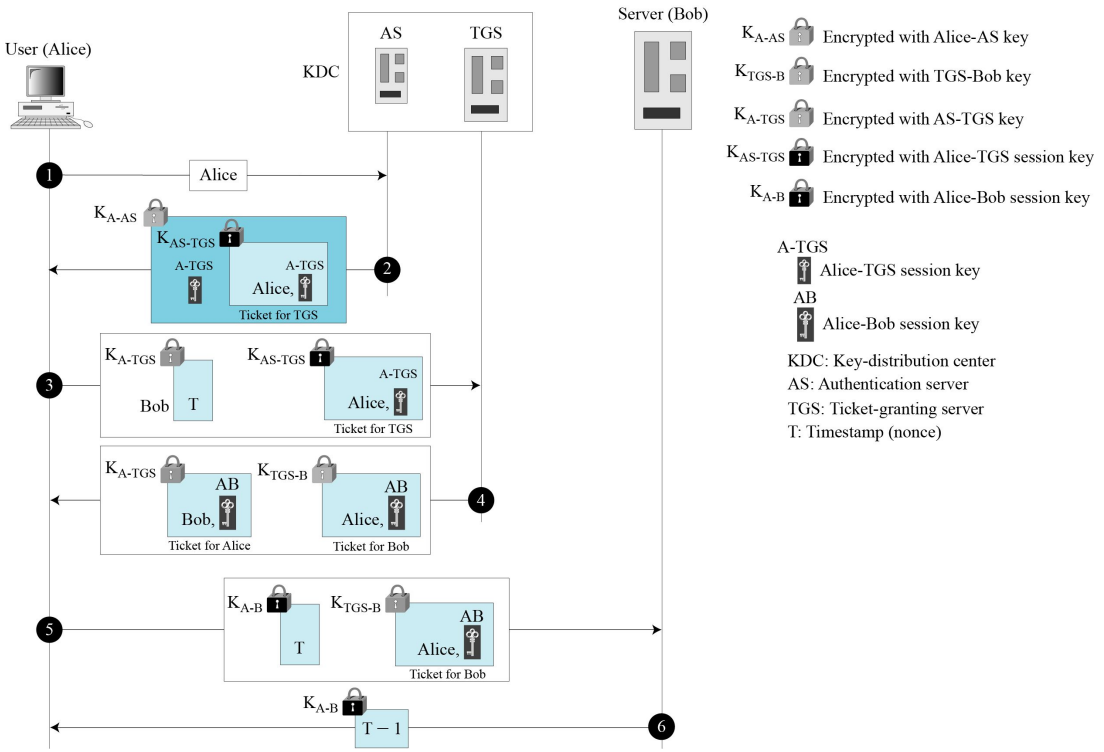
1. Alice sends her request to the AS in plain text using her registered identity which is partially encrypted with her secret key $K_{A\text{-}AS}$.
2. The AS sends a message encrypted with Alice's permanent symmetric key, $K_{A\text{-}AS}$.
   - The message contains two items:
     - a session key $K_{A\text{-}TGS}$, that is used by Alice to contact the TGS.
     - a ticket for TGS that is encrypted with the TGS symmetric key $K_{AS\text{-}TGS}$.
   - Alice does not know $K_{A\text{-}AS}$, but when the message arrives, she types her symmetric password. The password and the appropriate algorithm together create $K_{A\text{-}AS}$ if the password is correct.
   - The process now uses $K_{A\text{-}AS}$ to decrypt the message sent. $K_{A\text{-}TGS}$ and the ticket are extracted.



Operation of Kerberos

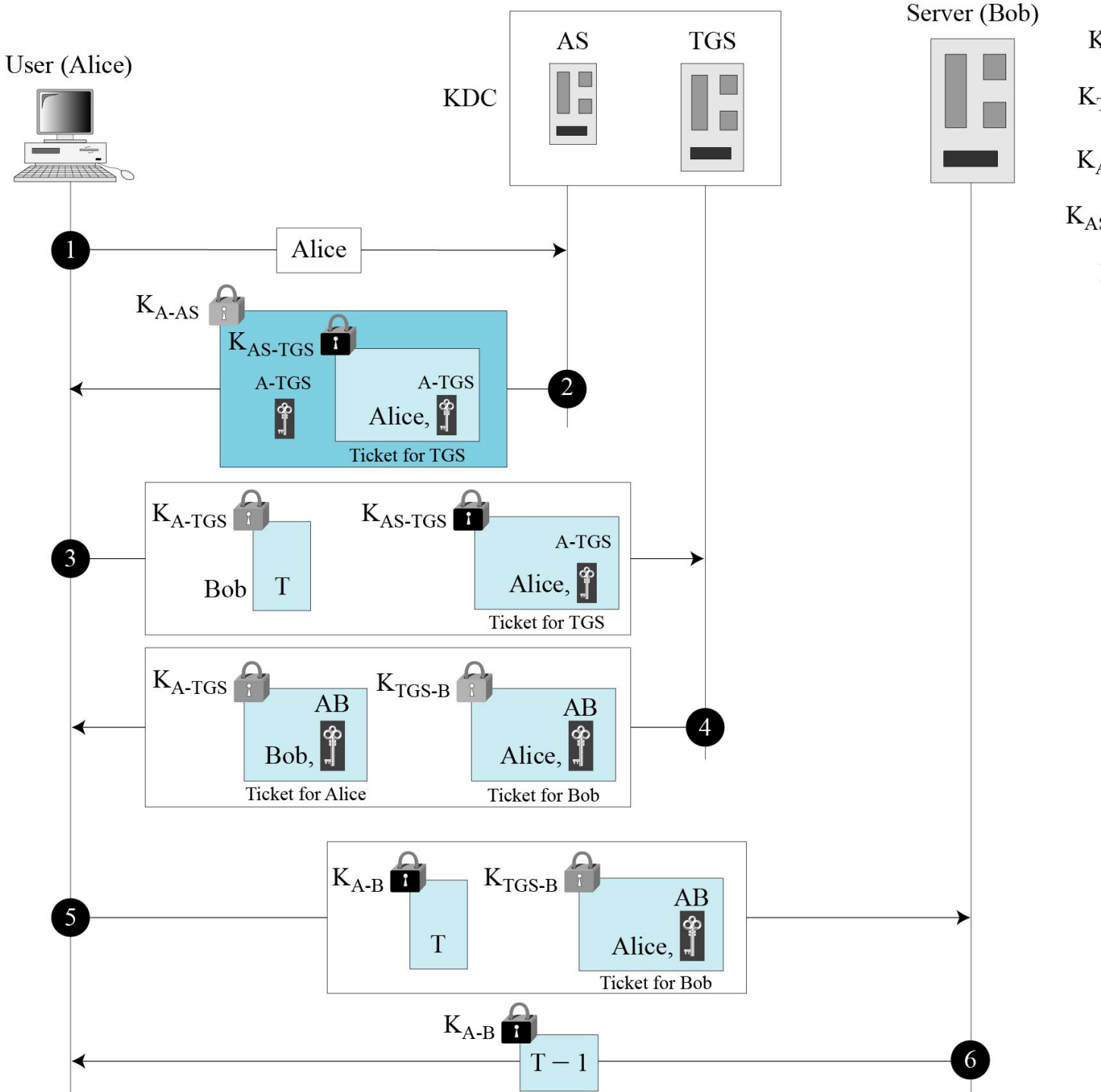
3. Alice now sends three items to the TGS:
   - The first is the ticket received from the AS.
   - The second is the name of the real server (Bob).
   - The third is a timestamp that is encrypted by $K_{A\text{-}TGS}$. The timestamp prevents a replay by Eve.
4. Now, the TGS sends two tickets, each containing the session key $K_{A\text{-}B}$ between Alice and Bob:
   - One ticket for Alice is encrypted with $K_{A\text{-}TGS}$;
   - Another ticket for Bob is encrypted with Bob's key, $K_{TGS\text{-}B}$.

❖ Note that Eve cannot extract $K_{A\text{-}B}$ because she does not know $K_{A\text{-}TGS}$ or $K_{TGS\text{-}B}$. She cannot replay step 3 because she cannot replace the timestamp with a new one (she does not know $K_{A\text{-}TGS}$). Even if she is very quick and sends the step 3 message before the timestamp has expired, she still receives the same two tickets that she cannot decipher.



Operation of Kerberos

- Alice sends Bob's ticket with the timestamp encrypted by K_{A-B}.
- Bob confirms the receipt by adding 1 to the timestamp. The message is encrypted with K_{A-B} and sent to Alice.



Kerberos Version 5:

- After its inception, Kerberos has gone through several versions. Among them, version 4 is the most popular.
- The minor differences between version 4 and version 5 of Kerberos are briefly listed below:
    1) Version 5 has a longer ticket lifetime.
    2) Version 5 allows tickets to be renewed.
    3) Version 5 can accept any symmetric-key algorithm.
    4) Version 5 uses a different protocol for describing data types.
    5) Version 5 has more overhead than version 4.

Symmetric-key Agreement

- Alice and Bob can create a session key between themselves without using a KDC. This method of session-key creation is referred to as the *symmetric-key agreement*.
- Two most common approaches for symmetric-key agreement are:
  1. Diffie-Hellman Key Agreement
  2. Station-to-Station Key Agreement

Symmetric-key Agreement: Diffie-Hellman Key Agreement

- Suppose Alice and Bob have no keys (shared or public), and want to come up with a joint key which they would use for private key cryptography. The Diffie-Hellman (DH) secret key exchange (SKE) protocol enables them to securely exchange a key without the need of a KDC that can then be used for subsequent encryption of messages.
- In Diffie-Hellman key exchange algorithm, there are two publicly known numbers:

1. a large prime number q.
2. an integer α called generator where $\alpha<q$ and α is a *primitive root* of q.

- The integer α termed as generator is called a primitive root of the prime number q if the powers of α modulo q generate all the integers from 1 to q-1.
- That is, if the numbers $\alpha \bmod q$, $\alpha^2 \bmod q$, $\alpha^3 \bmod q$, …., $\alpha^{q-1} \bmod q$ are distinct and consist of the integers ranging from 1 through q-1 in some permutation.

Symmetric-key Agreement: Diffie-Hellman Key Agreement

- Suppose two users Alice and Bob wish to exchange a secret key.
- The Diffie-Hellman key exchange method used in this case is shown in the figure below:

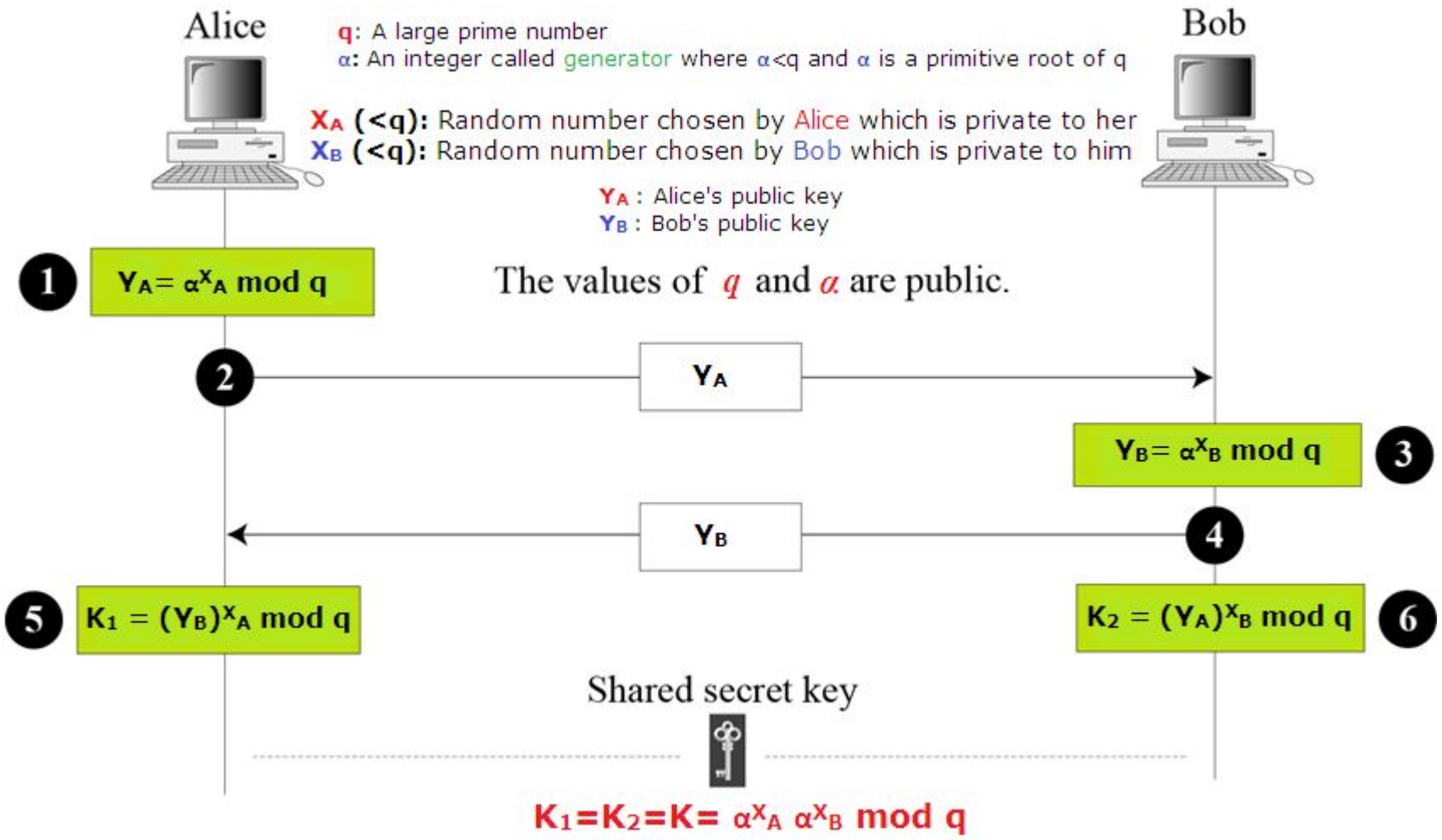
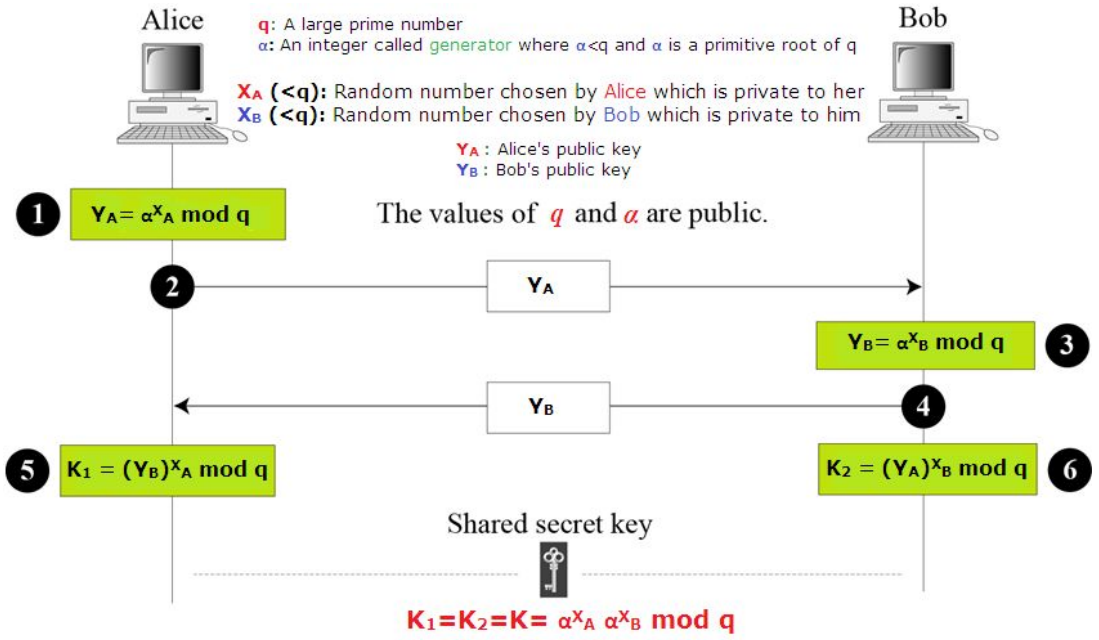


