Cryptographic Hash Function

CCS6314 Lecture 10

Outlines

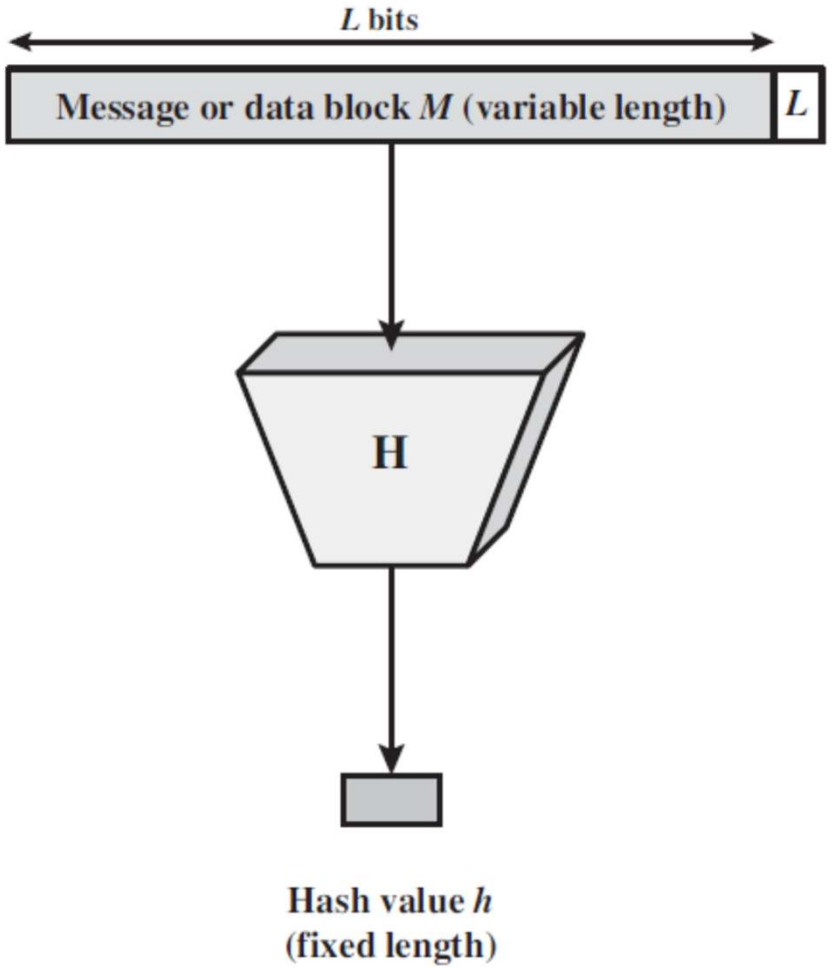
1. Hash Function and Cryptographic Hash Function.
2. Applications of Hash Function.
3. Cryptanalysis on Hash Function.
4. Types of Practical Hash Functions.

Learning Objectives

In the end of this Chapter 10, students will be able:

1. To explain the **hash function** and **its importance** in cryptography.
2. To explain the properties and requirements of **cryptographic hash function**.
3. To explain the **applications** of cryptographic hash function.
4. To explain the **cryptanalysis** on hash function.
5. To explain some **practical cryptographic hash functions**.

HASH FUNCTION
&
CRYPTOGRAPHIC HASH FUNCTION



Hash Function

- A **hash function** is a function $H(\cdot)$ that accepts a variable-length block of data M as input and produces a fixed-size hash value $h = H(M)$.
- The **principle objective** of hash function is **to ensure data integrity**. That is, to check whether a data has been changed/modified/altered.
- A good hash function possesses a property such that when applying $H(\cdot)$ to a large set of inputs, it should produces **evenly distributed** and **random outputs**.
- This function has many applications including **message authentication**, **digital signature**, strong one-way passwords, and intrusion/virus detection.

Cryptographic Hash Function

- A cryptographic hash function is a hash function needed for security applications.
- This hash function should be **computationally infeasible** for an potential attack to violate either:
  1. **Pre-Image Resistance** (One-Wayness): To find a data object that maps to a pre-defined hash result. That is, a M that maps to a known h.
  2. **Collision Resistance** (Collision-Free): To find two data objects that map to the same hash result. That is, M_1 and M_2 that produce the same h.

Cryptographic Hash Function - Requirements

1. Pre-Image Resistance

- The pre-image of a hashed value h is the value x such that $h = H(x)$.
- Since a hash function $H(\cdot)$ is a **many-to-one mapping**, there exists multiple pre-images x that produce the same hash value $h = H(x)$.
- For a hash function $H(\cdot)$ to be **cryptographically secure**, it has to satisfy the pre-image resistance such that given any hash value h, it is **computationally infeasible to find** an x such that $h = H(x)$.

