Week-8: Lecture Notes

Topic:

Message authentication,
Digital signature,
Key management,
Key exchange,
Hash function.

Message authentication

- As previously mentioned, either Key (private or public) may be used for encryption with the other used for subsequent decryption. This facilitates a different form of scheme as described in the following.

- In this case A prepares a message to B using his private key to encrypt and B can decrypt it

using A's public key.

$$Y = E_{KR_a}(X)$$

$$X = E_{KU_a}(Y)$$

where, A generates a pair of keys KUa (a public key) and KRa (a private key)

- As the message was prepared using A's private key it could only come from A therefore the entire message serves as digital signature (described later)

- This scheme does not provide confidentiality because everyone has access to A's public key.

- It is not efficient because B must maintain or store both the ciphertext (as a proof of authenticity) and the decoded plaintext (for the practical use of the document).

- A more efficient way of achieving the same result is to encrypt a

small block of bits that are a function of the document.

- This block, called an authenticator, must have the property that it is infeasible to change the document without changing the authenticator.

- If the authenticator is encrypted using the sender private key then it serves as a signature that verifies the origin, content and sequencing of the document.

- This procedure to verify that received message come from the alleged source and have not been altered is called message authentication.

Authentication requirements

- In the context of communication acaross a network, the following

attacks can be identified:

1. **Disclosure:** Release of message contents to any person or process not possessing the appropriate cryptographic key.

2. **Traffic analysis:** Discovery of the pattern of traffic between parties. In a connection oriented application, the frequency and duration of connections could be determined. In either a connection oriented or connectionless environment, the number and length of messages between parties could be determined.

3. **Masquerade:** Insertion of message into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity.

4. Content modification: Changes to the contents of a message, including insertion, deletion, transposition, or modification.

5. Sequence modification: Any modification to a sequence of messages between parties including insertion, deletion and reordering.

6. Timing modification: Delay or replay of messages. In a connection oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed.

7. Repudiation: Denial of receipt of message by destination or denial of transmission of message by source.

- Message authentication may also verify sequencing and timeliness.

- A digital signature is an authentication technique that also includes measures to counter repudiation by either source or destination.

- Any message authentication or digital signature mechanism can be viewed as having fundamentally two levels

- At the lower levels, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message.

- This lower level function is then used as primitive in a high level authentication protocol that enables a receiver to verify the authenticity of a message.

- This section is concerned with the types of functions that may be used to produce an authenticator. These functions may be grouped into three classes, as follows:

1. Message Encryption: The ciphertext of the entire message serves as its authenticator.

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

- We will mainly be concerned with the last class of function, however it must be noted that hash functions and MACs are very similar except that a hash code does not require a secret key.

- With regard to the first class, this can be seen to provide authentication by virtue of the fact that only the sender and receiver know the key.

- Therefore the message could only have come from the sender.

- However there is also the problem that the plaintext message should be recognisable as plaintext message.

Digital Signature

- Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other.

- several forms of dispute between the two are possible. For example, suppose that John sends an authenticated message to Mary using one of the scheme described earliar.

- Consider the following dispute that could arise:

  — Mary may forge a different message and claim that it came from john. Mary would simply have to create a message and append an authentication code using the key that John and Mary share.

- John can deny sending the message. Because it is possible for mary to forge a message, there is no way to prove that John did infact send the message.

- Both scenarios are of legitimate concern. In situations where there is not complete trust between sender and receiver, something more than authentication is needed.

- The most attractive sol^{n} to this problem is the digital signature.

- The digital signature is analogous to the handwritten signature.
  - It must verify the author and the date and time of the signature
  - It must authenticate the contents at the time of signature.
  - It must be verifiable by third parties, to resolve disputes.

- Thus, the digital signature function includes the authentication function. On the basis of these properties, we can formulate the following requirements for a digital signature :

  - The signature must be a bit pattern that depends on the message being signed.

  - The signature must use some information unique to the sender, to prevent both forgery and denial.

  - It must be relatively easy to produce the digital signature.

  - It must be relatively easy to recognise and verify the digital signature

  - It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing

a fraudulent digital signature for a given message.

- It must be practical to retain a compy of the digital signature in storage.

Signature Scheme

- A signature scheme is a five-tuple $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ where the following conditions are satisfied:

1. $\mathcal{P}$ is a finite set of all possible message.
2. $\mathcal{A}$ is a finite set of all possible signature.
3. $\mathcal{K}$, the keyspace, is ~~the~~ a finite set of all keys.
4. $\mathcal{S}$ is a set of all possible signature algorithm
5. $\mathcal{V}$ is a set of all possible verification algorithm.