Figure: Diffie-Hellman method

Diffie-Hellman Key Agreement

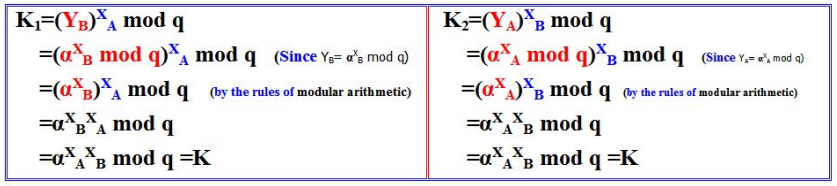
- The steps of the Diffie-Hellman key exchange algorithm for this case is summarized below:

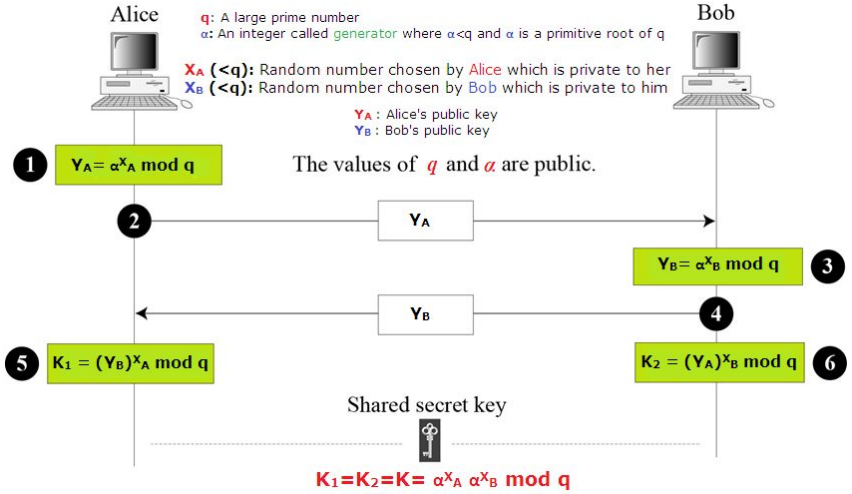
1. Alice selects a random integer X_A which is private to her where $X_A<q$. She then computes her public key $Y_A = \alpha^{X_A} mod\, q$.
2. Alice sends her public key Y_A to Bob.
3. Bob independently selects a random integer X_B which is private to him where $X_B<q$. He then computes his public key $Y_B = \alpha^{X_B} mod\, q$.



Diffie-Hellman Key Agreement

- 4. Bob sends his public key Y_B to Alice.
- 5. Using Bob's public key Y_B, Alice computes the secret key as $K_1 = (Y_B)^{X_A} \bmod q$.
- 6. Similarly, using Alice's public key Y_A, Bob computes the secret key as $K_2 = (Y_A)^{X_B} \bmod q$.
- These two calculations produce identical results which is proved below:





Diffie-Hellman Key Agreement

- In other words, the secret key K of both the users are identical and can be calculated from each other's public key and private random number X_A and X_B.
  - Now Alice can send a message encrypted with her own copy of key K. Bob can decrypt the message using his own copy of key K.
  - The result is that the two sides have exchanged a secret value.
  - Furthermore, because X_A and X_B are private, an adversary only has the following ingredients to work with to determine the key, which is nearly impossible for large prime number: q, α, Y_A and Y_B.

Symmetric-key Agreement: Diffie-Hellman Key Agreement

- The Diffie-Hellman key exchange algorithm is summarized in the figure below.

Global Public Elements

q	prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{XA} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{XB} \bmod q$

Calculation of Secret Key by User A

$K = (Y_B)^{XA} \bmod q$

Calculation of Secret Key by User B

$K = (Y_A)^{XB} \bmod q$

Figure: Diffie-Hellman key exchange algorithm

Note:

- The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.

Symmetric-key Agreement: Diffie-Hellman Key Agreement

Example:

- Here is a very trivial example to clearly show the idea behind Diffie-Hellman protocol. This example uses small numbers, but in a real situation, the numbers are very large.

- Assume that q = 23 and $\alpha = 7$. The steps are as follows:

1. Alice chooses $X_A = 3$ which is private to her. She then calculates $Y_A = \alpha^{X_A} \bmod q = 7^3 \bmod 23 = 21$ which is her public key.
2. Bob chooses $X_B = 6$ which is private to him. He then and calculates $Y_B = \alpha^{X_B} \bmod q = 7^6 \bmod 23 = 4$ which is his public key.
3. Alice sends her public key 21 to Bob.
4. Bob sends his public key 4 to Alice.
5. Using Bob's public key $Y_B=4$, Alice computes the secret key as $K = (Y_B)^{X_A} \bmod q = 4^3 \bmod 23 = 18$.
6. Using Alice's public key $Y_A=21$, Bob computes the secret key as $K = (Y_A)^{X_B} \bmod q = 21^3 \bmod 23 = 18$.
7. The value of K is the same for both Alice and Bob which is the symmetric shared key in the Diffie-Hellman protocol.

Symmetric-key Agreement: Diffie-Hellman Key Agreement

Illustration of Diffie-Hellman Key Exchange Protocols:

- Figure below shows a simple protocol that makes use of the Diffie-Hellman calculation.
- Suppose that user A wishes to set up a connection with user B and use a secret key to encrypt messages on that connection.
- User A can generate a one-time private key X_A, calculate Y_A, and send that to user B.
- User B responds by generating a private value X_B, calculating Y_B, and sending Y_B to user A.
- Both users can now calculate the secret key K. The necessary public values q and α would need to be known ahead of time. Alternatively, user A could pick values for q and α and include those in the first message.

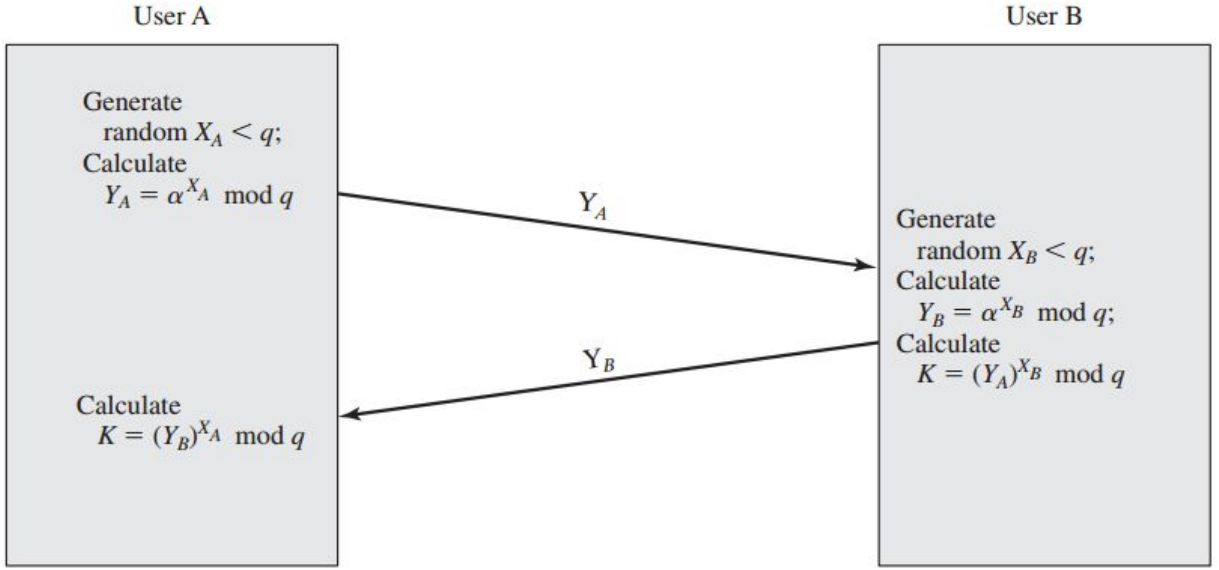


Figure: Illustration of Diffie-Hellman key exchange protocol

Security of Diffie-Hellman Key Exchange

- The Diffie-Hellman key exchange is susceptible to two attacks:

1. Discrete logarithm attack
2. Man-in-the-middle attack

Discrete Logarithm Attack:

- The security of the key exchange is based on the difficulty of the discrete logarithm problem. Eve can intercept Y_A and Y_B. If she can find X_A from $Y_A = \alpha^{X_A} \bmod q$ and X_B from $Y_B = \alpha^{X_B} \bmod q$, then she can calculate the symmetric key K. The secret key is not secret anymore.

- To make Diffie-Hellman safe from the discrete logarithm attack, the following are recommended:

1. The prime q must be very large (more than 300 decimal digits).
2. The prime q must be chosen such that q-1 has at least one large prime factor (more than 60 decimal digits).
3. The generator α must be a primitive root of the prime number q where $\alpha < q$. That is, the numbers $\alpha \bmod q$, $\alpha^2 \bmod q$, $\alpha^3 \bmod q$, …., $\alpha^{q-1} \bmod q$ are distinct and consist of the integers ranging from 1 through q-1 in some permutation.
4. Bob and Alice must destroy X_A and X_B after they have calculated the symmetric key. The values of X_A and X_B must be used only once.

Security of Diffie-Hellman Key Exchange

Man-in-the-Middle Attack:

- The Diffie-Hellman key exchange protocol has another weakness.
- Eve does not have to find the value of X_A and X_B to attack the protocol.
- She can fool Alice and Bob by creating two keys: one between herself and Alice, and another between herself and Bob.
- Figure below shows the situation.

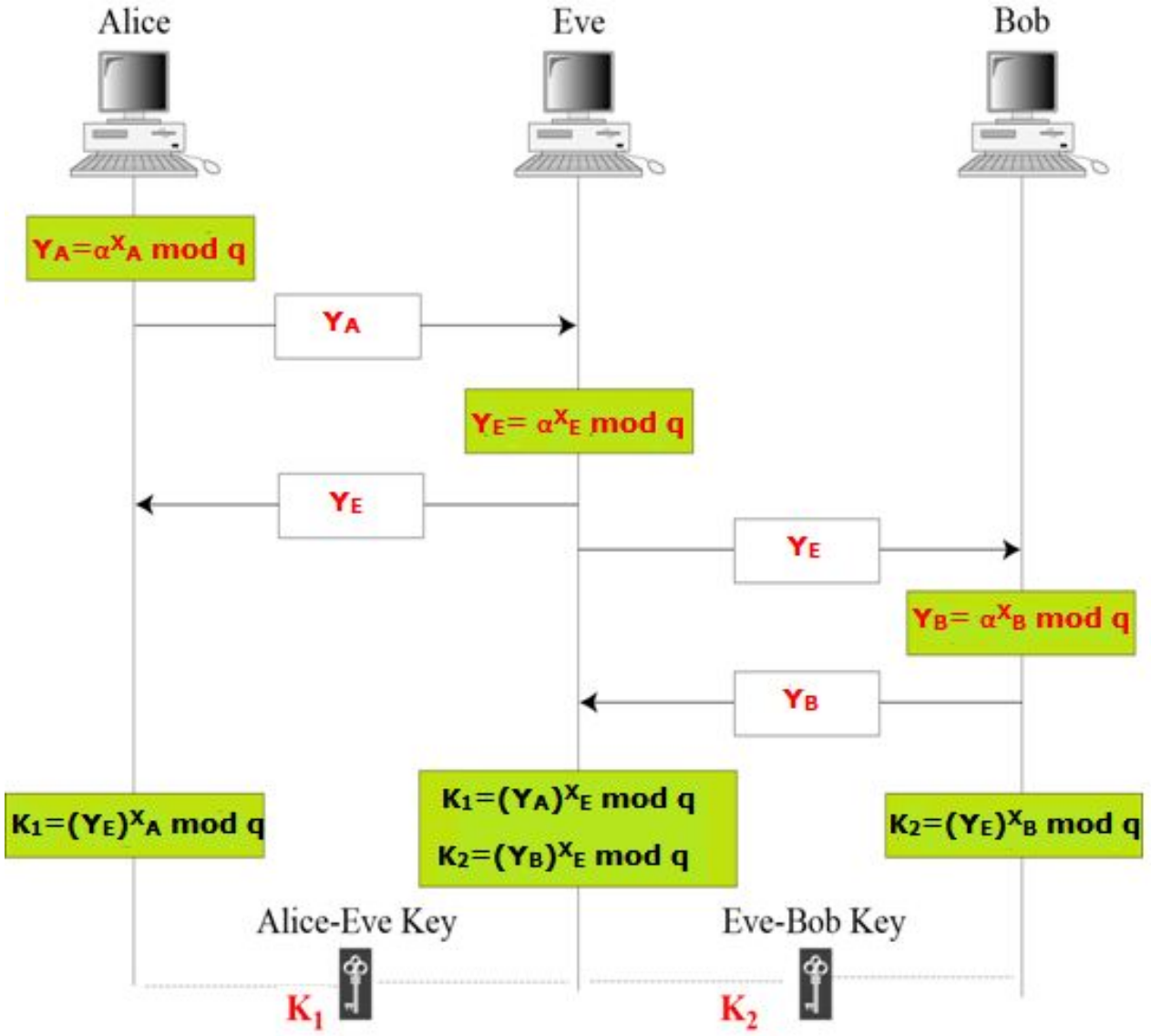
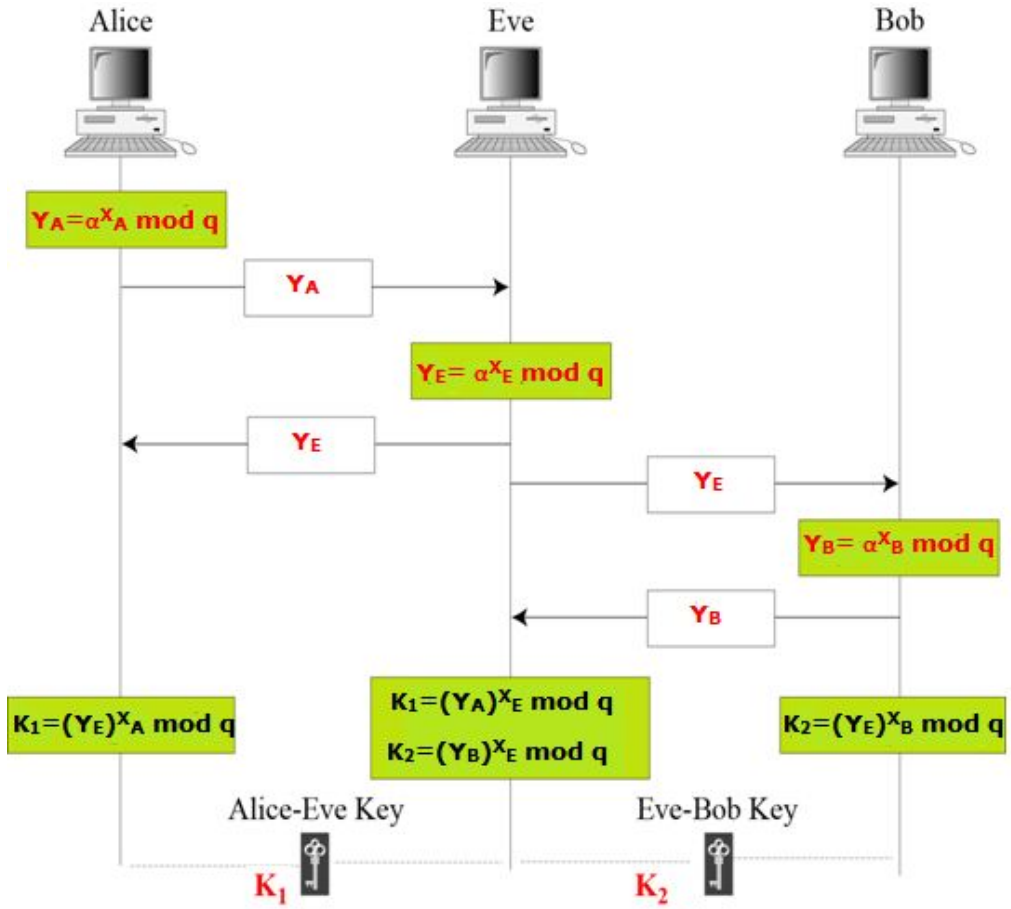