Cryptographic Hash Function - Requirements

2. Collision Resistance

- The collision of a hash function $H(\cdot)$ is the condition where given $x \neq y$, we have $H(x) = H(y)$.
- Collision resistance is considered in a **strong collision case** in cryptographic hash function.
- For a hash function $H(\cdot)$ to be **cryptographically secure**, it has to satisfy the collision resistance such that it is **computationally infeasible to find** any pair of x and y such that $H(x) = H(y)$.

Cryptographic Hash Function - Requirements

3. Second Pre-image Resistance

- The second pre-image of a hash function $H(\cdot)$ is the condition where given any x and $H(x)$, there exists a value $y \neq x$ such that $H(y) = H(x)$.
- Second pre-image resistance is considered in a **weak collision case** in cryptographic hash function.
- For a hash function $H(\cdot)$ to be **cryptographically secure**, it has to satisfy the second pre-image resistance such that given any x and $H(x)$, it is **computationally infeasible to find** a $y \neq x$ such that $H(y) = H(x)$.

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	$H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that $H(y) = h$.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.
Pseudorandomness	Output of H meets standard tests for pseudorandomness.

APPLICATIONS OF HASH FUNCTION:

MESSAGE AUTHENTICATION

Message Authentication

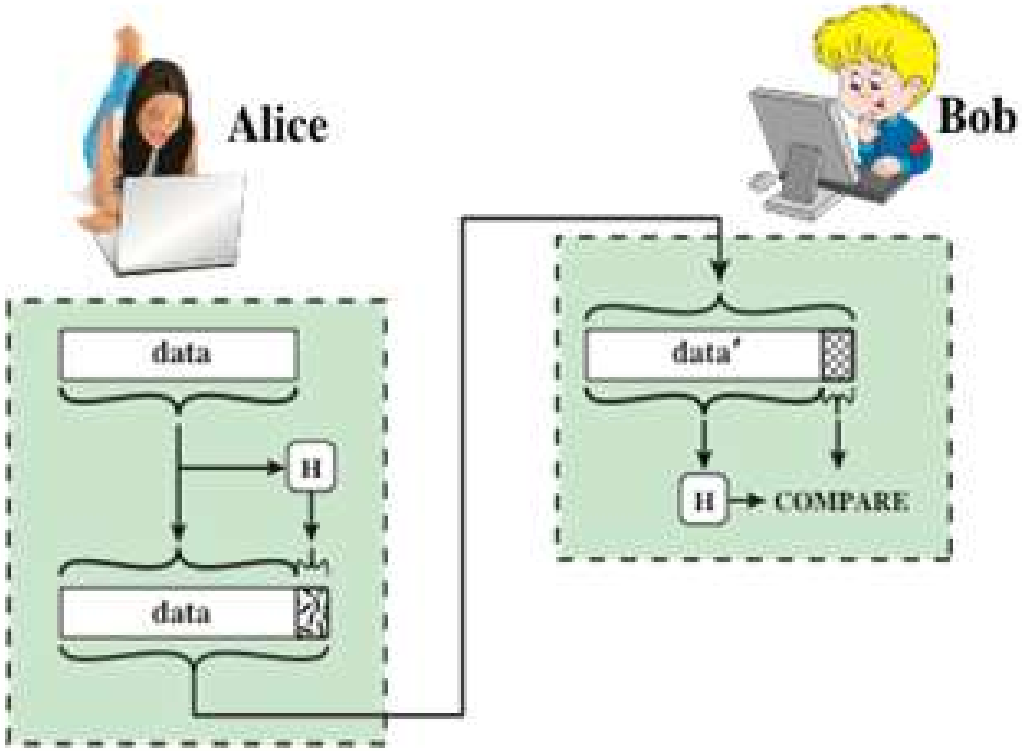
- **Message authentication** is a mechanism or service used to verify the integrity of a message.
  1. It ensures that the data received are exactly as sent (no modification, insertion, deletion, or replay).
  2. It assures that the identify of the sender is valid/authentic.
- When a hash function is used to provide message authentication, the hash function value is called as a **message digest** .

