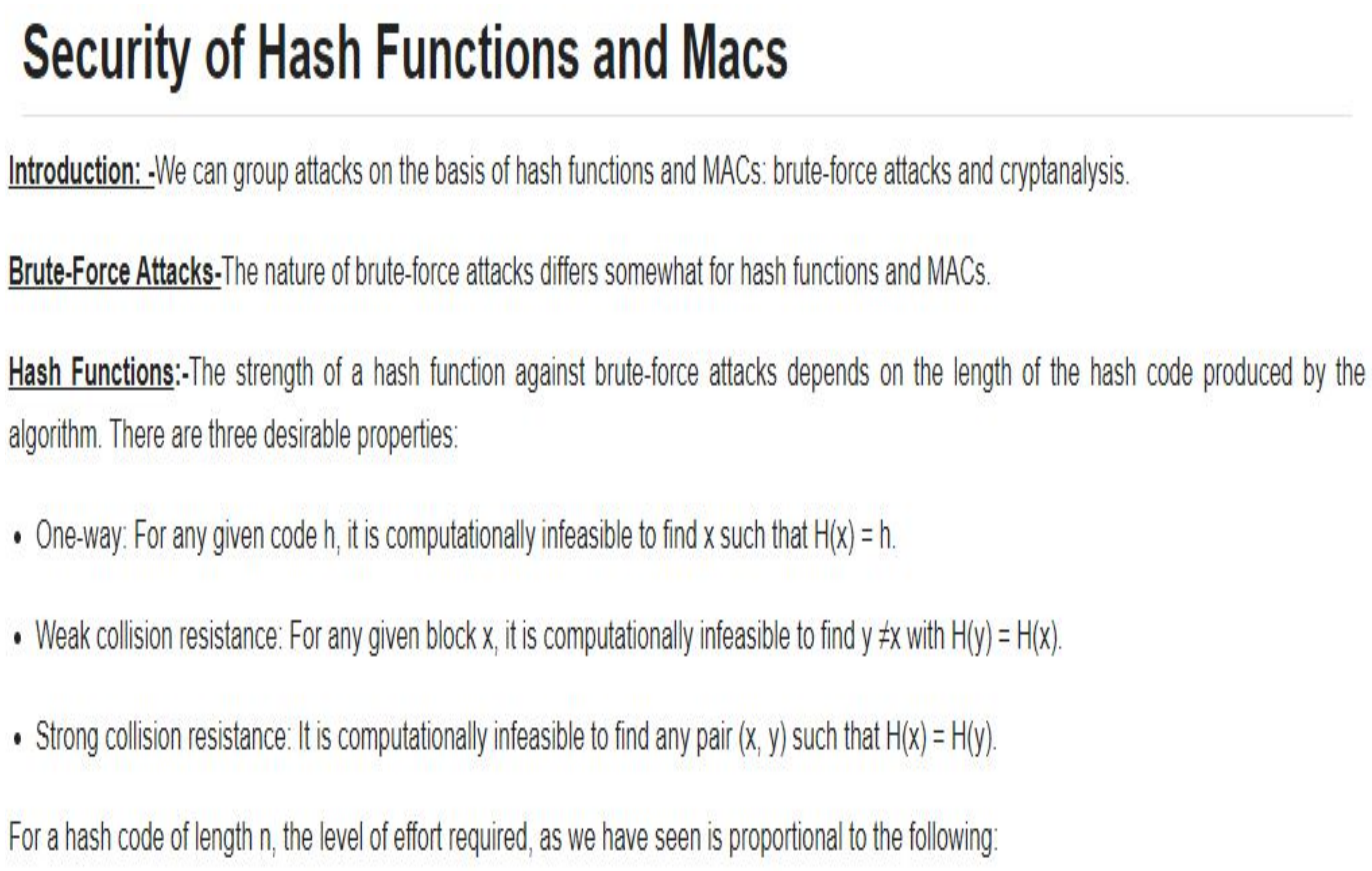


Security of Hash Functions and Macs

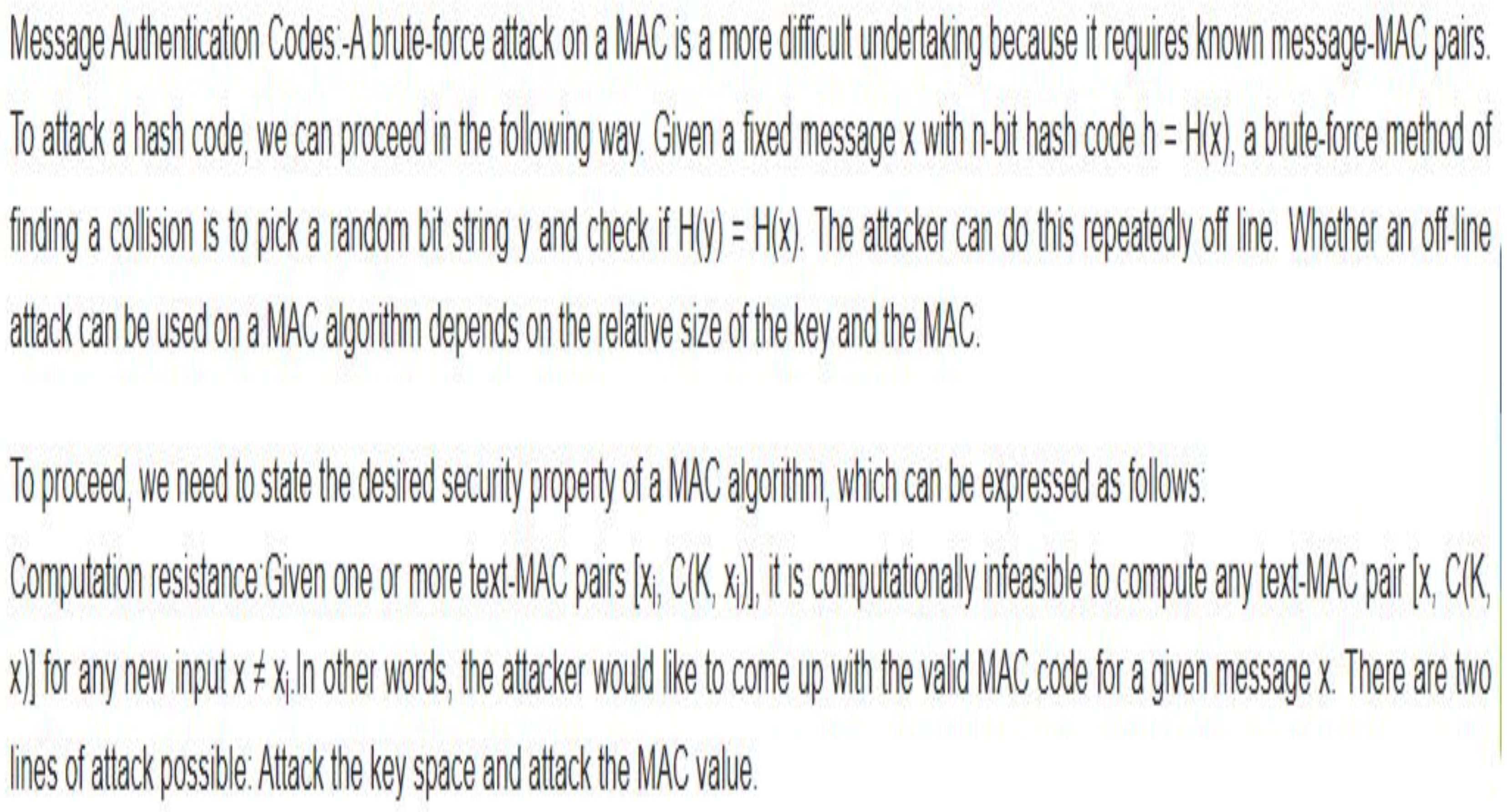
Introduction: -We can group attacks on the basis of hash functions and MACs: brute-force attacks and cryptanalysis.

Brute-Force Attacks-The nature of brute-force attacks differs somewhat for hash functions and MACs.

Hash Functions:-The strength of a hash function against brute-force attacks depends on the length of the hash code produced by the algorithm. There are three desirable properties:

- One-way: For any given code h, it is computationally infeasible to find x such that H(x) = h.
- Weak collision resistance: For any given block x, it is computationally infeasible to find y ≠x with H(y) = H(x).
- Strong collision resistance: It is computationally infeasible to find any pair (x, y) such that H(x) = H(y).

For a hash code of length n, the level of effort required, as we have seen is proportional to the following:

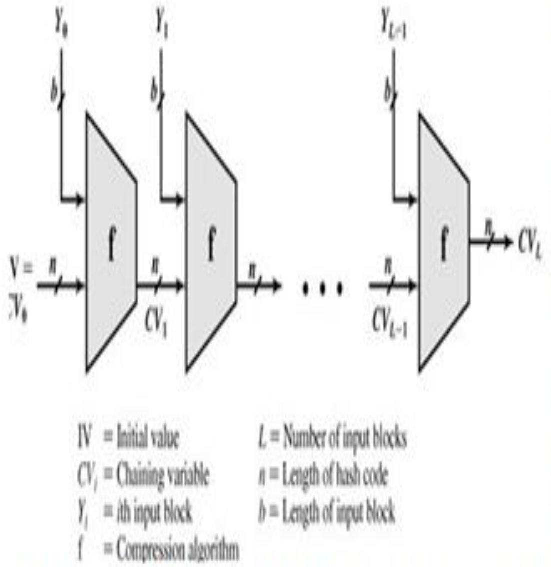


Message Authentication Codes:-A brute-force attack on a MAC is a more difficult undertaking because it requires known message-MAC pairs. To attack a hash code, we can proceed in the following way. Given a fixed message x with n-bit hash code h = H(x), a brute-force method of finding a collision is to pick a random bit string y and check if H(y) = H(x). The attacker can do this repeatedly off line. Whether an off-line attack can be used on a MAC algorithm depends on the relative size of the key and the MAC.

To proceed, we need to state the desired security property of a MAC algorithm, which can be expressed as follows:

Computation resistance:Given one or more text-MAC pairs $[x_i, C(K, x_i)]$, it is computationally infeasible to compute any text-MAC pair $[x, C(K, x)]$ for any new input $x \neq x_i$.In other words, the attacker would like to come up with the valid MAC code for a given message x. There are two lines of attack possible: Attack the key space and attack the MAC value.