Figure: Man-in-the-middle attack

Security of Diffie-Hellman Key Exchange

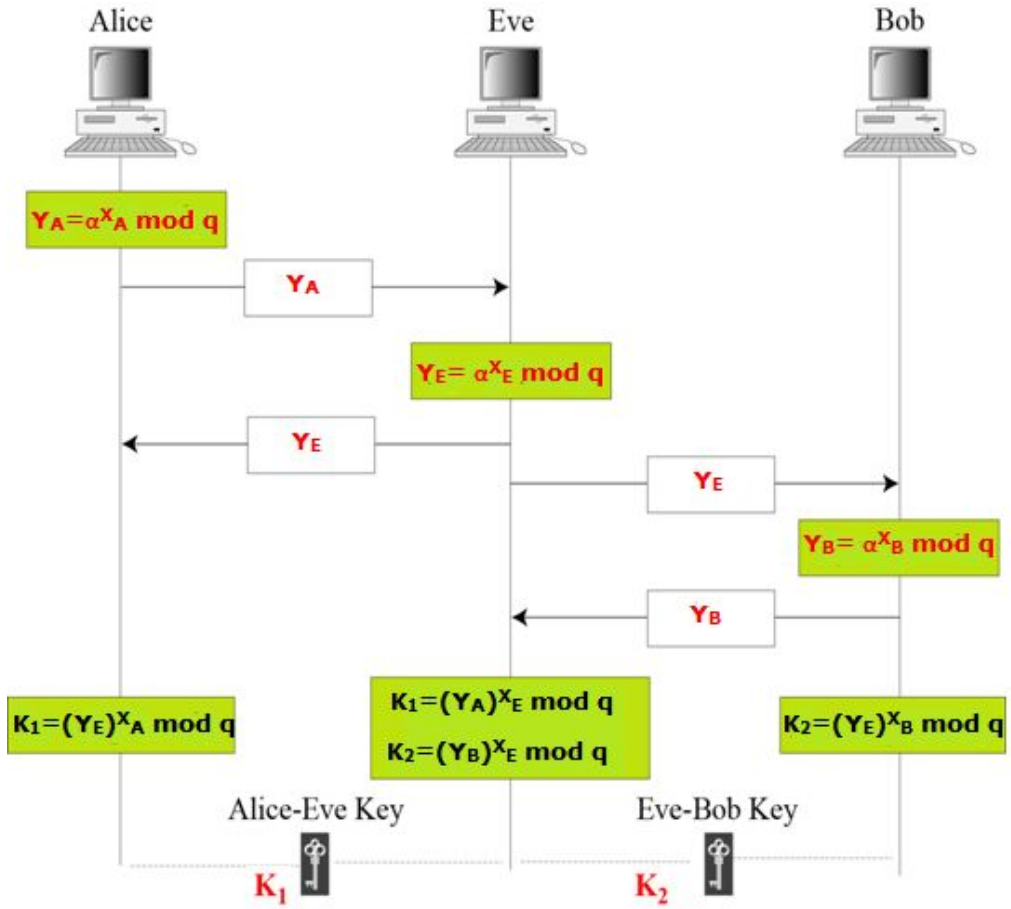
The following can happen:

1. Alice chooses X_A as her private key and calculates $Y_A = \alpha^{X_A} \bmod q$ as her public key, and sends Y_A to Bob.
2. Eve, the intruder, intercepts Y_A. She chooses X_E as her own key, calculates $Y_E = \alpha^{X_E} \bmod q$ as the public key, and sends Y_E to both Alice and Bob.
3. Bob chooses X_B as his private key and calculates $Y_B = \alpha^{X_B} \bmod q$, and sends Y_B to Alice. Y_B is intercepted by Eve and never reaches Alice.
4. Alice and Eve calculate K_1 which becomes a shared key between Alice and Eve. Alice, however, thinks that it is a key shared between Bob and herself.
5. Eve and Bob calculate K_2 which becomes a shared key between Eve and Bob. Bob, however, thinks that it is a key shared between Alice and himself.



Security of Diffie-Hellman Key Exchange

- In other words, two keys, instead of one, are created: one between Alice and Eve, one between Eve and Bob.
- When Alice sends data to Bob encrypted with K_1 (shared by Alice and Eve), it can be deciphered and read by Eve. Eve can send the message to Bob encrypted by K_2 (shared key between Eve and Bob); or she can even change the message or send a totally new message. Bob is fooled into believing that the message has come from Alice. A similar scenario can happen to Alice in the other direction.
- This situation is called a man-in-the-middle attack because Eve comes in between and intercepts Y_A sent by Alice to Bob, and Y_B sent by Bob to Alice.
- It is also known as a bucket brigade attack because it resembles a short line of volunteers passing a bucket of water from person to person



Symmetric-key Agreement: Station-to-Station Key Agreement

- The station-to-station protocol is a method of key agreement which is based on Diffie-Hellman.
- It uses digital signatures with public-key certificates to establish a session key between Alice and Bob, as shown in the figure here.

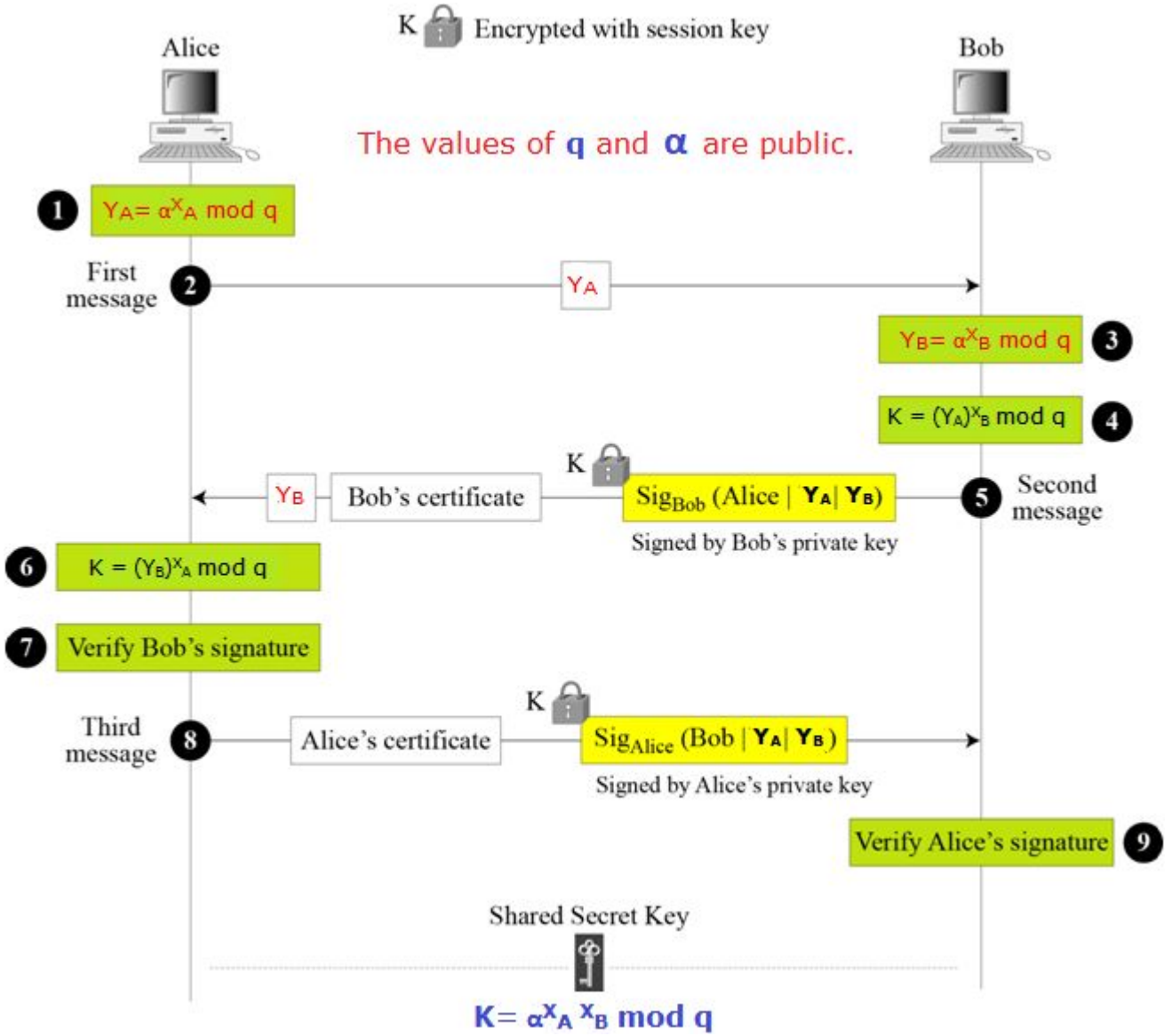
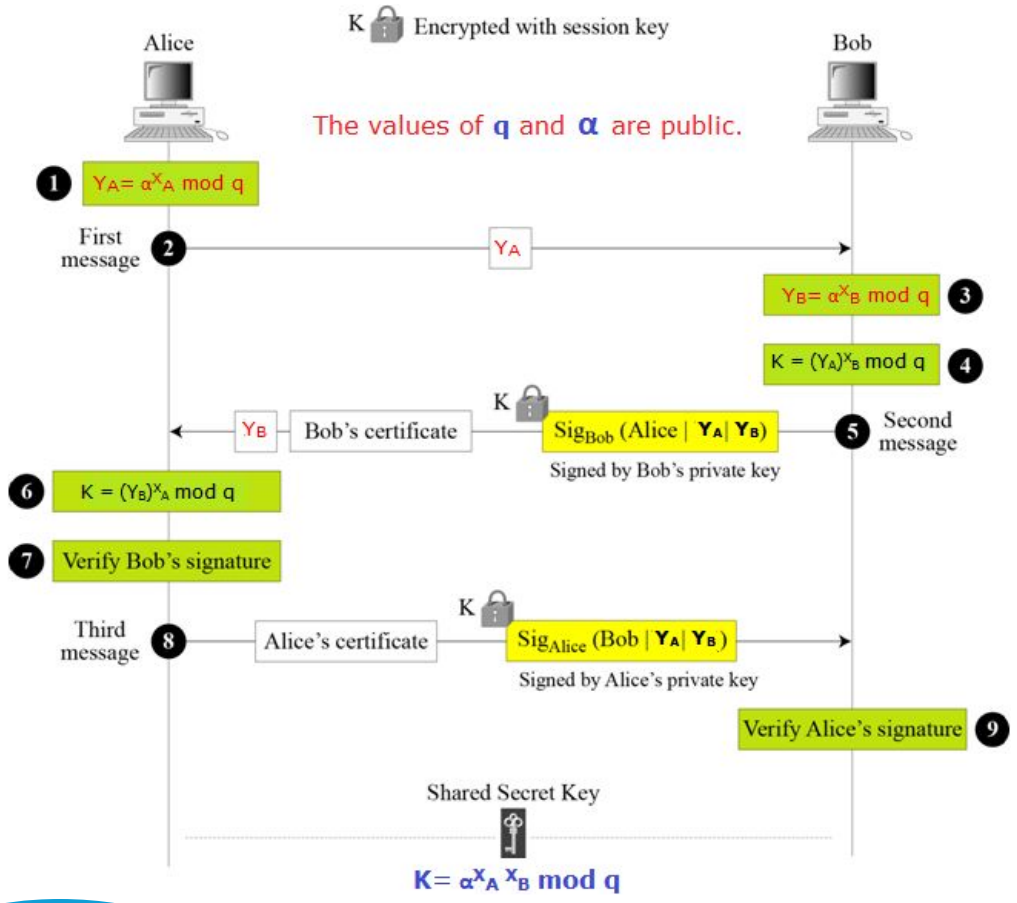


Figure: Station-to-station key agreement method

Symmetric-key Agreement: Station-to-Station Key Agreement



- The steps of the station-to-station key agreement protocol is summarized below:
  - After calculating Y_A, Alice sends it to Bob (steps 1 and 2 in the above figure).
  - After calculating Y_B and the session key, Bob concatenates Alice's ID, Y_A , and Y_B. He then signs the result with his private key.
  - Bob now sends Y_B, the signature, and his own public-key certificate to Alice. The signature is encrypted with the session key (steps 3, 4, and 5 in the above figure).
  - After calculating the session key, if Bob's signature is verified, Alice concatenates Bob's ID, Y_A, and Y_B. She then signs the result with her own private key and sends it to Bob. The signature is encrypted with the session key (steps 6, 7, and 8 in the above figure).
  - If Alice's signature is verified, Bob keeps the session key (step 9 in the figure).

Symmetric-key Agreement: Station-to-Station Key Agreement

Security of Station-to-Station Protocol:

- The station-to-station protocol prevents man-in-the-middle attacks.
  - After intercepting Y_A, Eve cannot send her own Y_B to Alice and pretend it is coming from Bob, because Eve cannot forge the private key of Bob to create the signature—the signature cannot be verified with Bob's public key defined in the certificate.
  - In the same way, Eve cannot forge Alice's private key to sign the third message sent by Alice.
  - The certificates are trusted because they are issued by trusted authorities.