- For each $K \in \mathcal{K}$, there is a signing algorithm $sig_K \in \mathcal{S}$ and a corresponding verification algorithm $ver_K \in \mathcal{V}$. Each $sig_K : \mathcal{P} \longrightarrow \mathcal{A}$ and $ver_K : \mathcal{P} \times \mathcal{A} \longrightarrow \{true, false\}$ are functions such that the following equation is satisfied for every message $x \in \mathcal{P}$ and for every signature $y \in \mathcal{A}$:

$$ver(x, y) = true \text{ if } y = sig(x)$$
$$= false \text{ if } y \neq sig(x)$$

- For every $K \in \mathcal{K}$, the function sig_K and ver_K should be polynomial time functions. ver_K will be a public function and sig_K will be secret.

RSA Signature Scheme

- Let $n = pq$ where p and q are primes. Let $\mathcal{P} = \mathcal{A} = \mathbb{Z}_n$ and

define

$$\mathcal{K} = \{ (n, p, q, a, b) : n = pq \text{ and } ab \equiv 1 \pmod{\phi(n)} \}$$

- The values n and b are public and the values p, q, a are secret.

- $K = (n, p, q, a, b)$, define

$$sig_K(x) = x^a \bmod n \quad \text{and}$$

$$Ver_K(x, y) = true \Leftrightarrow x \equiv y^b \pmod{n}$$

$$(x, y \in \mathbb{Z}_n)$$

Elgamal Signature Scheme

- Let p be a prime such that the discrete log problem in $\mathbb{Z}_p$ is intractable, and let $\alpha \in \mathbb{Z}_p^*$ be a primitive element.

- Let $\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{A} = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$ and define

$$\mathcal{K} = \{ (p, \alpha, a, \beta) : \beta = \alpha^a \pmod{p} \}$$

- The values p, α and β are public and a is secret.

- $K = (p, \alpha, a, \beta)$ for a (secret) random number $k \in \mathbb{Z}_{p-1}^*$, define

$$sig_K(x, y) = (\gamma, \delta) \text{ where}$$

$$\gamma = \alpha^k \pmod p \text{ and}$$

$$\delta = (x - a\gamma)k^{-1} \bmod (p-1)$$

- For $x, \gamma \in \mathbb{Z}_p^*$ and $\delta \in \mathbb{Z}_{p-1}$, define

$$ver_K(x, \gamma, \delta) = true \iff \beta^\gamma \gamma^\delta \equiv \alpha^x \pmod p$$

Example

- suppose $p = 467$, $\alpha = 2$, $a = 127$ and hence $\beta = \alpha^a \pmod p = 2^{127} \bmod 467$
$= 132$

- Suppose Bob wants to sign the message $x = 100$ and he chooses the random value $k = 213$ (note that $gcd(213, 466) = 1$ and $213^{-1} \bmod 466 = 431$). Then

$r = 2^{213} \bmod 467 = 29$ and

$s = (100 - 127 \times 29) \cdot 431 \bmod 466$

$= 51$

- Anyone can verify this signature by checking that

$$132^{29} \cdot 29^{51} \equiv 189 \pmod{467}$$

and $2^{100} \equiv 189 \pmod{467}$

- Hence, the signature is valid.

Key Management.

- There are actually two distinct aspects to the use of public key encryption in this regard:
  - The distribution of public keys.
  - The use of public key encryption to distribute secret keys.

Distribution of Public keys

- Several techniques have been proposed for the distribution of public keys. Virtually all of these proposals can be grouped into the following general schemes.

  A) Public announcement
  B) Publicly available directory
  C) Public key authority
  D) Public key certificate.

Public Announcement of public keys

- The point of public key encryption should be that the public key is public.

- Thus, if there is some broadly accepted public key algorithm such as RSA, any participant can send his or her public key to any other participant or broadcast

the key to the community at large (figure 1).

- Although this approach is convenient, it has a major weakness. Anyone can forge such a public announcement.

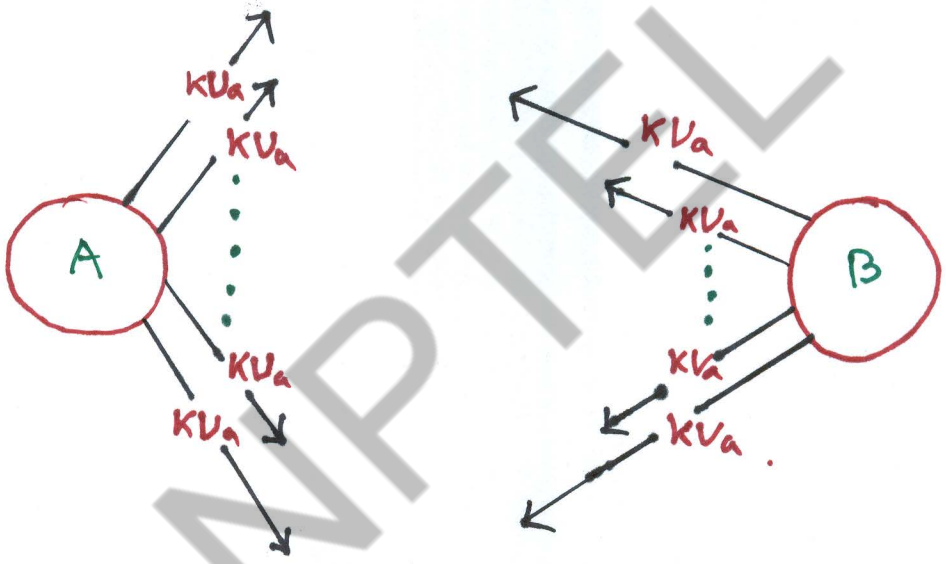


Figure 1: Uncontrolled public key.