General Structure of Secure Hash Code



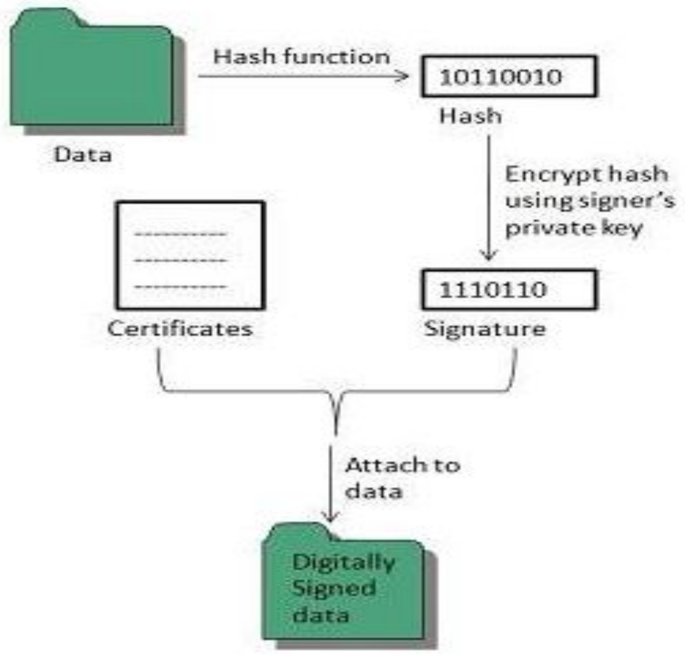
Cryptanalysis-The way to measure the resistance of a hash or MAC algorithm to cryptanalysis is to compare its strength to the effort required for a brute-force attack. That is, an ideal hash or MAC algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort.

Hash Functions:-In past few years there has been considerable effort, and some successes, in developing cryptanalytic attacks on hash functions. To understand these, we need to look at the overall structure of a typical secure hash function, This structure, referred to as an iterated hash function and is the structure of most hash functions in use today, including SHA and Whirlpool. The hash function takes an input message and partitions it into L fixed-sized blocks of b bits each. If necessary, the final block is padded to b bits. The final block also includes the value of the total length of the input to the hash function. The inclusion of the length makes the job of the opponent more difficult.

Message Authentication Codes:-There is much more variety in the structure of MACs than in hash functions, so it is difficult to generalize about the cryptanalysis of MACs.Far less work has been done on developing such attacks.

Digital Signatures

- A **digital signature** is an attachment to any piece of electronic information which identifies the originality of that document.
- To provide authenticity, integrity and non-repudiation, digital signature is the best thing. It makes our internet use safer. It is being used in electronic mail, electronic funds transfer, software distribution etc.



Requirements

- Message authentication protects two parties who exchange messages from any third party
- But it doesn't protect the two parties against each other
  1. Bob may forge a different message and claim that it came from Alice
  2. Alice can deny sending the message
- The most attractive solution to this problem is the digital signature

Digital Signature must have following properties:

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

Digital Signature requirements

- 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 recognize and verify the digital signature.

Digital Signature requirements…

- 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 copy of the digital signature in storage.

Digital Signatures

- Two categories
  - Direct Digital Signature.
  - Arbitrated Digital Signature.

Direct Digital Signature

- Involves only the communication parties
- Dig Sign can be formed in 2 ways:
  - encrypting entire message with senders Private Key
  - encrypting hash code of message with senders Pvt Key
- In case of disputes third party can verify

Direct Digital Signature

- Has some weakness
  - To deny a message, Sender claim that his Pvt Key was stolen
  - Solution is Time stamp and prompt reporting.

) Explain three different Arbitrated Digital Signature Techniques. (9)

Arbitrated Digital Signature Techniques

(a) Conventional Encryption, Arbiter sees message

$$\textbf{(1) } \mathrm{X} \rightarrow \mathrm{A} : M \,\|\, \mathrm{E}_{\mathrm{Kxa}} [\, ID_x \,\|\, \mathrm{H}(M) \,]$$

$$\textbf{(2) } \mathrm{A} \rightarrow \mathrm{Y} : \mathrm{E}_{\mathrm{Kay}} [\, ID_x \,\|\, M \,\|\, \mathrm{E}_{\mathrm{Kxa}} [\, ID_x \,\|\, \mathrm{H}(M) \,] \,\|\, T]$$

Notation:

X	= sender
Y	= recipient
A	= Arbiter
M	= message
T	= timestamp

- The problems associated with direct digital signatures can be addressed by using an arbiter.
- As with direct signature schemes; there is a variety of arbitrated signature schemes.
- In general terms, they all operate as follows. Every signed message from a sender X to a receiver Y goes first to an arbiter A, who subjects the message and its signature to a number of tests to check its origin and content.
- The message is then dated and sent to Y with an indication that it has been verified to the satisfaction of the arbiter. The presence of A solves the problem faced by direct signature schemes: that X might disown the message.

- The arbiter plays a sensitive and crucial role in this sort of scheme, and all parties must have a great deal of trust that the arbitration mechanism is working properly.
- In the first, symmetric encryption is used. It is assumed that the sender X and the arbiter A share a secret key K_{xa} and that A and Y share secret key K_{ay}.
- X constructs a message M and computes its hash value H(M). Then X transmits the message plus a signature to A. The signature consists of an identifier ID_X of X plus the hash value, all encrypted using K_{xa}. A decrypts the signature and checks the hash value to validate the message.