Public Key Distribution

Dr. Risala Tasin Khan

Professor

IIT, JU

Introduction

- In asymmetric-key cryptography, people do not need to know a symmetric shared key; everyone shields a private key and advertises a public key.
  - If Alice wants to send a message to Bob, she only needs to know Bob's public key, which is open to the public and available to everyone.
  - Similarly, if Bob needs to send a message to Alice, he only needs to know Alice's public key, which is also known to everyone.
- In public-key cryptography, everyone shields a private key and advertises a public key.
- Like secret keys, public keys need to be distributed.
- Now we will briefly discuss the way public keys can be distributed.

Ways of Distributing Public Key:

- There are several approaches to distribute a public key:
  1. Public announcement
  2. Trusted center
  3. Certifying authority

Public Announcement:

- The naive approach is to announce public keys publicly.

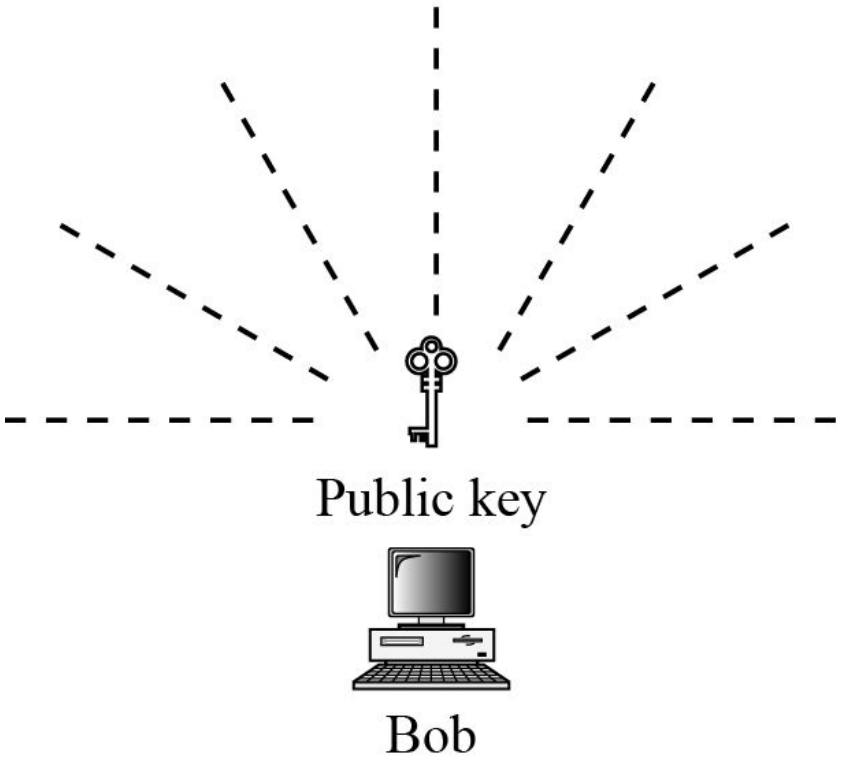


Figure: Announcing a public key

- Bob can put his public key on his website or announce it in a local or national newspaper.
- When Alice needs to send a confidential message to Bob, she can obtain Bob's public key from his site or from the newspaper, or even send a message to ask for it.
- Figure below shows the situation.
- However, this approach is not secure; it is subject to forgery.
  - For example, Eve could make such a public announcement. Before Bob can react, damage could be done. Eve can fool Alice into sending her a message that is intended for Bob.
  - Eve could also sign a document with a corresponding forged private key and make everyone believe it was signed by Bob.
  - The approach is also vulnerable if Alice directly requests Bob's public key. Eve can intercept Bob's response and substitute her own forged public key for Bob's public key.

Ways of Distributing Public Key:

Trusted Center:

- A more secure approach is to have a trusted center retain a directory of public keys.
  - The directory (like the one used in a telephone system) is dynamically updated.
  - Each user can select a private and public key, keep the private key, and deliver the public key for insertion into the directory.
  - The center requires that each user register in the center and prove his or her identity. The directory can be publicly advertised by the trusted center.
  - The center can also respond to any inquiry about a public key.
  - Figure shows the concept.

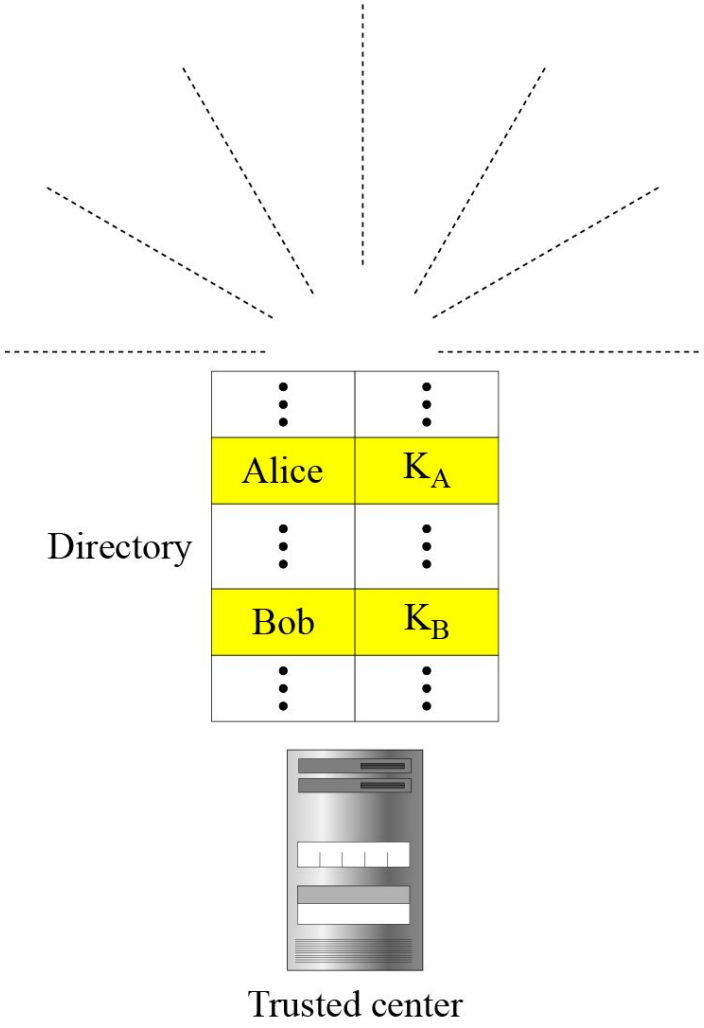
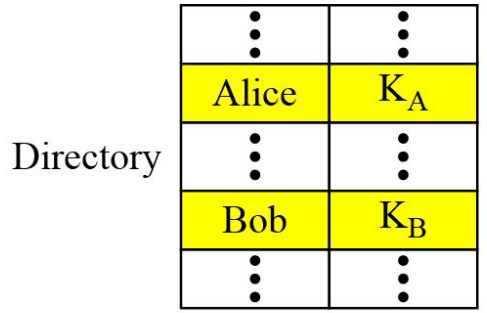


Figure: Trusted center

Ways of Distributing Public Key:

Controlled Trusted Center:

- A higher level of security can be achieved if there are added controls on the distribution of the public key.
- The public-key announcements can include a timestamp and be signed by an authority to prevent interception and modification of the response.
  - If Alice needs to know Bob's public key, she can send a request to the center including Bob's name and a timestamp.
  - The center responds with Bob's public key, the original request, and the timestamp signed with the private key of the center.
  - Alice uses the public key of the center, known by all, to verify the timestamp. If the timestamp is verified, she extracts Bob's public key.
  - Figure shows one scenario.



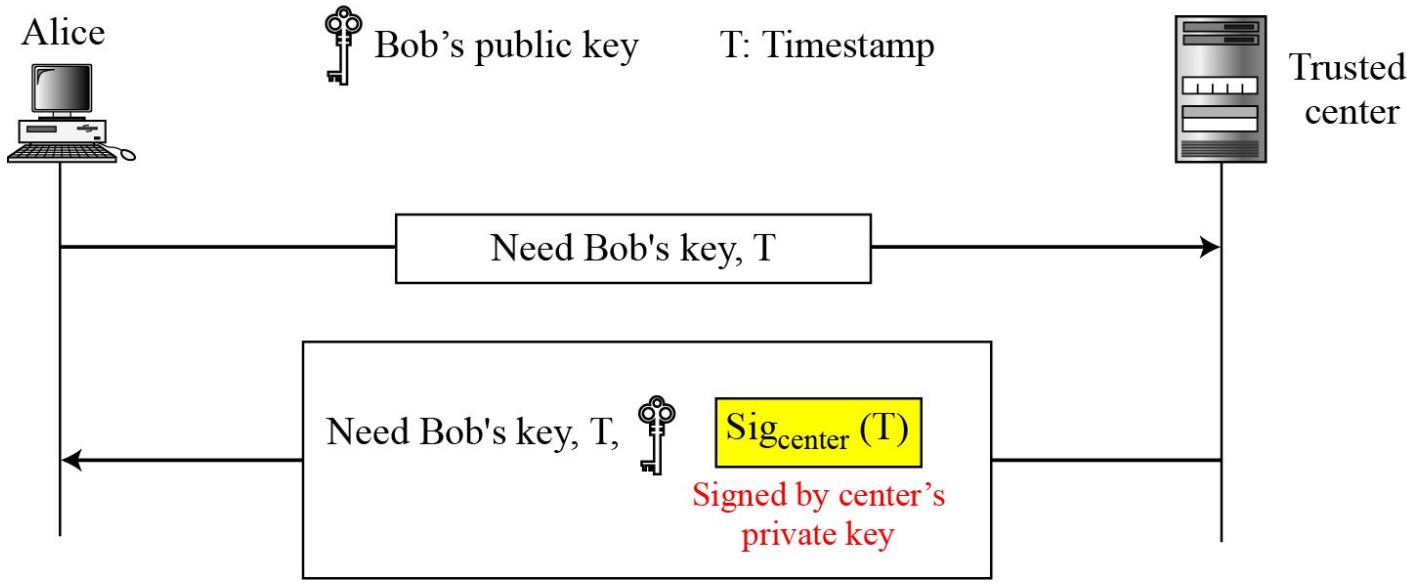


Figure: Controlled trusted center

Ways of Distributing Public Key:

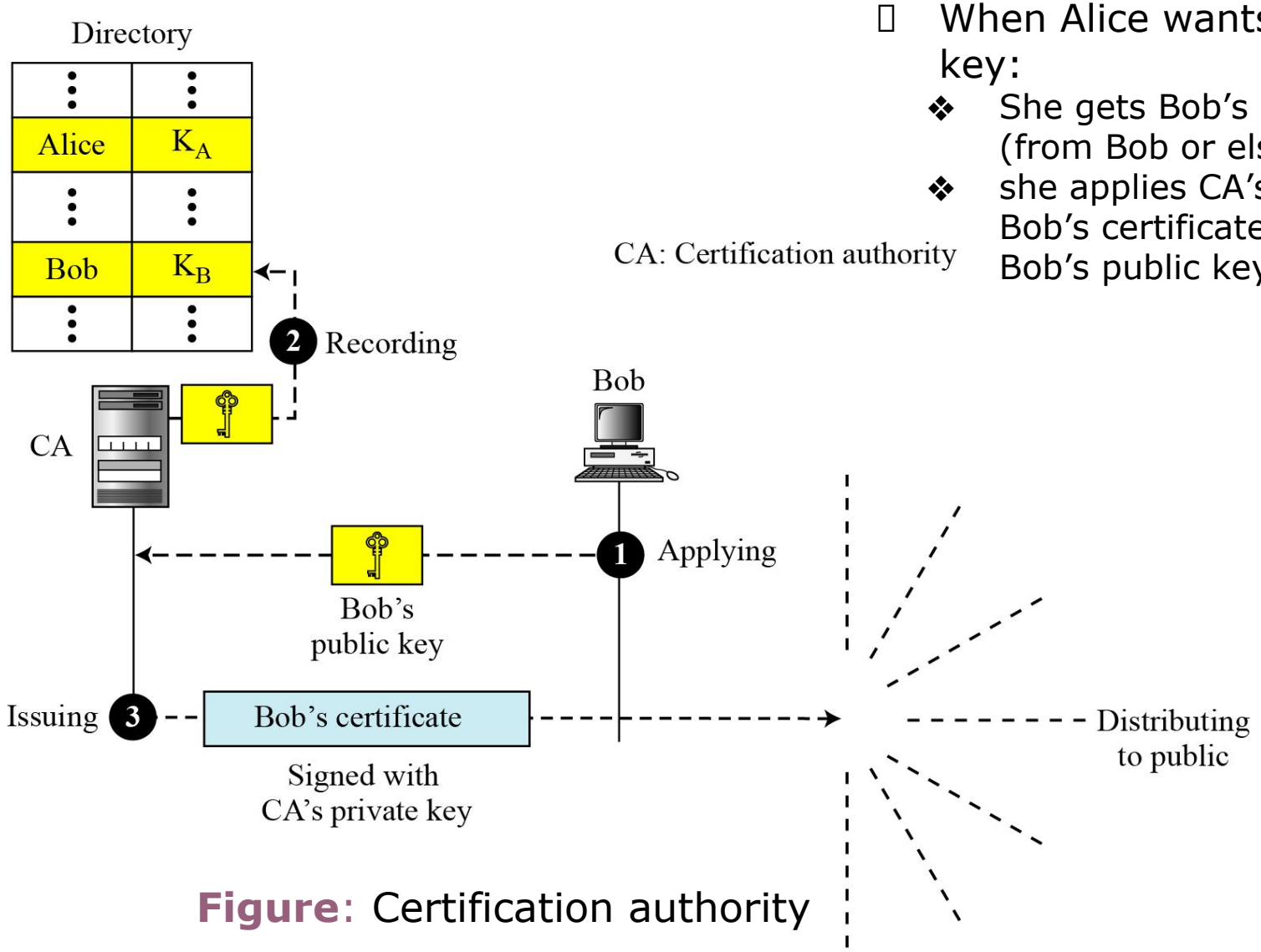
Certification Authority (CA):

- The distribution of public key through controlled trusted center (discussed before) can create a heavy load on the center if the number of requests is large. The alternative is to create public-key certificates.
- Suppose, Bob wants two things:
  1. He wants people to know his public key.
  2. He wants no one to accept a forged public key as his.
     - How can he do this?
- The steps he may follow are:
  - Bob can go to a certification authority (CA), a federal or state organization that binds a public key to an entity and issues a certificate. The CA has a well-known public key itself that cannot be forged.
  - The CA checks Bob's identification (using a picture ID along with other proof).
  - It then asks for Bob's public key and writes it on the certificate it will issue for Bob.
    - To prevent the certificate itself from being forged, the CA signs the certificate with its private key.
  - Now Bob can upload the signed certificate.
  - Anyone who wants Bob's public key downloads the signed certificate and uses the CA's public key to extract Bob's public key.