Publicly Available Directory

- A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys.

- Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organisation (figure 2).

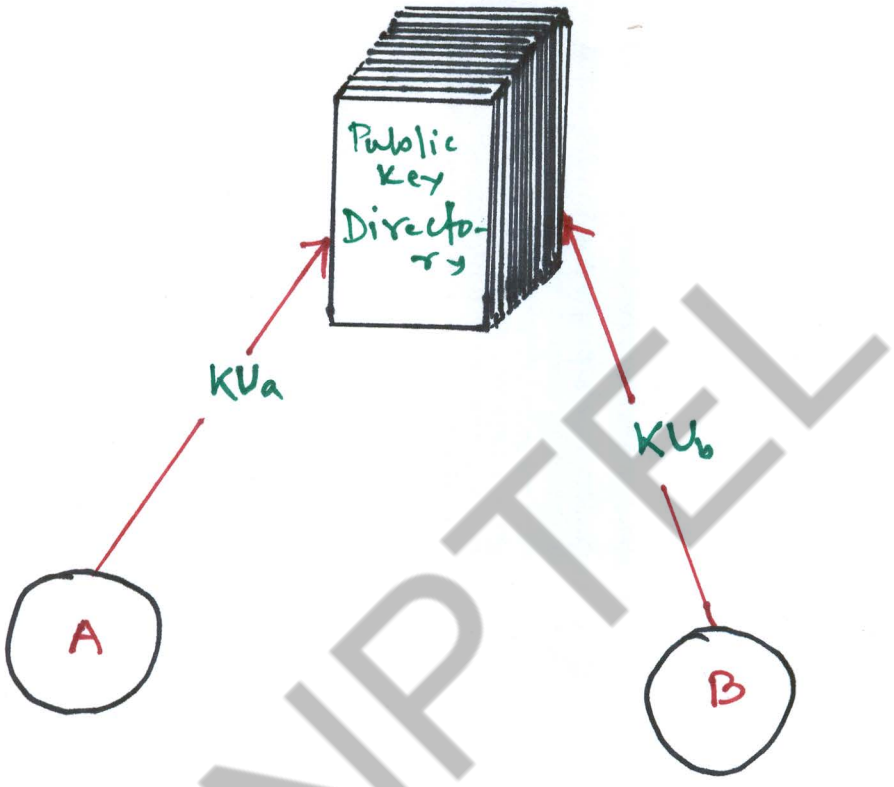


Figure 2: Public Key Publication.

- Such a scheme would include the following elements:
  - The authority maintains a directory with a {name, public key} entry for each participant.

- Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.

- A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.

- Periodically, the authority publishes the entire directory or updates to the directory. For example, a hard copy version much like a telephone book could be published, or updates, could be listed in a widely circulated newspaper.

- Participants could also access the directory electronically.

For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

- This scheme is clearly more secure than individual public announcements, but still has vulnerabilities.
- If an opponent succeeds in obtaining or computing the private key of the directory authority, the opponent could authoritatively pass out counterfeit public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participants.
- Another way to achieve the same end is for the opponent to tamper with the records kept by the authority.

Public Key Authority

- stronger security for public key distribution can be achieved by

providing tighter control over the distribution of public keys from the directory.

- As before, the scenario assumes that a central authority maintains a dynamic directory of public keys of all participants.

- In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key.

- However this is not perfect as the public key authority could be somewhat of a bottleneck in the system.

- The reason for this is that a user must appeal to the authority for a public key for every other user that it wishes to contact.

- Also the directory of names and public keys maintained by the authority is vulnerable to tampering.

Public Key Certificate

- An alternative approach to the above is the use of certificate that can be used by participants to exchange keys without contacting a public key authority.

- Each certificate, containing a public key and other information, is created by a certificate authority and is given to the participant with the matching private key.

- A participant conveys its key information to another by transmitting its certificate.

- Other participants can verify that the certificate was created by the authority.

- Four requirements can be placed on this particular scheme:

  1. Any participant can read a certificate to determine the name and public key of the certificate's owner.

  2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.

  3. Only the certificate authority can create and update certificates.

  4. Any participant can verify the currency of the certificate.

- An example of this scheme can be seen using the following transaction:

$$C_A = E_{KR_{auth}}[T, ID_A, KU_A]$$

- Where C_A is A's certificate, KRauth is the private key of the certificate authority, ID_A is A's identification and KU_A is A's public key.