Message Authentication

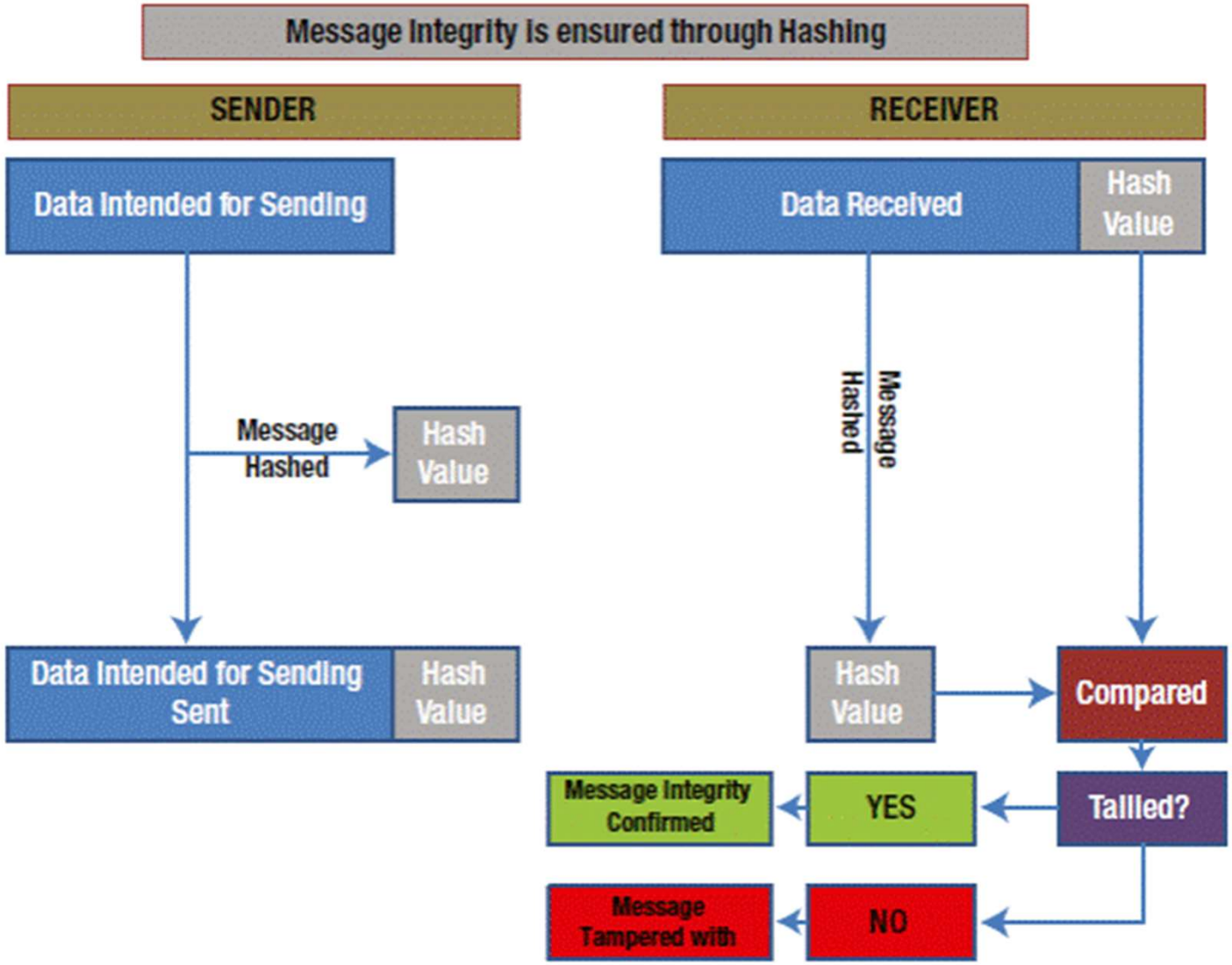
- Using hash function for data integrity check.

1. The sender **computes a hash value h** as a function of the bits in the message M and transmits both the hash value and the message (h, M) (usually in the **concatenated form** $(h \parallel M)$).
2. The receiver **performs the same hash calculation** on the message bits and compares this value with the incoming hash value.
3. If there is a mismatch, the receiver knows that the message (or possibly the hash value) has been altered.