Ways of Distributing Public Key:

Certification Authority (Cont…):

- The steps stated before are illustrated in the figure below.



- When Alice wants Bob's public key:
  - She gets Bob's certificate (from Bob or elsewhere).
  - she applies CA's public key to Bob's certificate, and gets Bob's public key

Figure: Certification authority

Digital Certificate

- A problem in public-key systems is the authenticity of the public key.
  - An attacker may offer the sender her own public key and pretend that it origins from the legitimate receiver.
  - The sender then uses the fake public key to perform her encryption and the attacker can simply decrypt the message using her private key.
  - This technique may be used to set up a man-in-the-middle attack in which a third party is able to monitor and modify the communication between two parties, even when encryption is used.
- In order to thwart an attacker that attempts to substitute her public key for the victim's one, digital certificates are used.
  - A certificate combines user information with the user's public key and the digital signature of a trusted third party that guarantees that the key belongs to the mentioned person.
  - The trusted third party is usually called a certification authority (CA).
  - A digital certificate is just a file or a software program, digitally signed by a signing authority, that can be installed in a browser. Once installed, the digital certificate identifies the user of that browser to websites equipped to check it automatically. It is like an electronic "credit card" that establishes one's credentials when doing business on the Web.

- Therefore, a digital certificate, issued by a certifying authority, is an electronic attachment to an electronic message that is used to verify that a user sending a message is who they claim to be.
- Those wishing to send encrypted messages obtain a digital certificate from a certifying authority.
  - The certifying authority issues an encrypted digital certificate.
- The recipient of an encrypted message uses the certifying authority's public key to decode the digital certificate attached to the message.
  - The recipient verifies it as issued by the certifying authority.
  - Then it obtains the sender's public key and identification information held within the digital certificate.
  - With this information, the recipient can then send an encrypted reply.
- The most widely used standard for digital certificates is X.509. Hence, digital certificates are sometimes called X.509 certificates.

Components of a Digital Certificate

- A typical digital certificate contains several key elements:

1. **Certificate Holder's Information**:
   - This includes the name, email address, organization, and other identifying details of the certificate holder.
2. **Public Key**:
   - The public key of the certificate holder, which others can use to encrypt data sent to them or to verify digital signatures made by the holder.
3. **Digital Signature**:
   - The certificate is signed by a trusted **Certificate Authority (CA)**, which certifies the authenticity of the certificate.
4. **Validity Period**:
   - Specifies the certificate's start and expiration dates.
5. **Certificate Serial Number**:
   - A unique number assigned to the certificate, allowing it to be identified and tracked.

How Digital Certificate works

1. **Requesting a Certificate**:
   - The certificate holder (individual or organization) generates a public-private key pair and sends a **Certificate Signing Request (CSR)** to a trusted Certificate Authority (CA) such as VeriSign, DigiCert, or Let's Encrypt.
2. **Issuance by Certificate Authority (CA)**:
   - The CA verifies the identity of the requester through various checks.
   - Once verified, the CA issues a digital certificate, binding the requester's identity to their public key and signing the certificate with the CA's own private key.
3. **Using the Digital Certificate**:
   - When someone wants to establish a secure connection (e.g., accessing a website or sending encrypted data), the server or individual presents their digital certificate.
   - The recipient uses the CA's public key to verify the digital signature on the certificate. This assures them that the certificate is genuine and hasn't been tampered with.
   - The recipient can then use the certificate's public key to establish a secure, encrypted communication channel or verify a digital signature.

Digital Certificate

For what purposes you can use the digital certificates?

- You can use the digital certificate to digitally sign email, documents, files etc. to prove you were the author, and that they have not been tampered with.
- You can also use some types of certificate as digital ID. Others can electronically challenge you to prove you know the private key that fits with the public key in the certificate by encrypting a message they provide.
  - The problem with that is, all the information in the certificate is revealed to whoever you show it to.
  - If you want to selectively reveal information, you need several certificates.
    - You might want one with just your birth date for entry to porn sites, but no other information. You might want one that revealed only a very minimal amount of information when dealing with on-line vendors to avoid being bombarded with junk electronic and snail mail.
- Digital certificates can also be used instead of passwords to verify who you are to some site.
  - The site challenges you by sending you a message that you digitally sign and send back. If some spy had snooped on you logging in before, it would not help him to spoof you, the way it would had you used a password.
  - Thus, a digital certificate eliminates remembering multiple passwords and enhances security, because it can not be guessed, forgotten, forged, or intercepted.

Digital Certificate

For what purposes you can use the digital certificates (cont…)?

- Other types of certificate allow you to encrypt and sign all HTML traffic leaving your web server, thus proving it came from you and providing privacy.
  - Recipients can determine whether data did indeed come from you by checking the digital signature.
  - To verify, all they need is a master certificate from the signing authority, which comes built into their browser or email software. They don't need to check up your key in an on-line database unless they want to check to see if the certificate has been revoked.
- In many ways, digital certificates are the heart of secure online transactions.
  - In shopping on the Internet, buyers need evidence that they can trust the vendor. Digital certificate establishes a merchant's identity and thus ensures secure e-commerce transaction.
  - MasterCard and Visa have designed the SET certificate that can be used for secure financial transactions over the web. VeriSign supplies the certificates.

Digital Certificate

Different Classes of Digital Certificate:

- A digital certificate can be issued (for a fee) in one of FOUR classes:

1. **Class 1 Certificate:**
   - Certificates of this class are the quickest and simplest to issue because they contain minimum checks on the user's background. Only the name, address and e-mail address of the user are checked. Think of it as a library card.
2. **Class 2 Certificate:**
   - Certificates of this class check for information like real name, SSN (social security number), and date of birth of the user. They require proof of physical address, locale, and e-mail address as well. This is more like a credit card, because the company giving out the certificate will consult with a credit database for verification with a third party.
3. **Class 3 Certificate:**
   - Certificates of this class are the strongest type in terms of specifics. They are like a driver's license: To get them, you need to prove exactly who you are and that you are responsible. Organizations whose specialty is the security business foresee class 3 certificates being used for things like loans acquired online and other sensitive transactions.
4. **Class 4 Certificate:**
   - Certificates of this class are the most thorough. In addition to class 3 requirements, the certificate authority checks on things like the user's position at work.

X.509 Digital Certificate

- Although the use of a CA has solved the problem of public-key fraud, it has created a side-effect.
  - Each certificate may have a different format.
  - If Alice wants to use a program to automatically download different certificates and digests belonging to different people, the program may not be able to do this.
    - One certificate may have the public key in one format and another in a different format.
    - The public key may be on the first line in one certificate, and on the third line in another.
  - Anything that needs to be used universally must have a universal format.
- To remove this side effect, the ITU has designed X.509, a recommendation that been accepted by the Internet with some changes.
- X.509 is a way to describe the certificate in a structured way. It uses a well-known protocol called ASN. 1 (Abstract Syntax Notation 1) that defines fields familiar to C programmers.

Common Components of X.509 Certificate

An X.509 certificate typically contains the following elements:

1. **Version**: Identifies the version of the X.509 standard being used (usually version 3 in modern systems).
2. **Serial Number**: A unique identifier assigned by the Certificate Authority (CA) to distinguish this certificate from others.
3. **Signature Algorithm**: Specifies the cryptographic algorithm (e.g., SHA-256) that the CA used to sign the certificate.
4. **Issuer**: Details about the Certificate Authority that issued the certificate, including its name and potentially other identifying information.
5. **Validity Period**: Specifies the start and end dates for which the certificate is valid. After the expiration date, the certificate is no longer trusted.
6. **Subject**: Information about the entity the certificate is issued to (e.g., a user, a website, or an organization). This can include details such as the organization name, domain name, and location.
7. .

1. **Subject Public Key Info**: Contains the subject's public key and the algorithm used to generate it. This key is used for secure communications with the subject.
2. **Extensions (Version 3 only)**: Optional fields that provide additional information about the certificate’s capabilities.
3. **Signature:**

   This field is made of three sections-

   - **The first section contains all other fields in the certificate.**
   - **The second section contains the digest of the first section encrypted with the CA’s public key.**
   - **The third section contains the algorithm identifier used to create the second section**

X.509 Certificate

Format of X.509 Certificate:

- Figure below shows the format of X.509 certificate.

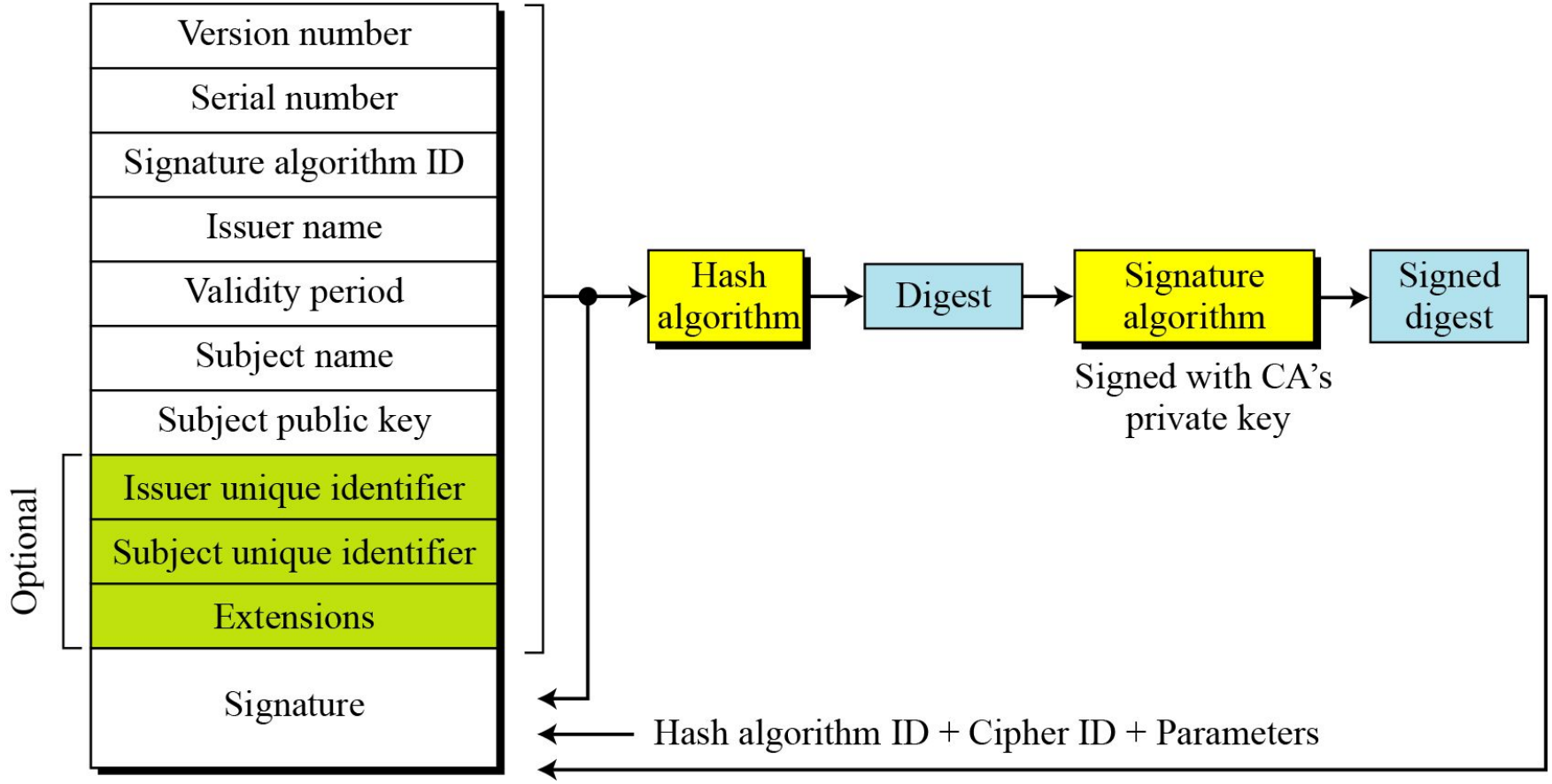


Figure: X.509 certificate format

X.509 Certificate

Certificate Renewal:

- Each certificate has a period of validity.
- If there is no problem with the certificate, the CA issues a new certificate before the old one expires.
  - The process is like the renewal of credit cards by a credit card company; the credit card holder normally receives a renewed credit card before the one expires.

X.509 Certificate

Certificate Revocation:

- In some cases a certificate must be revoked before its expiration.
- Here are some examples:
  a) The user's (subject's) private key (corresponding to the public key listed in the certificate) might have been comprised.
  b) The CA is no longer willing to certify the user. For example, the user's certificate relates to an organization that she no longer works for.
  c) The CA's private key, which can verify certificates, may have been compromised. In this case, the CA needs to revoke all unexpired certificates.
- The revocation is done by periodically issuing a certificate revocation list (CRL).
  - The list contains all revoked certificates that are not expired on the date the CRL is issued.
  - When a user wants to use a certificate, she first needs to check the directory of the corresponding CA for the last certificate revocation list.

X.509 Certificate

Certificate Revocation Format (cont…):

- A certificate revocation list has the following fields:
  - **Signature algorithm ID:**
    This field is the same as the one in the certificate.
  - **Issuer name:**
    This field is the same as the one in the certificate.
  - **This update date:**
    This field defines when the list is released.
  - **Next update date:**
    This field defines the next date when the new list will be released.
  - **Revoked certificate.**
    This is a repeated list of all unexpired certificates that ha been revoked. Each list contains two sections: user certificate serial number and revocation date.
  - **Signature.**
    This field is the same as the one in the certificate list.

Certificate Authority

Selecting a Certificate Vendor:

- Some criteria to consider when buying your certificate are:
  - Cost, both initial and renewal.
  - Does the company provide all the different kinds of certificate you will need. It is much less hassle to get everything from one source.
  - Are the root certificates (*root certificates are typically pre-installed at the factory in browsers*) of that vendor built into the browsers your clients will be using? If not, it will be a hassle for your users to manually install them.
  - What sort of reputation does the vendor have for service? Basically you are paying them to verify that you are you. You want them to do that thoroughly without driving you crazy.

Public-key Infrastructure (PKI)

- Public-Key Infrastructure (PKI) is a model for creating, distributing, and revoking certificates based on the X.509.
- The Internet Engineering Task Force has created the Public-Key Infrastructure X.509 (PKIX).

Duties of PKI:

- Several duties have been defined for a PKI. The most important ones are shown in the figure below:

- Certificates' issuing, renewal, and revocation: These are duties defined in the X.509. Because the PKIX is based on X.509, it needs to handle all duties related to certificates.
- Keys' storage and update: A PKI should be a storage place for private keys of those members that need to hold their private keys somewhere safe. In addition, a PKI is responsible for updating these keys on members' demands.