- A can then pass C_A to any participant who reads and verifies it as follows:

$$D_{KU_{auth}}[C_A] = D_{KU_{auth}}[E_{KU_{auth}}[T, ID_A, KU_A]]$$

$$= (T, ID_A, KU_A).$$

Public Key Distribution of Secret Key

- Once public keys have been distributed or have become accessible, secure communication that thwarts eavesdropping, tampering or both is possible.

- However, few users will wish to make exclusive use of public-key encryption for communications because of the relatively slow data rates that can be achieved.

- Accordingly, public key encryption is more reasonably viewed as a vehicle for the distribution of secret keys to be used for conventional encryption.

- Simply Secret Key Distribution: An extremely simple scheme put forward by Ralph Merkle is illustrated in figure 9. If A wishes to communicate with B, the following procedure is employed

1. A generates a public/private key pair $\{KU_A, KR_A\}$ and transmits a message to B consisting of KU_A and an identifier of A, ID_A.

2. B generates a secret key K_s and transmit it to A, encrypted with A's public key.

3. A computes $D_{KR_A}[E_{KU_A}[K_s]]$ to recover the secret key. Since only A can decrypt the message, only A and B will know the identity of K_s.

4. A discards KU_A and KR_A and B discards KU_A.

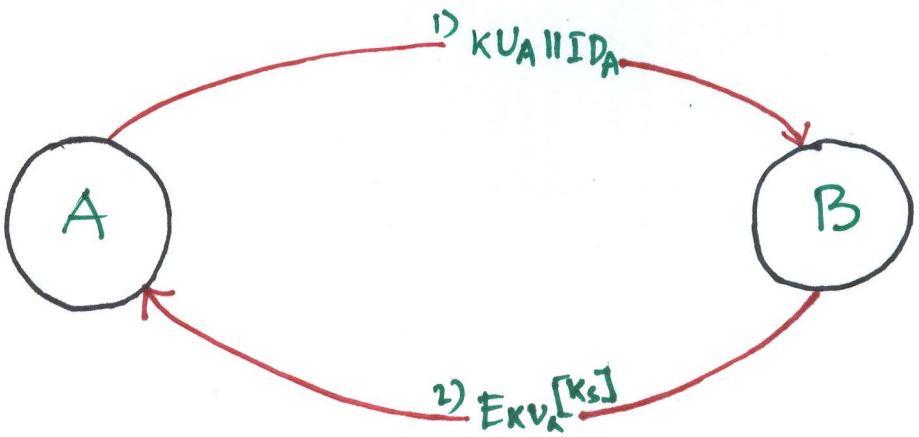


Figure 3: Simple use of Public-key Encryption to establish a session key.

- A and B can now securely communicate encryption and the session keys K_s.
- At the completion of the exchange, both A and B discard K_s.
- Despite its simplicity, this is an attractive protocol.
- No keys exist before the start of the communication and none exist after the completion of communication.
- Thus, the risk of compromise of the keys is minimal. At the same time, the communication is secure from eavesdropping.
- The protocol is vulnerable to an active attack.
- If an opponent E has control of the intervening communications channel, then he can compromise the communications in the following way without being detected :

1. A generates a public/private key pair $\{KU_A, KR_A\}$ and transmits a message intended for B consisting of KU_A and identifier of A, ID_A.

2. E intercepts the message, creates its own public/private key pair $\{KU_E, KR_E\}$ and transmits $KU_E || ID_A$ to B.

3. B generates a secret key K_s and transmits $E_{KU_E}[K_s]$.

4. E intercepts the message and learns K_s by computing $D_{KR_E}[E_{KR_E}[K_s]]$

5. E transmits $E_{KU_A}[K_s]$ to A.

- The result is that both A and B know K_s and are unaware that K_s has also been revealed to E.

- A and B can now exchange messages using K_s. E no longer actively interferes with the communications channel but simply eavesdrops.

- Knowing K_s, E can decrypt all messages, and both A and B are aware of the problem.

- Thus, this simple protocol is only useful in an environment where the only threat is eavesdropping.

- Next we will discuss Secret key Distribution with confidentiality and Authentication.

- Figure 4, provides protection against both active and passive attacks.

- For this example it is assumed that A and B have exchanged public keys by one of the schemes described earliar in this section.

- Then the following steps occure:

  1. A uses B's public key to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1) which

is used to uniquely identify this transaction.

2. B sends a message to A encrypted with KU_A and containing A's nonce (N_1) as well as a new nonce (N_2) generated by B. Since only B could have decrypted message (1), the presence of N_1 in message (2) assures A that the correspondent is B.

3. A returns N_2, encrypted using B's public key, to assure B that its correspondent is A.

4. A selects a secret key K_s and sends $M = E_{KU_b}[E_{KR_A}[K_s]]$ to B. Encryption of this message with B's public key ensures that only B can read it, encryption with A's private key ensures that only A could have sent it.