Message Authentication



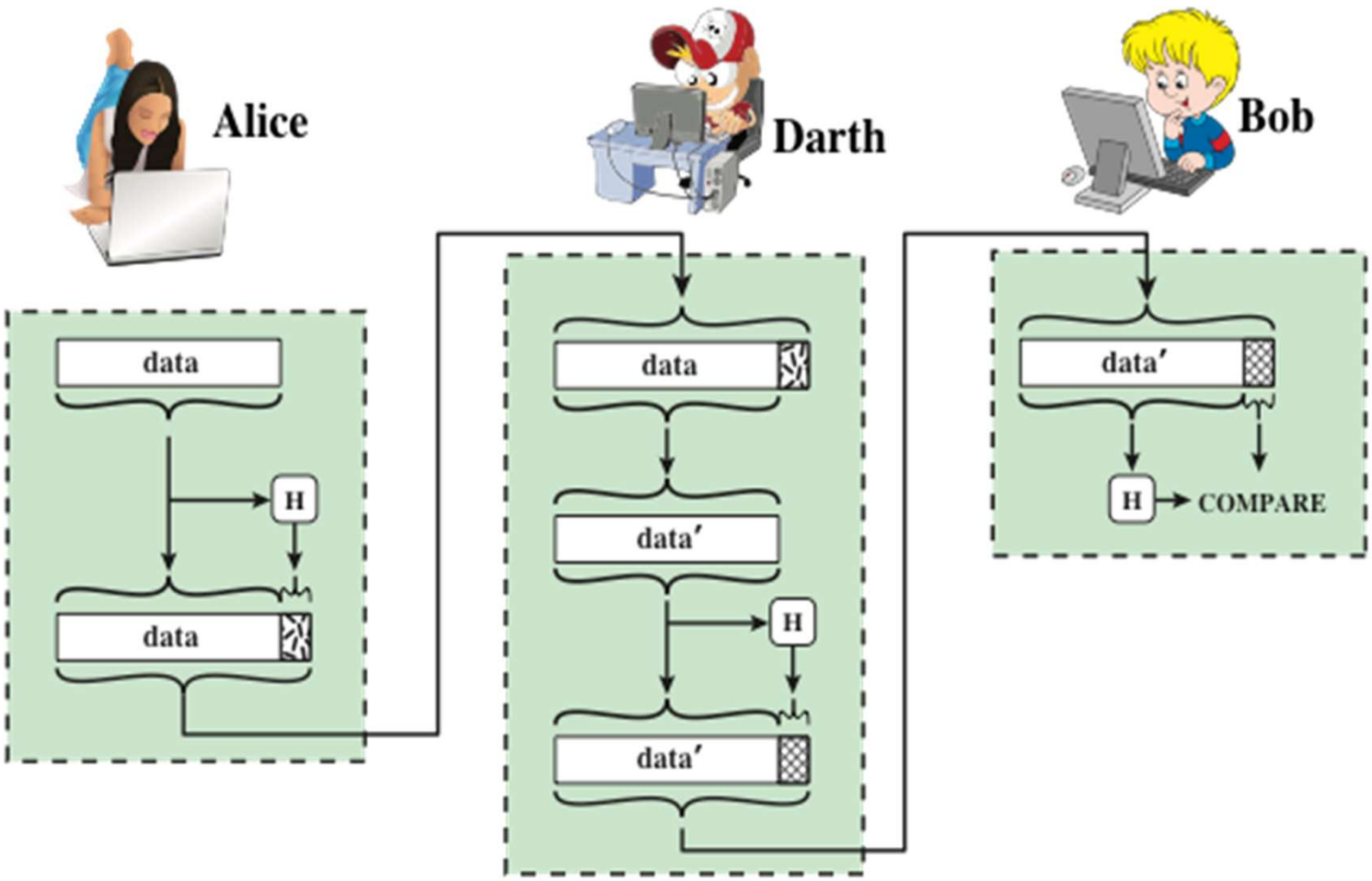
(a) Use of hash function to check data integrity


Message Integrity is ensured through Hashing
SENDER
RECEIVER
Data Intended for Sending
Data Received
Hash Value
Message Hashed
Hash Value
Message Hashed
Data Intended for Sending Sent
Hash Value
Hash Value
Compared
Tallied?
YES
Message Integrity Confirmed
NO
Message Tampered with

Message Authentication – Security Issue

- The default data integrity check suffers from the simple **Man-in-the-Middle attack**, as both the message M and its message digest $h = H(M)$ are transmitted through the channel.

1. When Alice transmitted (h, M), Darth (the adversary) intercepts the pair.
2. Darth next modifies the message M' and computes $h' = H(M')$. He next **replaces the whole pair** (h, M) with (h', M') and transmits it to Bob.
3. Bob, without knowing the original message and its digest, will always verify the correctness of the received pair successfully.