- Then A transmits a message to Y, encrypted with Kay. The message includes ID_X, the original message from X, the signature, and a timestamp.
- Y can decrypt this to recover the message and the signature. The timestamp informs Y that this message is timely and not a replay. Y can store M and the signature. In case of dispute, Y, who claims to have received M from X, sends the following message to A:

Arbitrated Digital Signature Techniques

(b) Conventional Encryption, Arbiter does not see message

(1) $X \rightarrow A : ID_x \parallel E_{Kxy}[M] \parallel E_{Kxa}[ID_x \parallel H(E_{Kxy}[M])]$

(2) $A \rightarrow Y : E_{Kay}[\, ID_x \parallel E_{Kxy}[M] \parallel E_{Kxa}[ID_x \parallel H(E_{Kxy}[M])] \parallel T]$

Arbitrated Digital Signature Techniques

(c) Public-key Encryption, Arbiter does not see message

(1) $\mathbf{X \rightarrow A} : \mathit{ID}_x \parallel \mathrm{E}_{\mathrm{KRx}}[\, \mathit{ID}_x \parallel \mathrm{E}_{\mathrm{KUy}}(\mathrm{E}_{\mathrm{KRx}}[M])\,]$

(2) $\mathbf{A \rightarrow Y} : \mathrm{E}_{\mathrm{KRa}}[\, \mathit{ID}_x \parallel \mathrm{E}_{\mathrm{KUy}}[\mathrm{E}_{\mathrm{KRx}}[M]] \parallel T\,]$

Authentication Protocols

Authentication Protocols

- Used to convince parties of each others identity and to exchange session keys
- Focus on two areas:
    - Mutual Authentication
    - One-way Authentication

Mutual Authentication

- Key issues are
  - confidentiality – to protect session keys
  - timeliness – to prevent replay attacks

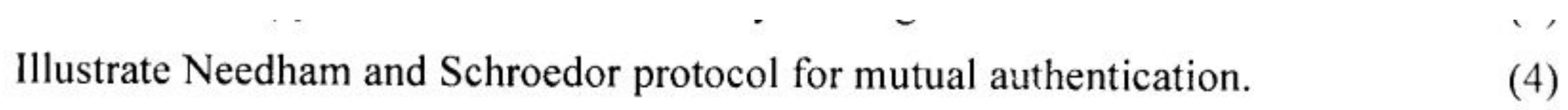
Replay Attacks

- where a valid signed message is copied and later resent
  - simple replay
  - repetition that can be logged
  - repetition that cannot be detected
  - backward replay without modification

- Countermeasures for Replay Attacks:
  - use of sequence numbers (generally impractical)
  - timestamps (needs synchronized clocks)
  - challenge/response (using unique nonce)

Using Symmetric Encryption

- Uses a two-level hierarchy of keys
- Usually with a trusted Key Distribution Center (KDC)
  - Each party shares own master key with KDC
  - KDC generates session keys used for connections between parties
  - Master keys used to distribute these to them


Illustrate Needham and Schroedor protocol for mutual authentication. (4)

Needham-Schroeder Protocol

- The goal of the protocol is to establish mutual authentication between two parties A and B in the presence of adversary, who can
- Intercept messages;
- Delay messages;
- Read and copy messages;
- Generate messages, But who does not know
- secret keys of principals, which they share with the authentication server S.
- A and B obtain a secret shared key though authentication server S.
- The protocol uses shared keys encryption/decryption

Needham-Schroeder Protocol

- For session between a and b mediated by KDC
- Protocol overview is:

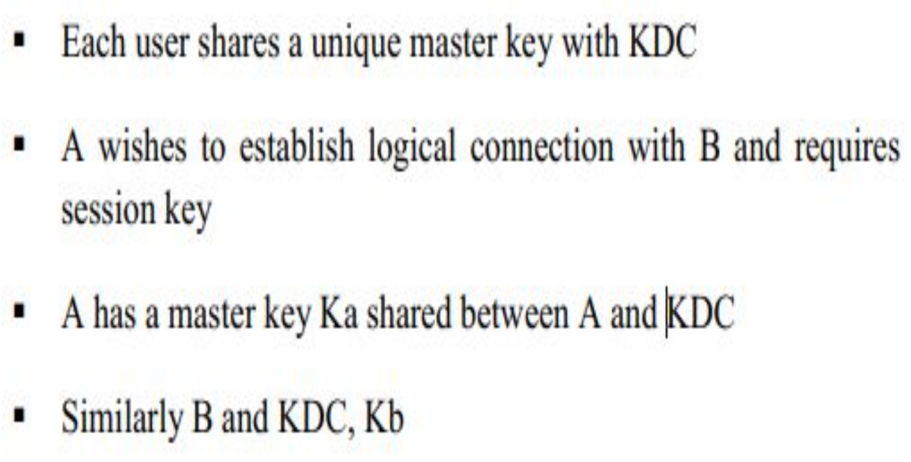
1. A→KDC: $ID_A \,||\, ID_B \,||\, N_1$

2. KDC→A: $E_{Ka}[K_S||ID_B||N_1|| E_{Kb}[K_S||ID_A]]$

3. A→B: $E_{Kb}[K_S||ID_A]$

4. B→A: $E_{KS}[N_2]$

5. A→B: $E_{KS}[f(N_2)]$



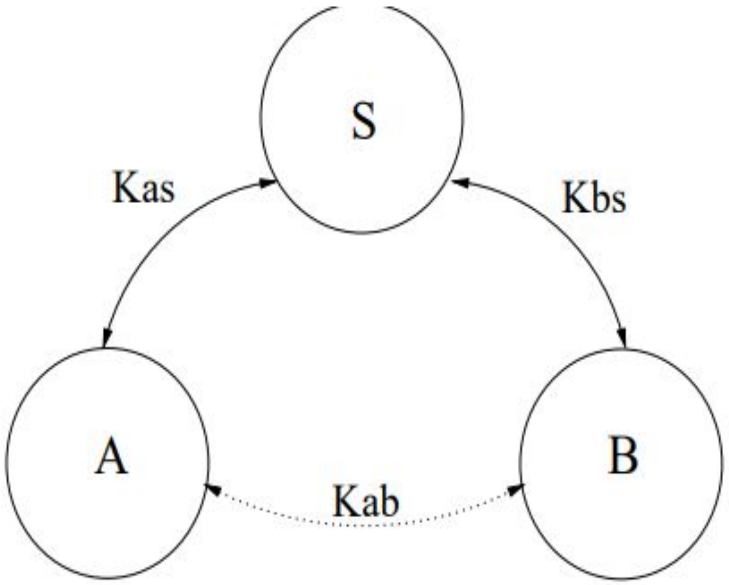
- Each user shares a unique master key with KDC
- A wishes to establish logical connection with B and requires session key
- A has a master key Ka shared between A and KDC
- Similarly B and KDC, Kb

- A issues a request to KDC for session key . Msg includes identity of A and B and a unique identifier N1 for this transaction

- KDC responds->msg encrypted with Ka,only A can read it. Msg also includes
  - One time session key Ks
  - Original request msg,to match this response with appropriate request

- Msg also includes 2 items for B:
  - One time session key Ks
  - Identifier of A,IDa
  - These two items encrypted with Ekb
  - They are to be sent to B to establish the connection and prove A's identity
  - A stores session key and forward to B the information that originated at the KDC for B
  - Session key has been securely delivered to A and B,and they begin their protected exchange.
  - B sends a nonce ,N2 to A
  - Using EKs,A responds with f(N2),f is a function that performs transformation on N2

- There are three principals: A and B, two principals desiring mutual communication, and S, a trusted key server.



It is assumed that A and B already have secure symmetric communication with S using keys K_{as} and K_{bs}, respectively.

Needham-Schroeder Protocol

- Used to securely distribute a new session key for communications between A & B
- But is vulnerable to a replay attack if an old session key has been compromised
  - Then message 3 can be resent convincing B that is communicating with A
- Modifications to address this require:
  - Timestamps
  - Using an extra nonce

Neuman Protocol

- Protocol overview:

1. A → B: $ID_A \,||\, N_a$

2. B → KDC: $ID_B \,||\, N_b \,||\, E_{Kb}[ID_A \,||\, N_a \,||\, T_a]]$

3. KDC → A: $E_{Ka}[\, ID_B \,||\, N_a \,||\, K_S \,||\, T_b \,] \,||$

$E_{Kb}[ID_A \,||\, K_S \,||\, T_b \,]] \,||\, N_b$

4. A → B: $E_{Kb}[ID_A \,||\, K_S \,||\, T_b \,] \,||\, E_{KS}[N_b]$

Using Public-Key Encryption

- Have a range of approaches based on the use of public-key encryption
- Need to ensure, each have correct public keys for other parties
- Using a central authentication server (AS)
- Various protocols exist using timestamps or nonce numbers

Denning AS Protocol

1. A→AS: $ID_A \,||\, ID_B$

2. AS→A: $E_{KRas}[ID_A||KU_a||T] \,||\, E_{KRas}[ID_B||KU_b||T]$

3. A→B: $E_{KRas}[ID_A||KU_a||T] \,||\, E_{KRas}[ID_B||KU_b||T]$

|| $E_{KUb}[E_{KRa}[K_s||T]]$

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

One-Way Authentication

- Required when sender & receiver are not in communications at same time (eg. Email)
- Have header in clear so can be delivered by email system
- May want contents of body protected & sender authenticated

Using Symmetric Encryption

- Can refine use of KDC but can't have final exchange of nonces, vis:
  1. A→KDC: $ID_A \| ID_B \| N_1$
  2. KDC→A: $E_{Ka}[Ks \| ID_B \| N_1 \| E_{Kb}[Ks \| ID_A]]$
  3. A→B: $E_{Kb}[Ks \| ID_A] \| E_{Ks}[M]$
- Does not protect against replays
  - could rely on timestamp in message, though email delays make this problematic

Public-Key Approaches

- Have seen some public-key approaches
- If confidentiality is major concern, can use:

  $A \rightarrow B: E_{KUb}[Ks] \| E_{Ks}[M]$

  - Has encrypted session key, encrypted message
- If authentication needed use a digital signature with a digital certificate:

  $A \rightarrow B: M \| E_{KRa}[H(M)] \| E_{KRas}[T \| ID_A \| KU_a]$

  - With message, signature, certificate

Digital Signature Standard

Digital Signature Standard (DSS)