5. B computes $D_{KU_A}[D_{KR_b}[M]]$ to recover the secret key.

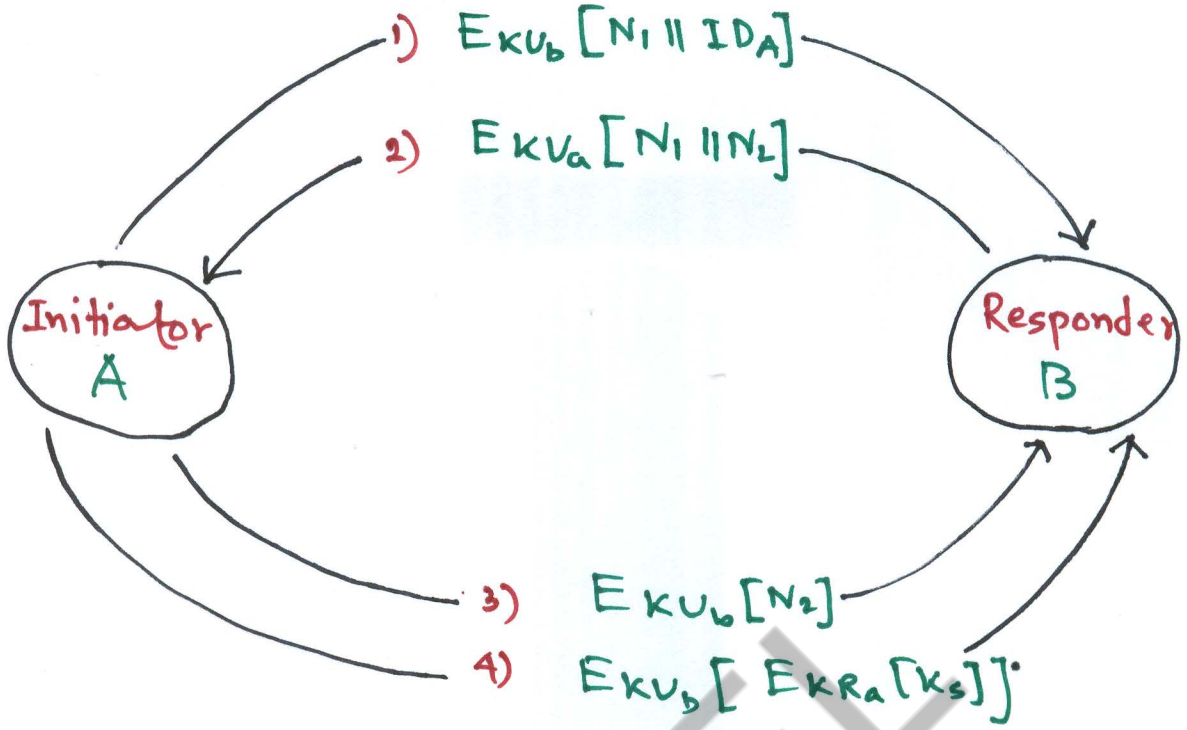


Figure 4: Public Key Distribution of secret keys.

- This scheme ensures both confidentiality and authentication in the exchange of a secret key.

A Hybrid Scheme

- Yet another way to use public key encryption to distribute secret keys is a hybrid approach in use on IBM mainframes.

- This scheme retains the use of a key distribution centre (KDC) that shares a secret master key with each user and distributes secret session keys encrypted with the master key.

- A public key scheme is used to distribute the master keys.

- The following rationale is provided for using this three level approach

  - Performance: There are many applications, especially transaction-oriented applications in which the session keys change frequently. Distribution of session keys by public key encryption could degrade overall system performance because of the relatively high computational load of public key encryption and decryption. With a three level hierarchy, public key encryption is used only occassionally to update the

master key between a user and the KDC.

- Backward Compatibility: The hybrid scheme is easily overlaid on an existing KDC scheme, with minimal disruption of software changes.

- The addition of a public key layer provides a secure, efficient means of distributing master keys. This is an advantage in a configuration in which a single KDC serves a widely distributed set of users.

Diffie-Hellman Key Exchange

- The first published P-K algorithm appeared in the paper by Diffie and Hellman that defined public key cryptography however it is limited to the secure exchange of a secret key and not of a message.

- The security of the scheme depends on the difficulty of computing discrete logarithms.

- The Diffie-Hellman key exchange consists of two publicly known numbers : a prime number p and an integer α that is the primitive root of q.

- Suppose the users A and B wish to exchange a key.

- User A selects a random integer $X_A < q$ and computes $Y_A = \alpha^{X_A} \bmod q$.

- Similarly, user B independently selects a random integer $X_B < q$ and computes $Y_B = \alpha^{X_B} \bmod q$.

- Each side keeps the X values private and makes the Y value available publicly to the other side.

- User A computes the key as $K = (Y_B)^{X_A} \bmod q$ and user B computes the key as $K = Y_A^{X_B} \bmod q$

- These two calculations produce identical results and the result is that the two sides have exchanged a secret key.