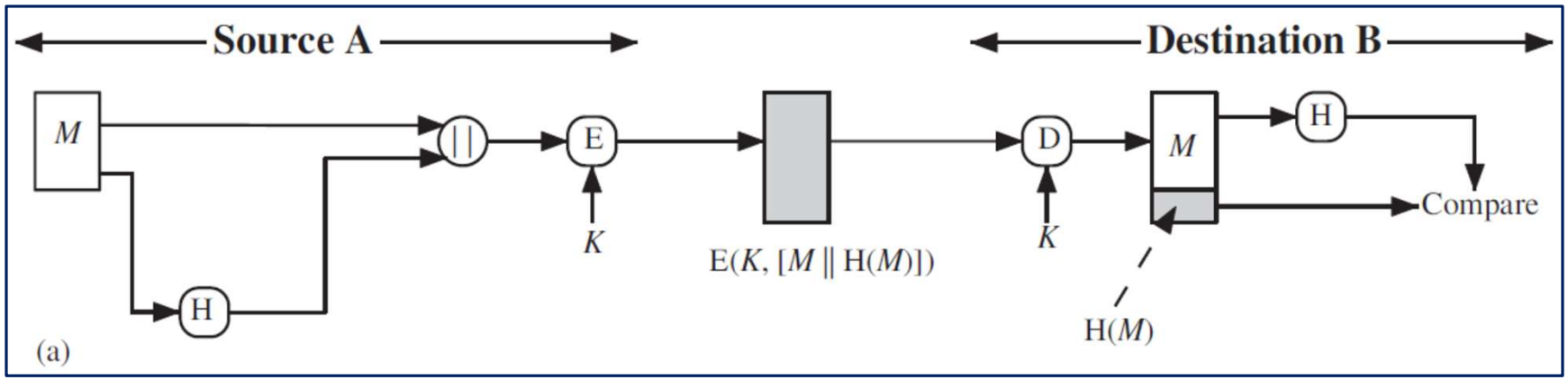
(b) Man-in-the-middle attack

Secure Message Authentication

- Hence, there is a need to provide alternative ways of ensuring a secure message authentication through hash function.
- In the next few slides, we discuss and observe **the four different alternatives** of achieving secure message authentication using hash function.

Secure Message Authentication – Alternative 1

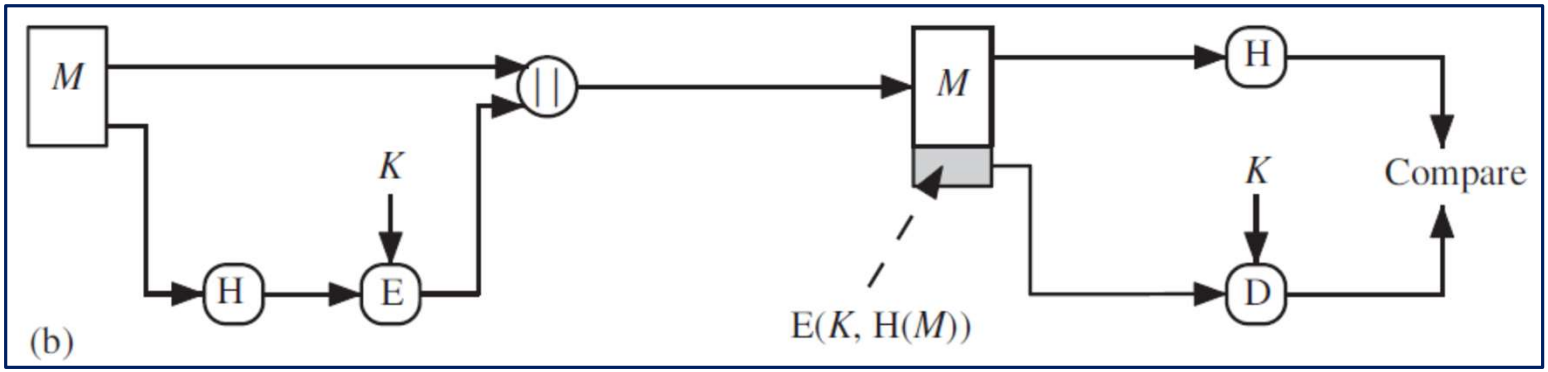
- **The message plus concatenated hash code is encrypted using symmetric encryption**.
- Since only A and B has the secret key, the message must be from A and has not been altered.
- The hash code ensures the **message authentication**.
- **Confidentiality is also provided** since the encryption is applied to the entire message plus hash code.



1. A on the message M computes $h = H(M)$ and concatenates them as $(M \parallel h)$.
2. A next encrypts $(M \parallel h)$ using symmetric key K, $C = \mathrm{Enc}\big(K, (M \parallel h)\big)$ and sends the ciphertext C to B.
3. B when receiving ciphertext C, decrypts it using the same symmetric key K to recover the concatenation $(M \parallel h) = \mathrm{Dec}(K, C)$.
4. B then extract M from $(M \parallel h)$ and computes $h' = H(M)$. B compares h' with the h he extracted. If it tallies, then B accepts them as authentic message and signature. Otherwise, B rejects the ciphertext.

Secure Message Authentication – Alternative 2

- **Only the hash code is encrypted, using symmetric encryption**.
- This reduces the processing burden for those applications that **do not require confidentiality**.