- NIST published, FIPS 186 known as DSS
- Uses the SHA hash algorithm
- DSS is the standard, DSA is the algorithm
- Creates a 320 bit signature, but with 512-1024 bit security
- Security depends on difficulty of computing discrete logarithms

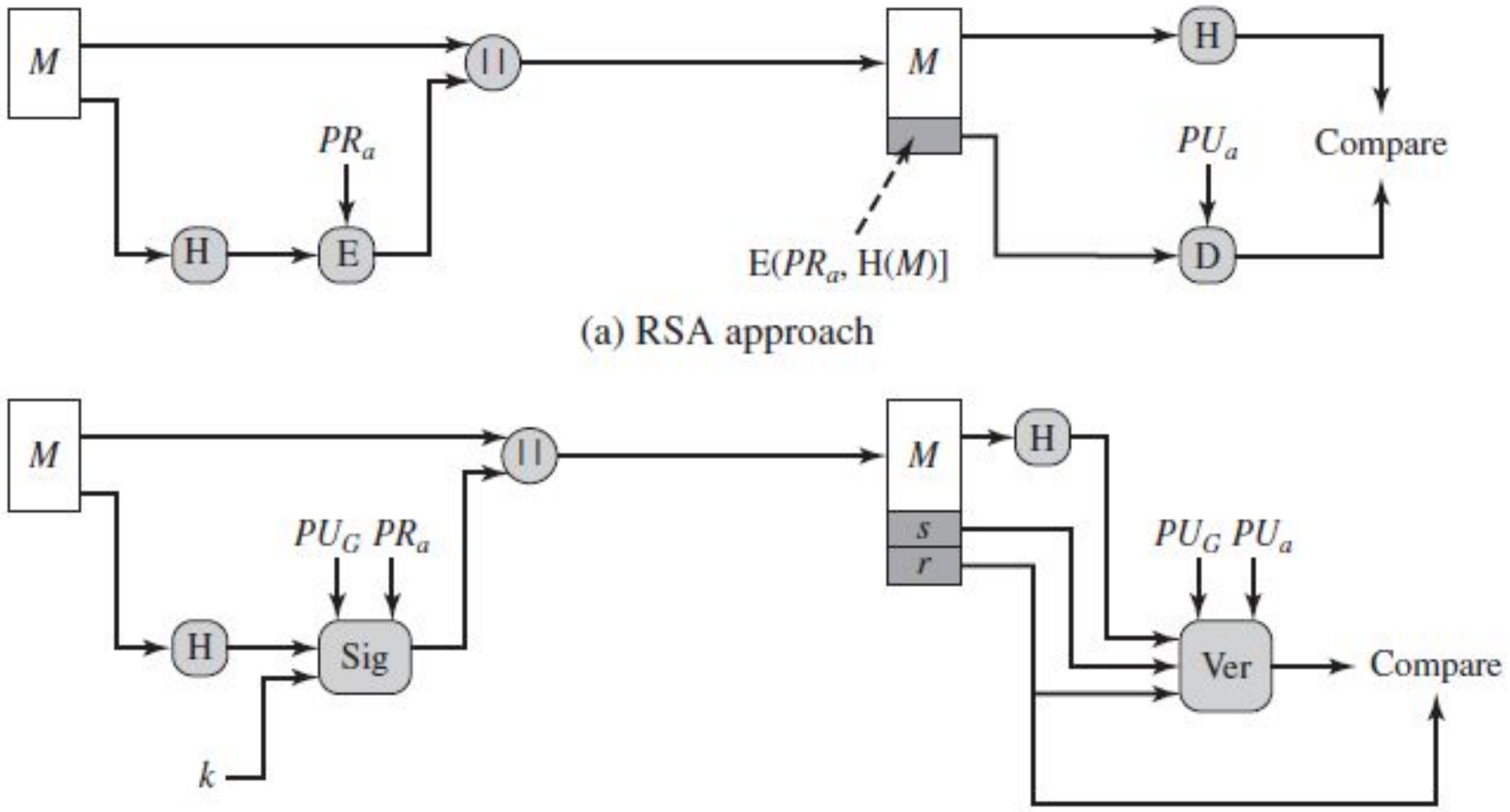
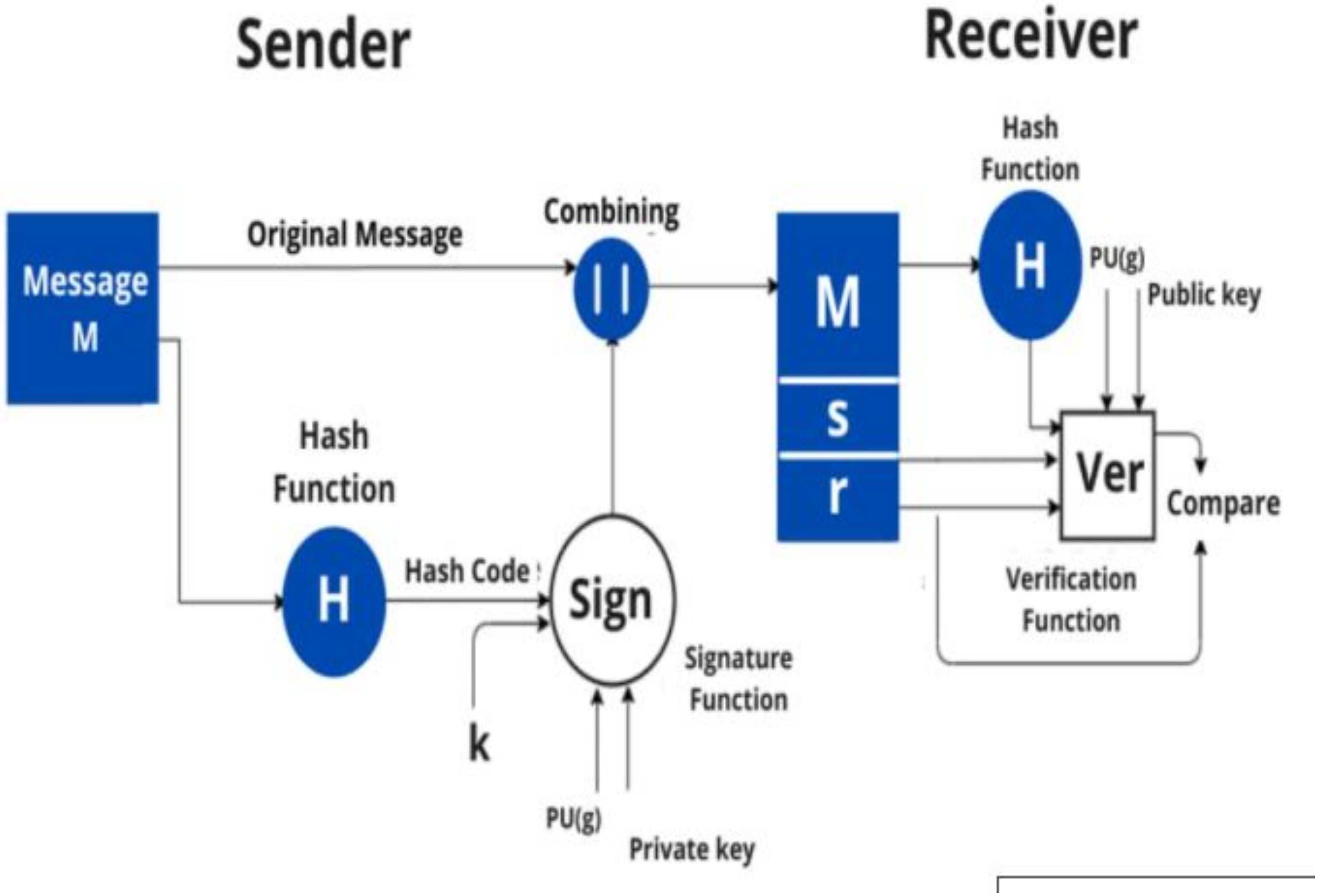