- This can be seen because:

$$K = Y_B^{X_A} \bmod q.$$
$$= (\alpha^{X_B} \bmod q)^{X_A} \bmod q$$
$$= (\alpha^{X_B})^{X_A} \bmod q$$
$$= (\alpha^{X_A})^{X_B} \bmod q$$
$$= (\alpha^{X_A} \bmod q)^{X_B} \bmod q.$$
$$= (Y_A)^{X_B} \bmod q.$$

- Furthermore, because X_A and X_B are private, an opponent is forced to take a discrete logarithm to determine the key.

- For example, attacking the secret key of user B, the opponent must compute:

$$X_B = \text{ind}_{\alpha,q}(Y_B)$$

- where $\text{ind}_{\alpha,q}(Y_B)$ is the discrete logarithm, or index, of Y_B for the base $\alpha \bmod q$.

- The scheme can be summarised as shown in figure 5

- For example, lets say we have the value $q = 353$ and a primitive root $\alpha = 3$.

- We can see that $\alpha = 3$ is a primitive root of $q = 353$ due to the following reasoning.

- If α is a primitive root of prime q then the set of numbers $\{\alpha, \alpha^2, \ldots, \alpha^{\phi(q)}\}$ are distinct modulo q and hence form the set $\{1, 2, \ldots, q-1\}$ in some order.

- In this case $\alpha = 3$ and it can be seen to be a primitive root of $q = 353$ as $\{3 \bmod 353, 3^2 \bmod 353, \ldots, 3^{353} \bmod 353\}$ which contains all the elements of $\{1, 2, \ldots, 352\}$.

Global Public Elements Prime number $\alpha < q$ and α is a primitive root of q
User A Key Generation secret private X_A $\quad X_A < q$ Calculat public Y_A $\quad Y_A = \alpha^{X_A} \bmod q$
User B Key Generation secret private X_B $\quad X_B < q$ Calculate public Y_B $\quad Y_B = \alpha^{X_B} \bmod q$
Generation of secret key by user A $K = (Y_B)^{X_A} \bmod q$
Generation of secret key by user B. $K = (Y_A)^{X_B} \bmod q$

Figure 5: The Diffie-Hellman Key Exchange Algorithm.

- Suppose A and B select the private keys $X_A = 97$ and $X_B = 233$ respectively.

- To calculate the secret key K user A calculates:

$$Y_A = \alpha^{X_A} \bmod q = 3^{97} \bmod 353 = 40.$$

- Similarly user B calculates

$$Y_B = \alpha^{X_B} \bmod q = 3^{233} \bmod 353 = 248$$

- Then we have

$$K = 248^{97} \bmod 353 = 40^{233} \bmod 353$$
$$= 160.$$

- We assume the attacker would have q, α, Y_A, Y_B which for this example might be enough using a brute force approach. However with large numbers this becomes impractical.

Hash Functions

- A Hash value is generated by a function H of the form:

$$h = H(M)$$

- where M is a variable length message, and H(M) is the fixed length hash value (also referred to as a message digest or hash code)