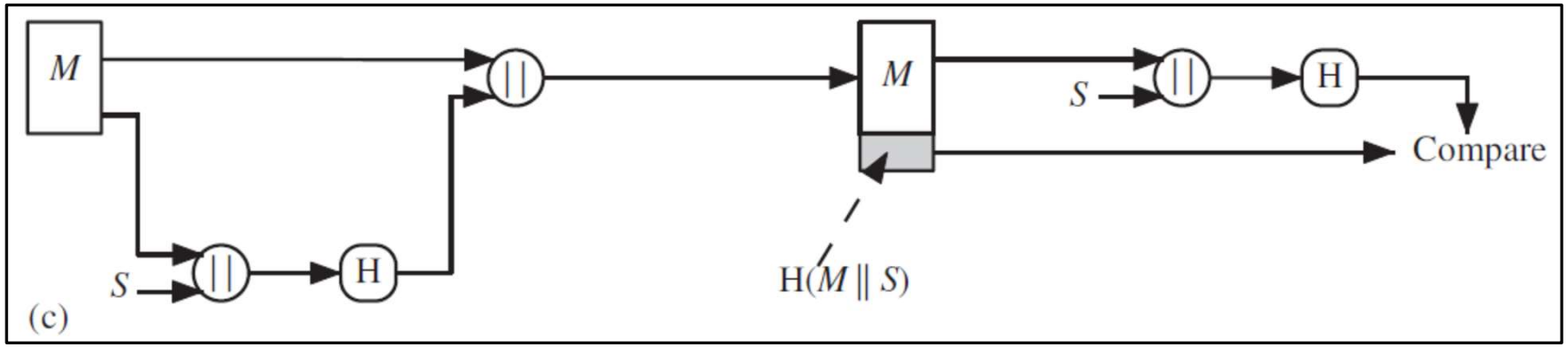


Secure Message Authentication – Alternative 2

1. A on the message M computes $h = H(M)$ and encrypts h using symmetric key K, $C = \mathrm{Enc}(K, h)$.
2. A next concatenates the message with the ciphertext $(M \parallel C)$ and sends it to B.
3. B when receiving $(M \parallel C)$, extract ciphertext C from $(M \parallel C)$ and decrypt the it using the same symmetric key K to recover the message digest $h = \mathrm{Dec}(K, C)$.
4. B then computes $h' = H(M)$ and compares h' with the h he decrypted. If it tallies, then B accepts them as authentic message and signature. Otherwise, B rejects them.

Secure Message Authentication – Alternative 3

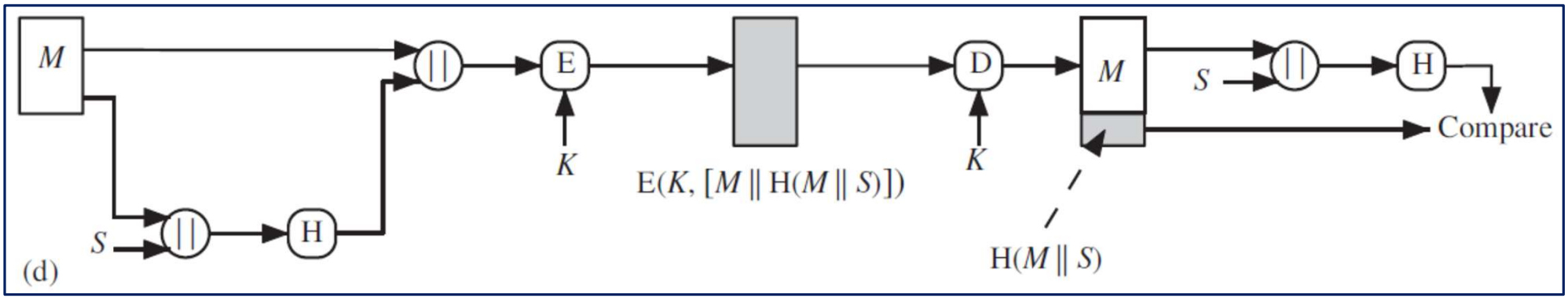
- **Use of Hash Function without encryption for message authentication**.
- This assumes the two communicating parties **share a common secret value S**.
- A computes the hash value over the concatenation of M and S and appends the resulting hash value to M.
- Since B possesses S, it can verify its authenticity.
- As the secret value S itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.



1. A on the message M, concatenates it with the secret value S and computes $h = H(M \parallel S)$.
2. A next concatenates message M with the computed message digest and sends $(M \parallel h)$ to B.
3. B when receiving $(M \parallel h)$, extract M from $(M \parallel h)$ and concatenates it with the same secret value S.
4. B then computes $h' = H(M \parallel S)$ and compares h' with the h he extracted. If it tallies, then B accepts them as authentic message and signature. Otherwise, B rejects them.