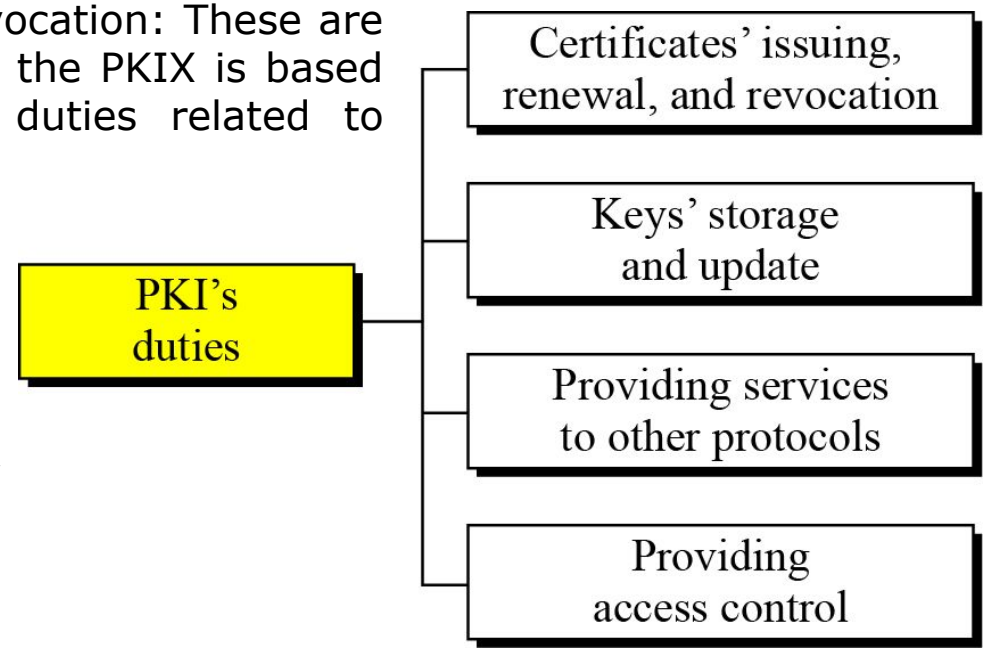


Figure: Some duties of a PKI

Public-key Infrastructure (PKI)

Duties of PKI (cont…):

- Providing services to other protocols: Some Internet security protocols, such as IPSec and TLS, are relying on the services by a PKI.
- Providing access control: A PKI can provide different levels of access to the information stored in its database. For example, an organization PKI may provide access to the whole database for the top management, but limited access for employees.

Public Key Infrastructure (PKI)

Trust Model:

- It is not possible to have just one CA issuing all certificates for all users in the world.
- There should be many CAs, each responsible for creating, storing, issuing, and revoking a limited number of certificates.
- The trust model defines rules that specify how a user can verify a certificate received from a CA.

Hierarchical Model:

- In this model, there is a tree-type structure with a root CA.
- The root CA has a self-signed, self-issued certificate; it needs to be trusted by other CAs and users for the system to work.

Public Key Infrastructure (PKI)

Hierarchical Model (cont..):

- Figure below shows a trust model of this kind with three hierarchical levels. The number of levels can be more than three in a real situation.

- The figure shows that the CA (the root) has signed certificates for CA1, CA2, and CA3; CA 1 has signed certificates for User1, User2, and User3; and so on. PKI uses X<<Y>> as the notation to mean the certificate issued by authority X for entity Y.

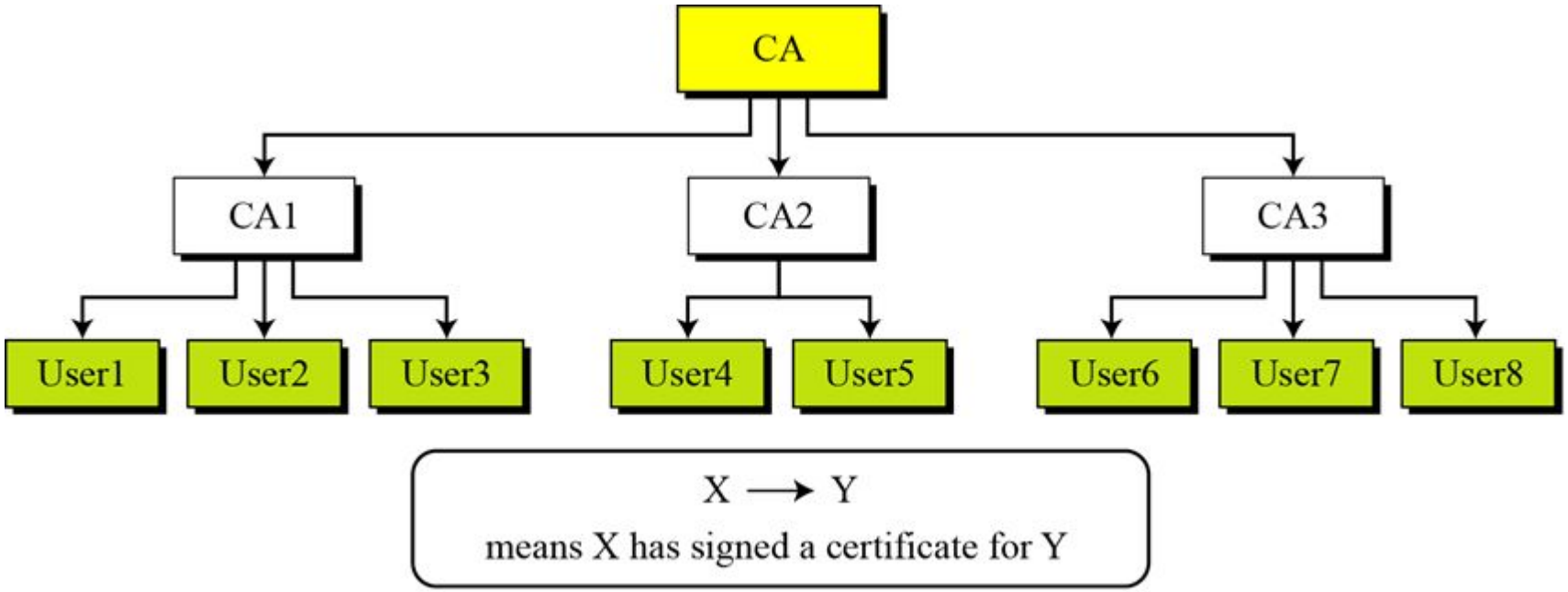


Figure: PKI hierarchical model

Public Key Infrastructure (PKI)

Example-1:

Show how User1, knowing only the public key of the CA (the root), can obtain a verified copy of User3's public key.

Solution:

User3 sends a chain of certificates, CA<<CA1>> and CA1<<User3>>, to User1.

a) User1 validates CA<<CA1>> using the public key of CA.

b) User1 extracts the public key of CA1 from CA<<CA1>>.

c) User1 validates CA1<<User3>> using the public key of CA1.

d) User1 extracts the public key of User3 from CA1<<User3>>.

DES Algorithm

Dr. Risala Tasin Khan

Professor

IIT, JU

Modern Symmetric-key Ciphers:

Character-oriented Vs. Bit-oriented Ciphers:

- The traditional symmetric-key ciphers are character-oriented ciphers.
- Now-a-days, the information to be encrypted is not just text; it can also consist of numbers, graphics, audio, and video data. It is convenient to convert these types of data into a stream of bits, to encrypt the stream, and then to send the encrypted stream.
- So, we need bit-oriented ciphers.
  - When data is treated as the collection of bits, it becomes larger. Mixing a larger number of symbols increases security.

Kinds of Modern Symmetric-key Ciphers:

Modern symmetric-key ciphers can be divided into two broad categories:

1. Stream ciphers:
   Stream cipher encrypts a single character or bit of plaintext at a time. It also decrypts a single character or bit of ciphertext at a time.
2. Block ciphers:
   A symmetric-key modern block cipher encrypts an n-bit block of plaintext or decrypts an n-bit block of ciphertext together.

Stream Ciphers (continued…)

Example:

- Given plaintext: 1001101111010000 1
  Let the keystream be a stream of 1s and 0s.
- If we use an exclusive or (XOR) with the keystream and plaintext, we get ciphertext.
- This keystream is called periodic, since the sequence '10' repeats over and over.

Plaintext : 1001101111010000 1
Keystream : 1010101010101010 1
Ciphertext : 0011000101111010 0 (by XORing each plaintext bit with corresponding keystream bit)

- To decrypt this ciphertext, all we need to do is again XOR the ciphertext with the keystream:

Ciphertext : 0011000101111010 0
Keystream : 1010101010101010 1
Plaintext (XOR) : 1001101111010000 1

Block Ciphers:

Example

Plaintext : The only thing we have to fear is fear itself

Modified plaintext : Theonlythingwehavetofearisfearitself

Plaintext blocks : Theonlyt hingweha vetofear isfearit selfXend (break the plaintext into 8-character block)

Ciphertext blocks : tylnoehT ahewgnih raefotev tiraefsi dneXfles (just reverse each plaintext block)

Ciphertext : tylnoehTahewgnihraefotevtiraefsidneXfles

Components of a Modern Block Ciphers:

- Modern block ciphers normally are keyed substitution ciphers in which the key allows only partial mappings from the possible inputs to the possible outputs.
- However, modern block ciphers normally are not designed as a single unit.
- To provide the required properties of a modern block cipher, such as diffusion and confusion, a modern block cipher is made of a combination of several units:
  - Transposition units (called P-boxes)
  - Substitution units (called S-boxes)
  - Some other units

Components of a Modern Block Ciphers (continued…):

P-Boxes:

- A P-box (permutation box) is a component in a modern block cipher that transposes bits.

Types of P-Boxes:

- Three types of P-boxes are used in modern block ciphers:
  - (1) Straight P-Boxes
  - (2) Expansion P-Boxes
  - (3) Compression P-Boxes

Components of a Modern Block Ciphers (continued…):

Straight P-Boxes:

- A straight P-Box is a permutation which has *n* inputs and *n* outputs.
- There are *n!* possible mappings.
- Figure below shows a 5 x 5 straight P-box.

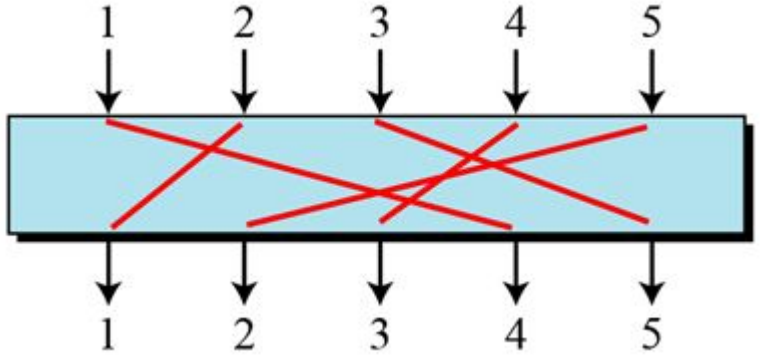


Figure: A 5x5 straight P-box

Example of all mappings for Straight P-Boxes:

- Figure below shows a 3x3 straight P-box with all 6 (3!) possible mappings.

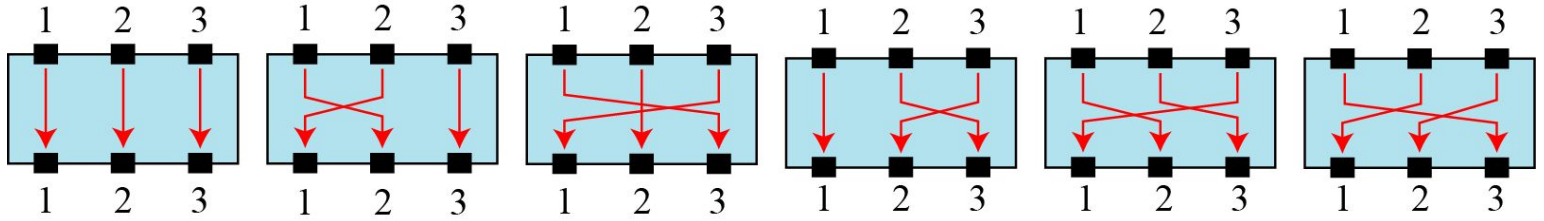


Figure: The possible mappings of a 3 × 3 Straight P-box

Components of a Modern Block Ciphers (continued…):

Compression P-Boxes:

- A compression P-box is a P-box with *n* inputs and *m* outputs where m<n.
- Some of the inputs are blocked and do not reach the output.
- Figure below shows a 5 x 3 compression P-box.

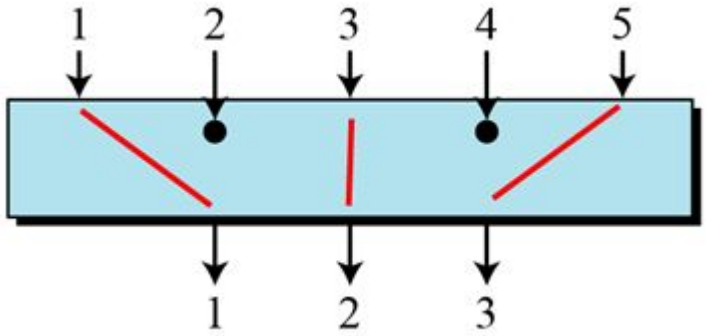


Figure: A 5x3 Compression P-box

- The compression P-boxes used in modem block ciphers are keyless normally, where a permutation table shows the rules for transposing bits.
- Compression P-boxes are used when we need to permute bits and the same time decrease the number of bits for the next stage of encryption/decryption.

Components of a Modern Block Ciphers (continued…):

Expansion P-Boxes:

- A expansion P-box is a P-box with *n* inputs and *m* outputs where m>n.
- Some of the inputs are connected to more than one output.
- Figure below shows a 3 x 5 expansion P-box.

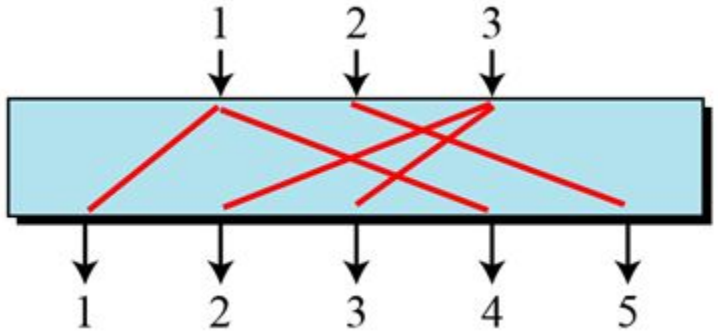


Figure: A 3 x 5 Expansion P-box

- The expansion P-boxes used in modem block ciphers normally are keyless, where a permutation table shows the rules for transposing bits.
- Expansion P-boxes are used when we need to permute bits and the same time increase the number of bits for the next stage of encryption/decryption.

Components of a Modern Block Ciphers (continued…):

S-Boxes:

- An S-box (substitution box) can be thought of as a miniature substitution cipher.
- However, an S-box can have a different number of inputs and outputs. In other words, the input to an S-box could be an *n*-bit word, but the output can be an *m*-bit word, where *m* and *n* are not necessarily the same.
- Although an S-box can be keyed or keyless, modern block ciphers normally use keyless S-boxes, where the mapping from the inputs to the outputs is predetermined.

Components of a Modern Block Ciphers (continued…):

Input-Output Relationship for a 3x2 S-Box by Table:

- The following table defines the input/output relationship for an S-box of size 3 × 2.
- The leftmost bit of the input defines the row; the two rightmost bits of the input define the column.
- The two output bits are values on the cross section of the selected row and column.