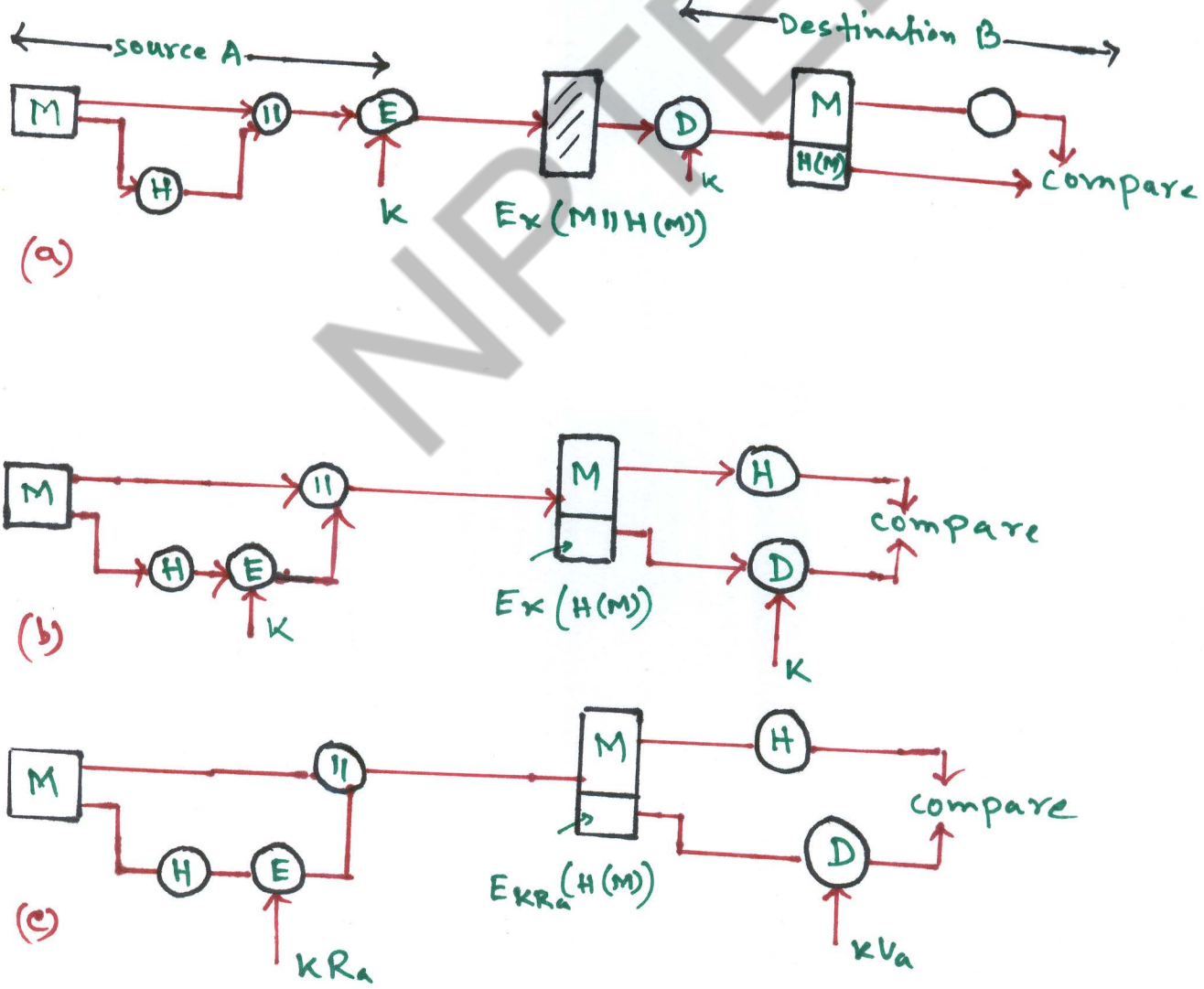


Figure 6: Basics uses of Hash function.

- Figure 6 and Figure 7 show the basic uses of hash function whereas figure 8 shows the general structure of a hash code.