Secure Message Authentication – Alternative 4

- Confidentiality can be added to the Alternative 3 by encrypting the entire message plus the hash code.



Secure Message Authentication – Alternative 4

1. A on the message M, concatenates it with the secret value S and computes $h = H(M \parallel S)$.
2. A next concatenates message M with the computed message digest and encrypts it using symmetric key K, $C = \mathrm{Enc}\big(K, (M \parallel h)\big)$.
3. A finally sends the ciphertext C to B.
4. B when receiving ciphertext C, decrypts it using the same symmetric key K, $(M \parallel h) = \mathrm{Enc}(K, C)$.
5. B next extracts M from $(M \parallel h)$ and concatenates it with the same secret value S.
6. B then computes $h' = H(M \parallel S)$ and compares h' with the h he extracted. If it tallies, then B accepts them as authentic message and signature. Otherwise, B rejects them.

Authentication vs Encryption

- In some applications, where **confidentiality is not required**, it is desirable to **avoid encryption** when performing authentication.

1. Encryption software is **relatively slow**. Even though the amount of data to be encrypted per message is small, there may be a steady stream of messages into and out of a system.
2. Encryption hardware **costs are not negligible**. Low-cost chip implementations of DES are available, but the cost adds up if all nodes in a network must have this capability.

Authentication vs Encryption

3. Encryption hardware is **optimized toward large data sizes**. For small blocks of data, a high proportion of the time is spent in initialization/invocation overhead.
4. Encryption algorithms may be **covered by patents**, and there is a cost associated with licensing their use.

APPLICATIONS OF HASH FUNCTION:

MESSAGE AUTHENTICATION CODE (MAC)

Message Authentication Code (MAC)

- Also known as **keyed hash function**, it is typically used between two parties that share a secret key to authenticate information exchanged between those parties.
- The algorithm takes in a **secret key** and a **data block** as inputs and produces MAC which is associated with the protected message.
- If the integrity of the message needs to be checked, the MAC function can be applied to the message and the result **compared** with the associated MAC value.
- An attacker who alters the message will be unable to alter the associated MAC value without the knowledge of the **secret key**.

APPLICATIONS OF HASH FUNCTION:

DIGITAL SIGNATURE

Digital Signature

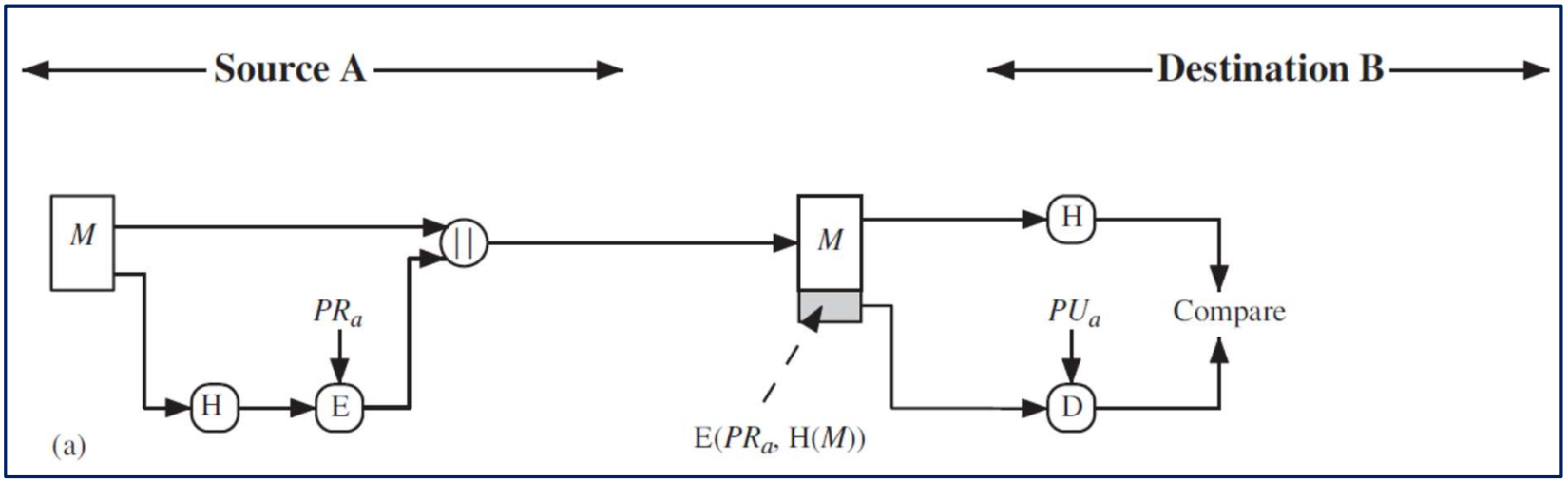
- A scheme utilizes **asymmetric-key** (**public-key**) cryptography to provide message authentication.
- The **aim** of digital signature is to **prove** to anyone that a message is originated from a particular user (signer).
- Digital signature is **NOT possible** via **symmetric-key** cryptography:
  1. Two users, A (sender) and B (receiver) share a secret key K.
  2. Receiver B of the message can verify that the message came from sender A.
  3. Another user C however cannot prove that message came from sender A. It might have come from receiver B also.