Leftmost bit

	00	01	10	11
0	00	10	01	11
1	10	00	11	01

Rightmost bits

Output bits

Table: Input-Output relationship for a 3 x 2 S-box

- Based on the above S-box table, an input of 010 yields the output 01. An input of 101 yields the output of 00.

Components of a Modern Block Ciphers (continued…):

Kinds of Product Ciphers:

- Modern block ciphers are all product ciphers, but they are divided into two classes:

- Feistel ciphers
- Non-Feistel ciphers

- Feistel ciphers:
  - In 1973, Feistel designed a very intelligent and interesting cipher that has been used for decades. Several block ciphers are based on the Feistel structure.
    - This type of ciphers use both invertible and noninvertible components.
    - A Feistel cipher can have three types of components: self-invertible, invertible, and noninvertible.
    - A Feistel cipher combines all noninvertible elements in a unit (called mixer) and uses the same unit in the encryption and decryption algorithms.
    - The block cipher DES, IDEA, RC5 (Rivest's Cipher) are good examples of a Feistel cipher. But Feistel design is not used in AES.

Components of a Modern Block Ciphers (continued…):

- Non-Feistel ciphers:
  - This type of ciphers use only invertible components.
  - A component in the encryption cipher has the corresponding component in the decryption cipher.
  - For example, S-boxes need to have an equal number of inputs and outputs to be compatible. No compression or expansion P-boxes are allowed, because they are not invertible.
    - In a non-Feistel cipher, there is no need to divide the plaintext into two halves as we saw in the Feistel ciphers.
    - The block cipher AES is a good example of a non-Feistel cipher.

Confusion and Diffusion:

- The terms diffusion and confusion were introduced by Claude Shannon to capture the two basic building blocks for product cipher.
- Every block cipher involves a transformation of a block of plaintext into a block of ciphertext, where the transformation depends on the key.
- Hence, the block cipher needs to completely obscure statistical properties of original message.
- Shannon suggested combining S & P elements to obtain diffusion and confusion.

Diffusion (Substitution):

- The idea of diffusion is to hide the relationship between the ciphertext and the plaintext. That is, the statistical relationship between the plaintext and ciphertext is made as complex as possible in order to thwart attempts to deduce the key. This will frustrate the adversary who uses ciphertext statistics to find the plaintext.
- Diffusion implies that each symbol (bit) in the ciphertext is dependent on some or all symbols in the plaintext. In other words, if a single symbol in the plaintext is changed, several or all symbols in the ciphertext will also be changed.

Confusion(Transposition):

- The idea of confusion is to hide the relationship between the ciphertext and the key. That is, the relationship between the ciphertext and the key is made as complex as possible in order to thwart attempts to discover the key. This will frustrate the adversary who tries to use the ciphertext to find the key.
- In other words, if a single bit in the key is changed, most or all bits in the ciphertext will also be changed.

Brief History of Data Encryption Standard (DES)

- The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).
  - In 1973, NIST published a request for proposals for a national symmetric-key cryptosystem.
  - A proposal from IBM, a modification of a research project called Lucifer, was accepted as DES.
  - DES was published in the Federal Register in March 1975 as a draft of the Federal Information Processing Standard (FIPS).
  - After the publication, IBM sought technical advice from the National Security Agency (NSA) for the modification of Lucifer.
- The modified version of LUCIFER was put forward as a proposal for the new national encryption standard requested by the National Bureau of Standards (NBS, now known as the National Institute of Standards and Technology, NIST). It was finally adopted in 1977 as the Data Encryption Standard -DES (FIPS PUB 46).
- Some of the changes made to LUCIFER have been the subject of much controversy even to the present day for two reasons:
  - First, the critics questioned the small key length (only 56 bits) which could make the cipher vulnerable to brute-force attack. Even though DES actually accepts a 64 bit key as input, the remaining eight bits are used for parity checking and have no effect on DES's security.
  - Second, critics were concerned about some hidden design behind the internal structure of DES. They were suspicious that some part of the structure (e.g. the S-boxes) may have some hidden trapdoor that would allow the NSA to decrypt the message without the need for the key.

Overview of DES

- DES is a 64 bit block cipher with key length 56 bits.
- In DES, the plaintext input bit string is divided into 64-bit blocks and each block is encrypted using the same 56-bit key. The same key is used for decryption. Hence, DES is a symmetric block cipher.
- It was designed by IBM in 1976 for the National Bureau of Standards (NBS), with approval from the National Security Agency (NSA).
- It had been used as a standard method of encryption until 2000, but with increase in speed in computers, it is no more considered secure as a cryptanalyst can break the code by exhaustively searching for all the keys using a fast computer.
- However, a modification of DES, called triple DES (or 3 DES), is now used which is more secure and is difficult to break.
- From 2001, DES has been replaced by a new standard known as the Advanced Encryption Standard (AES).
- After 25 years of analysis, the only security problem with DES found is that its key length is too short.
- Although it's wide spread use came to an end in 2000, its design idea is still used in most block ciphers.

DES Structure

- DES uses a 56-bit key.
- Actually, the initial key consists of 64 bits.
- However, before the DES process even starts, every 8th bit of the key is discarded to produce a 56-bit key. That is bit positions 8, 16, 24, 32, 40, 48, 56, and 64 are discarded.
- Thus, the discarding of every 8th bit of the key produces a **56-bit key** from the original **64-bit key**.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64

Figure - discording of every 8th bit of original key

DES Structure

- DES is based on the two fundamental attributes of cryptography: **substitution** (also called confusion) and **transposition** (also called diffusion).
- DES consists of 16 steps, each of which is called a **round**.
- Each round performs the steps of substitution and transposition.
- In the first step, the 64-bit plain text block is handed over to an **initial Permutation (IP)** function.
- The initial permutation is performed on plain text.
- Next, the initial permutation (IP) produces two halves of the permuted block; saying Left Plain Text (L_0) and Right Plain Text (R_0).
- Now each L_0 and R_0 go through 16 rounds of the encryption process.
- In the end, L_0 and R_0 are rejoined and a Final Permutation (FP) is performed on the combined block
- The result of this process produces 64-bit ciphertext.

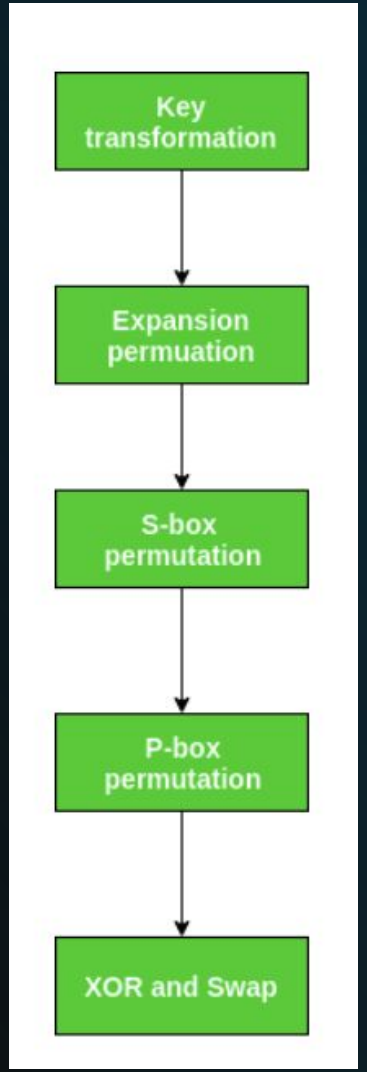
Initial Permutation(IP)

- The initial permutation (IP) happens only once and it happens before the first round.
- It suggests how the transposition in IP should proceed, as shown in the figure.
- For example, it says that the IP replaces the first bit of the original plain text block with the 58th bit of the original plain text, the second bit with the 50th bit of the original plain text block, and so on.
- This is nothing but jugglery of bit positions of the original plain text block. the same rule applies to all the other bit positions shown in the figure.

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	33	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Figure - Initial permutation table

- After IP is done, the resulting 64-bit permuted text block is divided into two half blocks.
- Each half-block consists of 32 bits, and each of the 16 rounds, in turn, consists of the broad-level steps outlined in the figure.
-



Key Transformation

- Initial 64-bit key is transformed into a 56-bit key by discarding every 8th bit of the initial key.
- Thus, for each a 56-bit key is available.
- From this 56-bit key, a different 48-bit Sub Key is generated during each round using a process called **key transformation**.
- For this, the 56-bit key is divided into two halves, each of 28 bits.
- These halves are circularly shifted left by one or two positions, depending on the round.
- **For example:** if the round numbers 1, 2, 9, or 16 the shift is done by only one position for other rounds, the circular shift is done by two positions.

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
#key bits shifted	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Figure - number of key bits shifted per round

Key Transformation(Cont..)

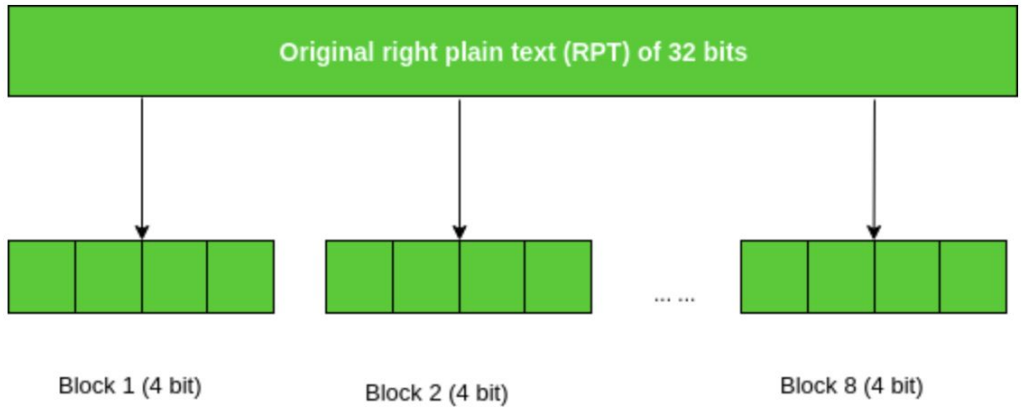
- After an appropriate shift, 48 of the 56 bits are selected.
- For selecting 48 of the 56 bits the table is shown in the figure .
- For instance, after the shift, bit number 14 moves to the first position, bit number 17 moves to the second position, and so on.
- If we observe the table , we will realize that it contains only 48-bit positions.
- Bit number 18 is discarded (we will not find it in the table), like 7 others, to reduce a 56-bit key to a 48-bit key.
- Since the key transformation process involves permutation as well as a selection of a 48-bit subset of the original 56-bit key it is called **Compression Permutation**.

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Figure - compression permutation

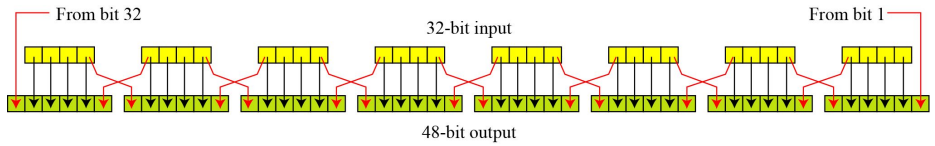
Expansion P-Box Permutation

- Recall that after the initial permutation, we had two 32-bit plain text areas called Left Plain Text(L_0) and Right Plain Text(R_0).
- During the expansion permutation, the R_0 is expanded from 32 bits to 48 bits.
- Bits are permuted as well hence called expansion permutation.
- This happens as the 32-bit R_0 is divided into 8 blocks, with each block consisting of 4 bits.
- Then, each 4-bit block of the previous step is then expanded to a corresponding 6-bit block, i.e., per 4-bit block, 2 more bits are added.



Expansion P-Box Permutation

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1



- Since R_0 is a 32-bit input and K is a 48-bit key, we first need to expand R_0 to 48 bits.
- To do this, the 32-bit R value is expanded to 48 bits using an expansion P-box permutation table (shown in Table).
- The expansion table defines a permutation plus an expansion.

Overall Process discussed so far

- The key transformation process compresses the 56-bit key to 48 bits.
- Then the expansion permutation process expands the **32-bit** $\mathbf{R_0}$ to **48-bits**.
- Now the 48-bit key is XOR with 48-bit R_0 and the resulting output is given to the next step, which is the **S-Box substitution**.

DES Rounds:

- In DES, substitution and permutation are used a number of times in iterations called rounds. Generally, the more rounds there are, the more secure the algorithm is.
- DES uses 16 rounds. Each round of DES is a Feistel cipher.

- The round takes L_{i-1} and R_{i-1} from previous round (or the initial permutation box) and creates L_i and R_i, which go to the next round (or final permutation box).
- Each round has two cipher elements: mixer and swapper. Each of these elements is invertible.
  - The swapper swaps the left half of the text with the right half. The mixer performs XOR operation.

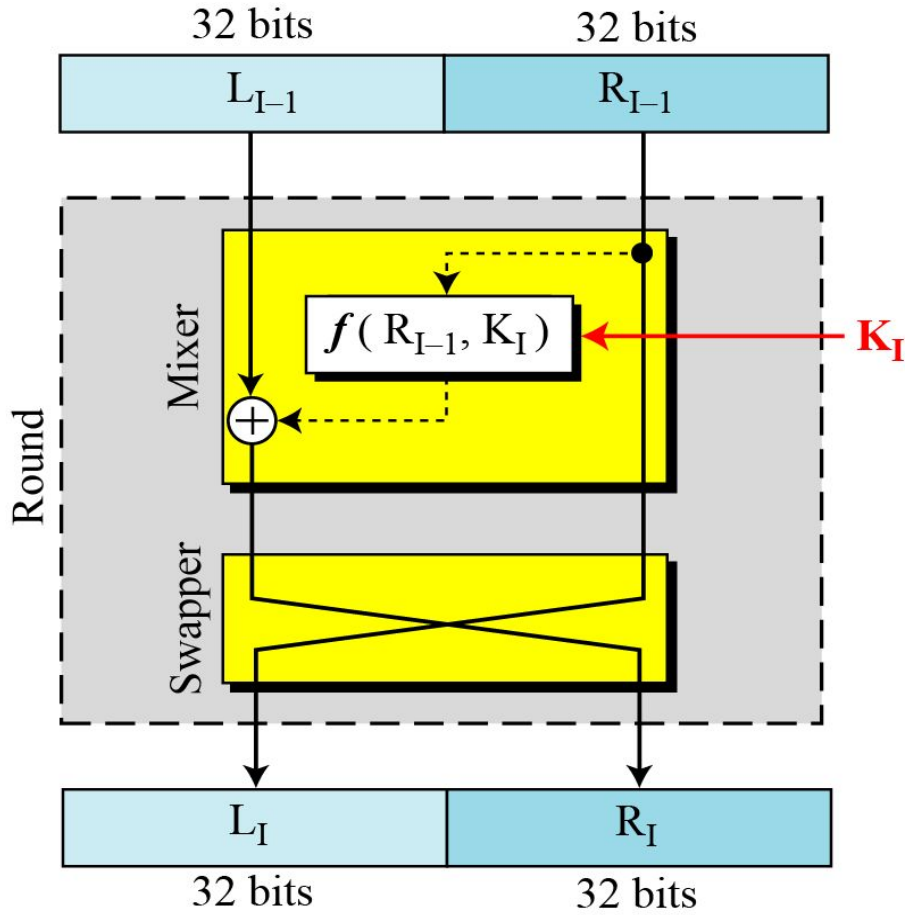


Figure: A round in DES (encryption site)