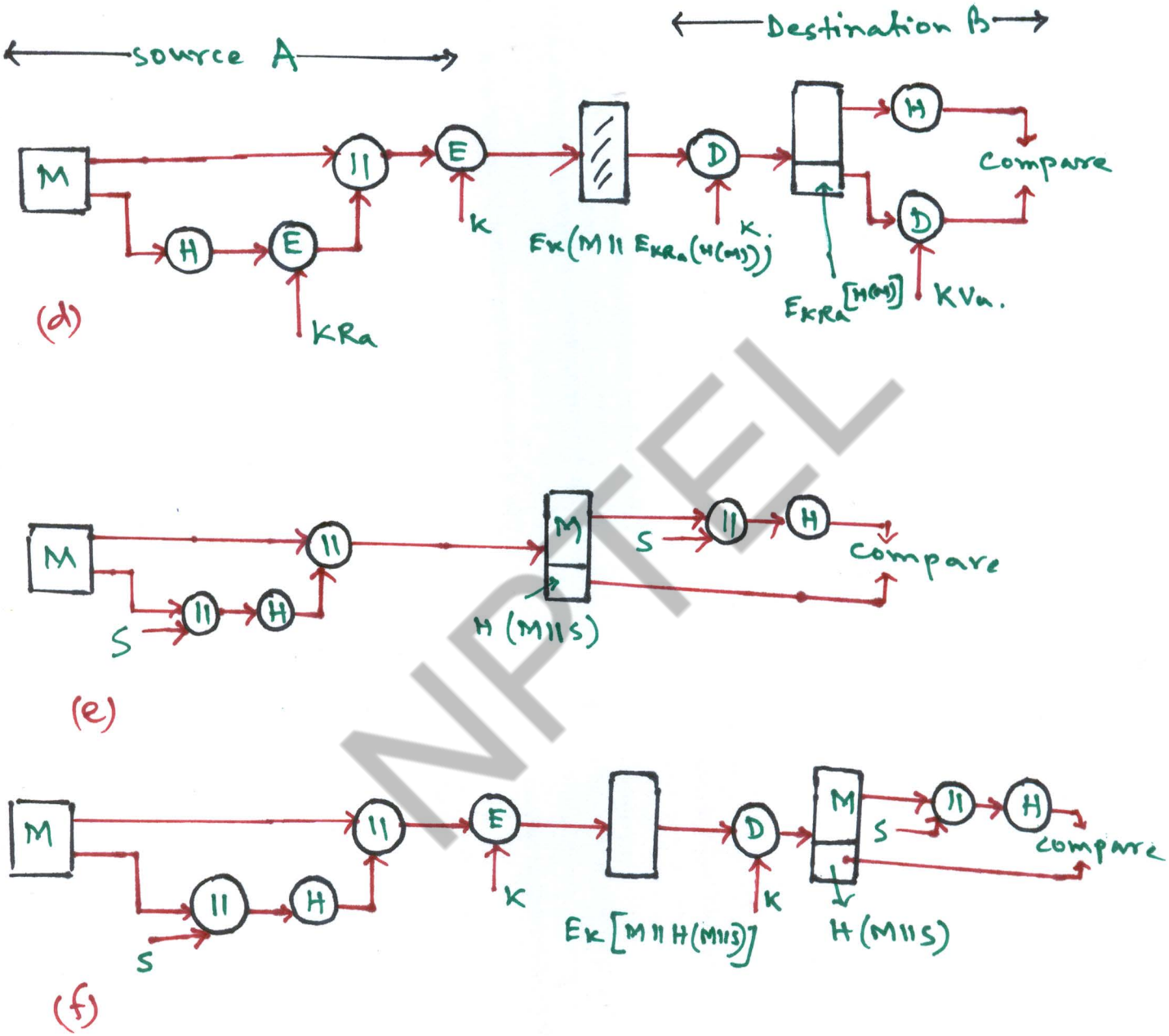


Figure 7: Basic use of hash function (contd.)

- The hash value is appended to the message at the source at the time when the message is assumed or known to be correct.

- The receiver authenticates that message by recomputing the hash value.

- Because the hash function itself is not considered to be secret, some means is required to protect the hash value (see figure 6 and 7).

- We begin by examining the require ments for a hash function to be used for message authentication.

- The purpose of a hash function is to produce a "fingerprint" of a file, message or other block of data.

- To be useful for message authen tication, a hash function H must have the following properties:

1. H can be applied to a block of data of any size.

2. H produces a fixed length output.

3. $H(x)$ is relatively easy to compute for any given x, making both hardware and software implementation practical.

4. For any given code h, it is computationally infeasible to find x such that $H(x) = h$.

5. For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$ (sometimes referred to as weak collision property.

6. It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$ (sometimes referred to as strong collision property).

- The first three properties are requirements for the practical application of a hash function to message authentication.

- The fourth property is the "one way" property; it is easy to generate a code given a message but virtually impossible to generate a message given a code.

- This property is important if the authentication technique involves the use of secret value.

- The secret value itself is not sent, however, if the hash function is not one-way, an attacker can easily discover the secret value.

- If the attacker can observe or intercept a transmission, the attacker obtains the message M and the hash code $C = H(S_{AB} \| M)$

- The attacker then inverts the hash function to obtain $S_{AB} || M = H^{-1}(c)$.
- Because, the attacker now has both M and $S_{AB} || M$, it is a trivial matter to recover S_{AB}.
- The fifth property guarantees that an alternative message hashing to the same value as a given message can not be found.
- This prevents forgery when an encrypted hash code is used (see figure 6b and 6c)
- For these cases, the opponent can read the message and therefore generate its hash code. But, because the opponent does not have the secret key, the opponent should not be able to alter the message without detection.

- If this property were not true, an attacker would be capable of the following sequence:

  1. Observe or intercept a message plus its encrypted hash code.

  2. Generate an unencrypted hash code from the message.

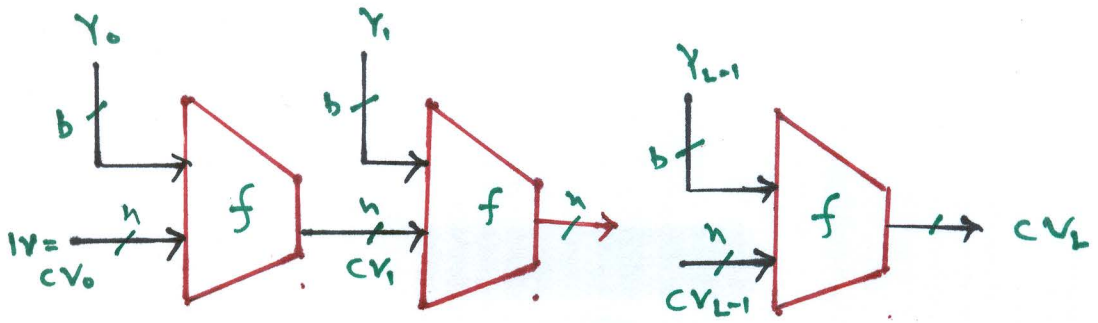
  3. Generate an alternate message with the same hash code.

- A hash function that satisfies the first five properties in the preceding list is referred to as a weak hash function.

- If the sixth property is also satisfied, then it is referred to as a strong hash function.

- The sixth property protects against a sophisticated class of attack known as birthday attack

- Figure 8 shows the general structure of a secure hash code.



IV = Initial Value
CV = Chaining variable
Y_i = ith input block
f = compression algorithm
L = number of input blocks.
n = length of hash code
b = length of input block.

- In the next week we are going to study a specific algorithm (SHA-1) which will be seen to have this format.