Figure 13.3 Two Approaches to Digital Signatures


Sender
Receiver
Message M
Original Message
Combining
| |
Hash Function
H
Hash Code
Sign
Signature Function
k
PU(g)
Private key
M
s
r
Hash Function
H
PU(g)
Public key
Ver
Compare
Verification Function

- DSS employs SHA (**Secure Hash Algorithm)** to create digital signatures and offers a new digital signature mechanism known as the Digital Signature Algorithm.
- The DSS is different from the fact that the RSA algorithm uses the public key, private key and hash function whereas the DSS uses the public key, private key, hash function, a random number k, and global public key. Therefore, DSS provides more security than RSA algorithms.

- A hash code is generated from the message and given as an input to the signature function on the sender side.
- The other inputs to a signature function include a unique random number k for the signature, the private key of sender PR(a), and global public key i.e., PU(g).
- The output of the signature function consists of two components: s & r, which is concatenated with the input message and then sent to the receiver.
  Signature = {s, r}.

- At the receiver side, the hash code for the message sent is generated by the receiver by applying a hash function.
- The verification function is used for verifying the message and signature sent by the sender.
- The verification function takes hash code generated, signature components s and r, the public key of the sender (PU(a)), and global public key.
-

  The signature function is compared with the output of the verification function and if both the values match, the signature is valid because A valid signature can only be generated by the sender using its private key.

DSA Key Generation

- Have shared global public key values (p, q, g):

p a large prime number where $2^{L-1} < p < 2^{L}$ for L= 512 to 1024 bits and is a multiple of 64

q a 160 bit prime divisor of p-1

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

where $1 < h < p-1$, such that $g > 1$

DSA Key Generation

- Users choose private & compute public key:
  - choose private key x with $0<x<q$
  - compute public key $y = g^x \bmod p$
- User's Per-Message Secret Number
  - generates a random integer k, with $0<k<q$
    - Note: k must be random, be destroyed after use, and never be reused