DES Round Function $f(R_{i-1},K_i)$:

- The heart of DES is DES round function.
- The round function mixes the bits of the right (R) portion using the subkey for the current round.
- It applies a 48-bit key to the rightmost 32 bits (R_{i-1}) to produce a 32-bit output.
- This function is the main part of every round and consists of four sections:

1. An expansion P-box (E-box, for 32 bit to 48 bit conversion)
2. A whitener (Exclusive-or that adds key)
3. A group of S-boxes (for 48 bit to 32 bit conversion)
4. A straight permutation P-box

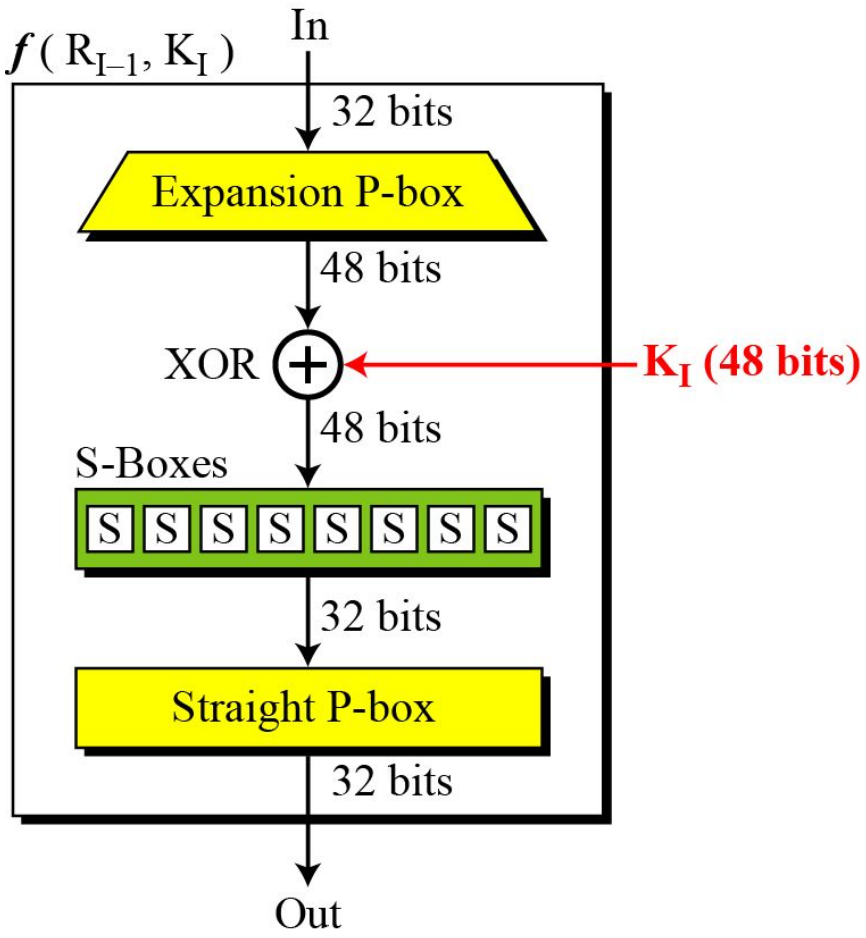


Figure: DES function

DES Round Function $f(R_{i-1},K_i)$:

Whitener (Exclusive-or):

- After the expansion permutation, DES uses the XOR operation on the expanded right section and the round key.
  - Note that both the right section and the key are 48-bits in length. Also note that the round key is used only in this operation.
- That expanded value is then exclusive-or'ed with the 48-bit subkey.

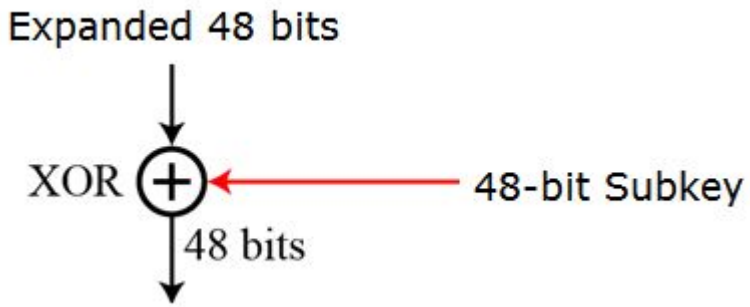


Figure: Whitener

The S-boxes (substitute 48 bits to 32 bits):

- In DES, a non-linearity is introduced into the encryption so that decryption will be computationally infeasible without the secret key.
- This is achieved with the use of S-boxes . which are basically non-linear substitution tables where either the output is smaller than the input or vice versa.
  - The S-boxes are the only non-linear operation in DES that do the real mixing (confusion).
- DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output, that is it accepts a 48-bit input and produces 32-bit number as output (defined in tables).
  - The resulting 48 bits from XOR operation are divided into eight 6-bit chunks, each of which is fed into an S-Box that mixes the bits and produces a 4-bit output (The 8 S-boxes are shown in table).
  - Those 4-bit outputs are combined into a 32-bit value.
  - The first and last bits of the 6-bit input of each S-box determine which column permutation is used. It provides non-linearity (confusion).

Table 6.3 *S-box 1*

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

Table 6.4 *S-box 2*

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	15	01	08	14	06	11	03	04	09	07	02	13	12	00	05	10
1	03	13	04	07	15	02	08	14	12	00	01	10	06	09	11	05
2	00	14	07	11	10	04	13	01	05	08	12	06	09	03	02	15
3	13	08	10	01	03	15	04	02	11	06	07	12	00	05	14	09

Table 6.5 *S-box 3*

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	10	00	09	14	06	03	15	05	01	13	12	07	11	04	02	08
1	13	07	00	09	03	04	06	10	02	08	05	14	12	11	15	01
2	13	06	04	09	08	15	03	00	11	01	02	12	05	10	14	07
3	01	10	13	00	06	09	08	07	04	15	14	03	11	05	02	12

Table 6.6 *S-box 4*

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	07	13	14	03	00	6	09	10	1	02	08	05	11	12	04	15
1	13	08	11	05	06	15	00	03	04	07	02	12	01	10	14	09
2	10	06	09	00	12	11	07	13	15	01	03	14	05	02	08	04
3	03	15	00	06	10	01	13	08	09	04	05	11	12	07	02	14

Table 6.7 *S-box 5*

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	02	12	04	01	07	10	11	06	08	05	03	15	13	00	14	09
1	14	11	02	12	04	07	13	01	05	00	15	10	03	09	08	06
2	04	02	01	11	10	13	07	08	15	09	12	05	06	03	00	14
3	11	08	12	07	01	14	02	13	06	15	00	09	10	04	05	03

Table 6.8 *S-box 6*

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	12	01	10	15	09	02	06	08	00	13	03	04	14	07	05	11
1	10	15	04	02	07	12	09	05	06	01	13	14	00	11	03	08
2	09	14	15	05	02	08	12	03	07	00	04	10	01	13	11	06
3	04	03	02	12	09	05	15	10	11	14	01	07	10	00	08	13

Table 6.10 *S-box 8*

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	13	02	08	04	06	15	11	01	10	09	03	14	05	00	12	07
1	01	15	13	08	10	03	07	04	12	05	06	11	10	14	09	02
2	07	11	04	01	09	12	14	02	00	06	10	10	15	03	05	08
3	02	01	14	07	04	10	8	13	15	12	09	09	03	05	06	11

Table 6.9 *S-box 7*

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	4	11	2	14	15	00	08	13	03	12	09	07	05	10	06	01
1	13	00	11	07	04	09	01	10	14	03	05	12	02	15	08	06
2	01	04	11	13	12	03	07	14	10	15	06	08	00	05	09	02
3	06	11	13	08	01	04	10	07	09	05	00	15	14	02	03	12

DES Round Function $f(R_{i-1},K_i)$:

4. Straight Permutation (P-box):

- The combined 32 bits from the previous step are permuted once again to produce the 32 bits output of the f-function using expansion P-box table (Shown in Table-4).

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Table-4: Straight permutation table

DES Algorithm/DES Structure/ Encryption of the DES:

- Plaintext is broken into blocks of length 64 bits.
- Each 64-bit block of plaintext is encrypted using a 56-bit key.
- A 56-bit key k is fed into a subkey generating algorithm to produce 16 round subkeys k_1, k_2, k_3, ……., k_{16} of length 48 bits each.
  - At first, an initial permutation (IP) is performed on the 64-bit block of plaintext. (The initial permutation rearranges the bits of the plaintext to form the "permuted input" based on the IP table.
  - After initial permutation, the 64-bit permuted block is divided into two 32-bit sub-blocks represented by L_0 and R_0 as the left and right sub-block respectively.
  - The encryption then proceeds through 16 rounds of identical operations using a different sub-key of length 48-bit in each round on the left and right halves of the block.
  - The 48-bit subkey ki for round i (where i=1, 2, 3,4, ……16) is generated from the original 56-bit key.

DES Algorithm/DES Structure/ Encryption of the DES (continued…):

- The output found using key k_i after i_{th} round is represented by L_i and R_i respectively where i=1, 2, 3, …..,16.
- Round i has input $L_{i-1}||R_{i-1}$ and output $L_i||R_i$ where
  - $L_i=R_{i-1}$
  - $R_i=L_{i-1} \oplus f(R_{i-1},k_i)$
- **In the final round, the left (L) and right (R) halves are swapped, so that the decryption algorithm has the same structure as the encryption algorithm.**
- After the final round (16^{th} round), the right and left halves are joined or concatenated.
- Then, a final permutation IP^{-1} (which is the inverse of the initial permutation), is applied to the 64-bit joining block.
- The output of this final permutation is the 64 bit encrypted output (ciphertext).

DES Algorithm/DES Structure/ Encryption of the DES (continued…):

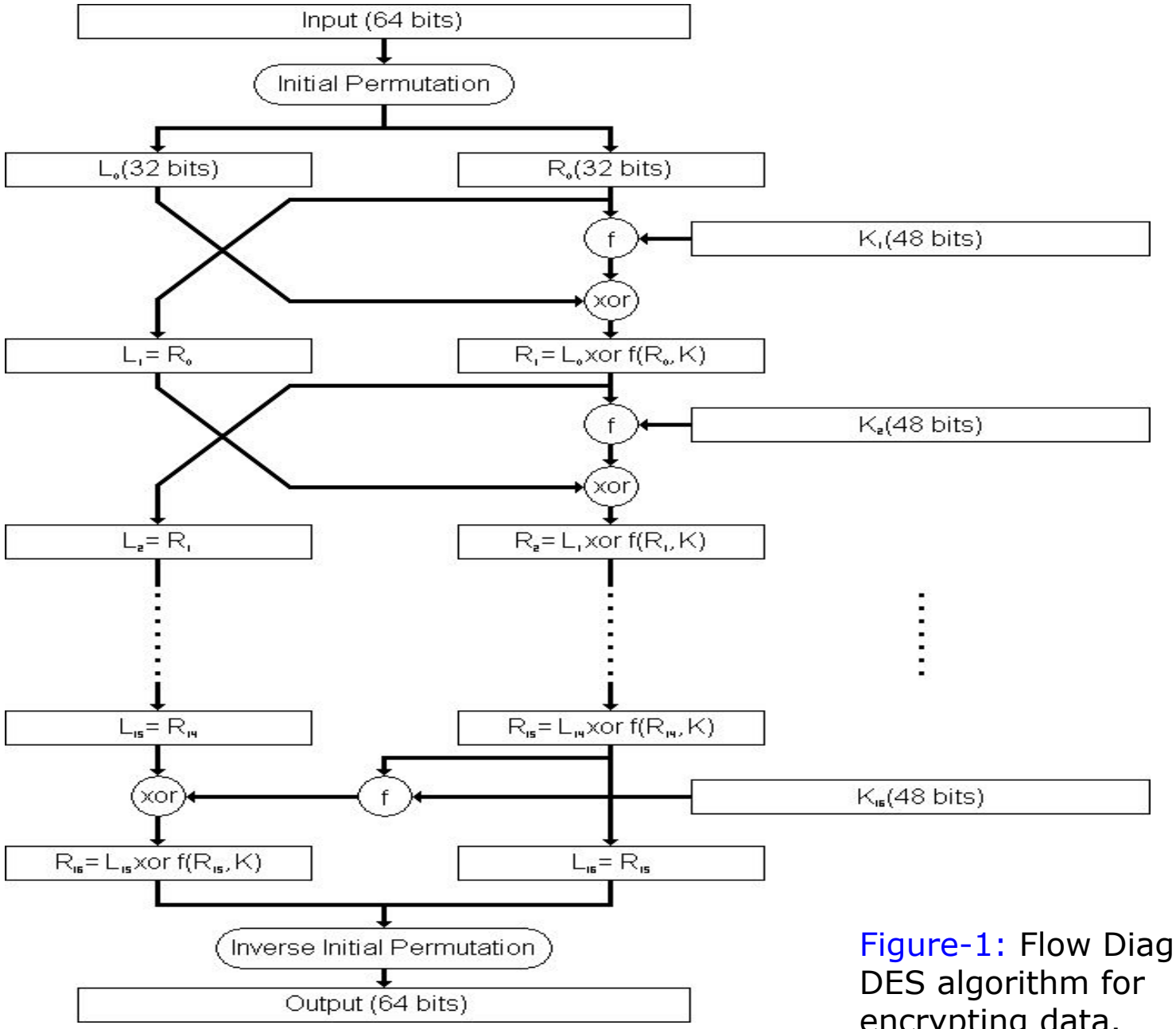


Figure-1: Flow Diagram of DES algorithm for encrypting data.

Table 6.1 *Initial and final permutation tables*

Initial Permutation	*Final Permutation*
58 50 42 34 26 18 10 02	40 08 48 16 56 24 64 32
60 52 44 36 28 20 12 04	39 07 47 15 55 23 63 31
62 54 46 38 30 22 14 06	38 06 46 14 54 22 62 30
64 56 48 40 32 24 16 08	37 05 45 13 53 21 61 29
57 49 41 33 25 17 09 01	36 04 44 12 52 20 60 28
59 51 43 35 27 19 11 03	35 03 43 11 51 19 59 27
61 53 45 37 29 21 13 05	34 02 42 10 50 18 58 26
63 55 47 39 31 23 15 07	33 01 41 09 49 17 57 25

Example on how S-Box works

Q: The input to S-box 1 is 100011. What is the output?

Answer:

- If we write the first and the sixth bits together, we get 11 in binary, which is 3 in decimal. The remaining bits are 0001 in binary, which is 1 in decimal.
- We look for the value in row 3, column 1, in Table 6.3 (S-box 1). The result is 12 in decimal, which in binary is 1100. So the input 100011 yields the output 1100.

Table 6.3 *S-box 1*

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13