DSA Signature Creation

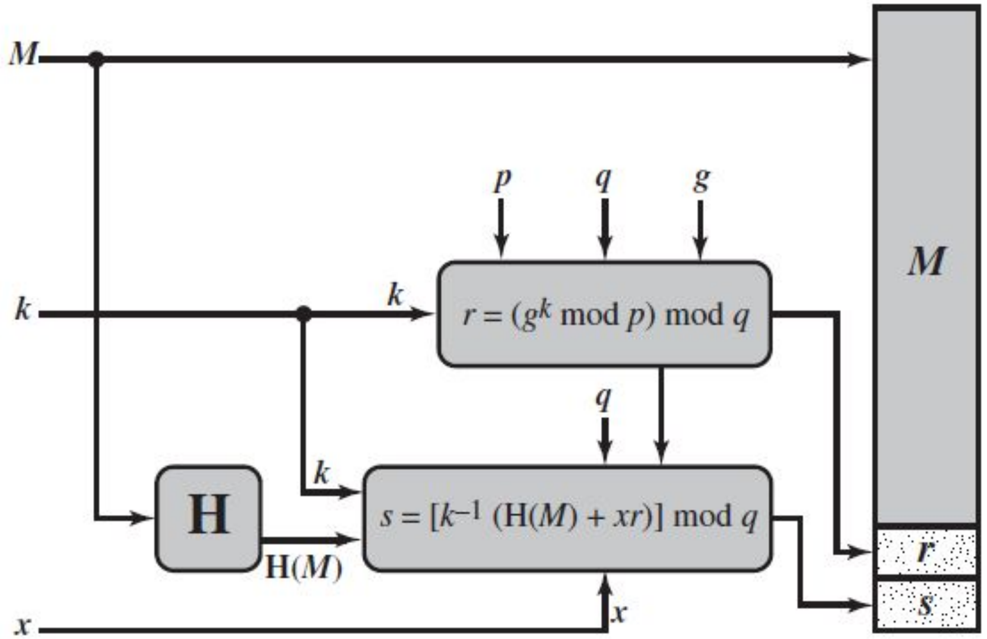
- Computes signature pair:

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

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

- Sends signature (r, s) with message M

DSA Signature Creation



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

(a) Signing

DSA Signature Verification

- Having received M & signature (r, s)
  - Consider it as M′ and (r′, s′)
- To **verify** a signature, recipient computes:

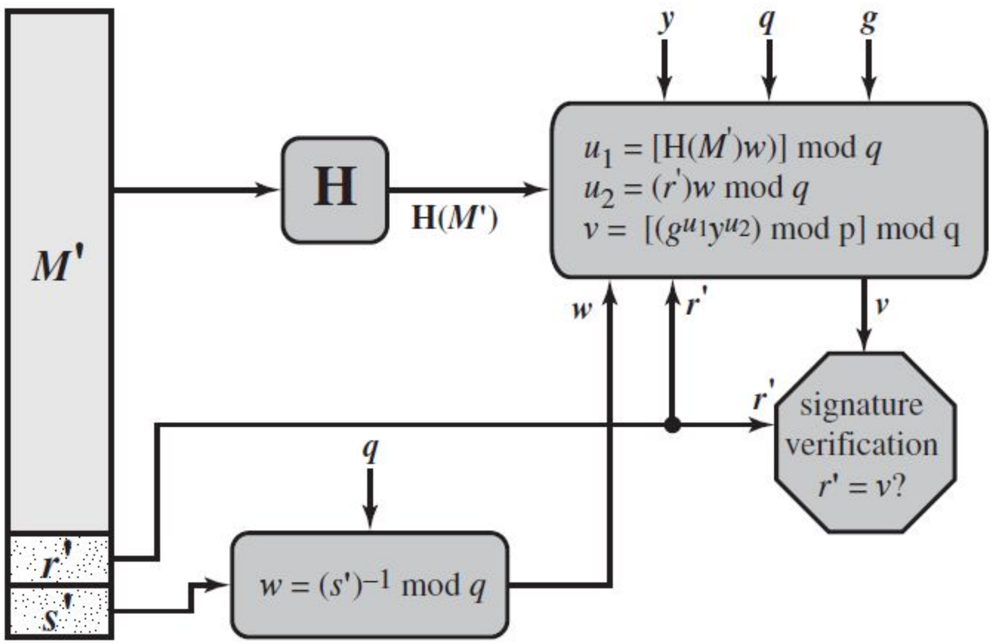
  $w = (s')^{-1} \bmod q$

  $u1 = (H(M').w) \bmod q$

  $u2 = (r'.w) \bmod q$

  $v = [(g^{u1}.y^{u2}) \bmod p] \bmod q$
- if $v = r'$ then signature is verified

DSA Signature Verification



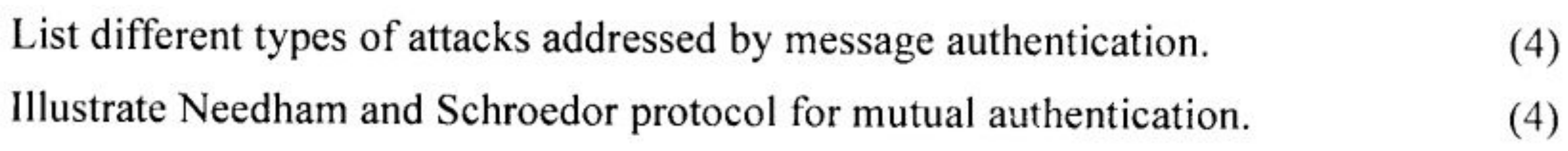
(b) Verifying

Figure 13.5 DSA Signing and Verifying

Summary

- have considered:
  - digital signatures
  - authentication protocols (mutual & one-way)
  - digital signature standard

Questions



List different types of attacks addressed by message authentication. (4)

Illustrate Needham and Schroedor protocol for mutual authentication. (4)