Digital Signature

- The operation of digital signature is similar to message authentication code (MAC).
- The hash value of a message, $h = H(M)$ is encrypted with a signer's private key to produce a signature, $\sigma = \mathrm{Sig}(K_{\mathrm{Prv}}, h)$.
- Anyone who knows the signer's public key can verify the integrity of the message associated with the digital signature.
- An attacker who wishes to **alter the message** (or **forge the signature**) would need to know the signer's private key.

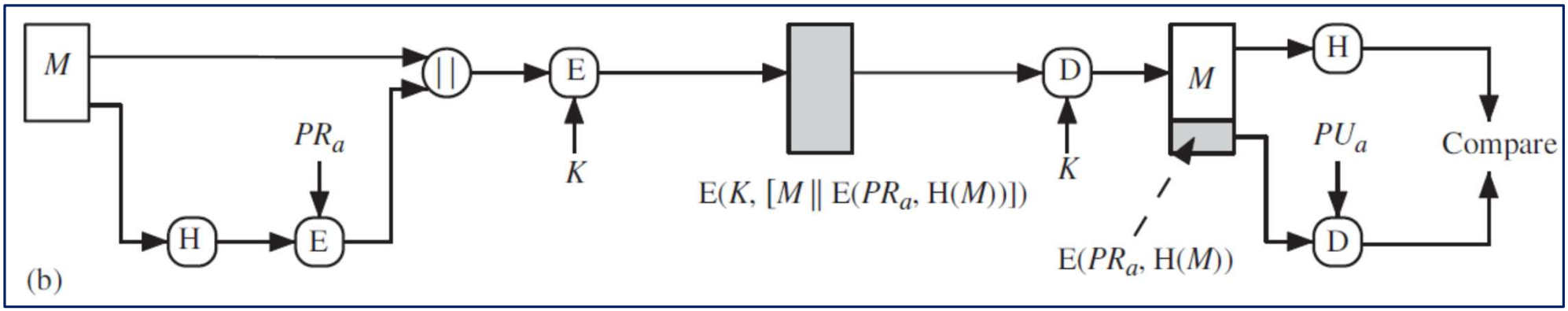
Digital Signature – First Approach

- The hash code is encrypted, using public-key encryption with the sender's private key.
- This provides **authentication** and a **digital signature**, because only the sender could have produced the encrypted hash code.



Digital Signature – Second Approach

- If **confidentiality** and a **digital signature** is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key.



APPLICATIONS OF HASH FUNCTION:

OTHER APPLICATIONS

Other Application of Hash Functions

1. **Creation of a one-way password file**.
   - When a user enters a password, the hash of that password is compared to the stored hash value for verification.
   - This approach to password protection is used by most operating systems.

Other Application of Hash Functions

2. **Intrusion and virus detection**.
   - Store $H(\text{File})$ for each file on a system and secure the hash values.
   - One can later determine if a file has been modified by recomputing $H(\text{File})$.
   - An intruder would need to change File without changing $H(\text{File})$.

3. **Construction of a pseudorandom function (PRF) or a pseudorandom number generator (PRNG)** .
   - A common application for a hash-based PRF to generate the symmetric keys.

Application of Hash Functions – Requirements

	Preimage Resistant	Second Preimage Resistant	Collision Resistant
Hash + digital signature	yes	yes	yes*
Intrusion detection and virus detection		yes	
Hash + symmetric encryption			
One-way password file	yes		
MAC	yes	yes	yes*

* Resistance required if attacker is able to mount a chosen message attack

ATTACKS & CRYPTANALYSIS ON HASH FUNCTION

Brute-Force Attacks on Hash Function

- The attack that does not depend on the specific algorithm, instead only depends on bit length.

1. **Pre-image and Second Pre-image Attacks**:
   - For a hash function, the brute force attack depends only on the **bit length** of the hash value h.
   - The common technique is to pick any random values x' and try computing each hash value $h = H(x')$ until a **collision** occurs.
   - For a n-bit hash value, the adversary **on average** has to try 2^{n-1} values of x' to such a collision.

Brute-Force Attacks on Hash Function

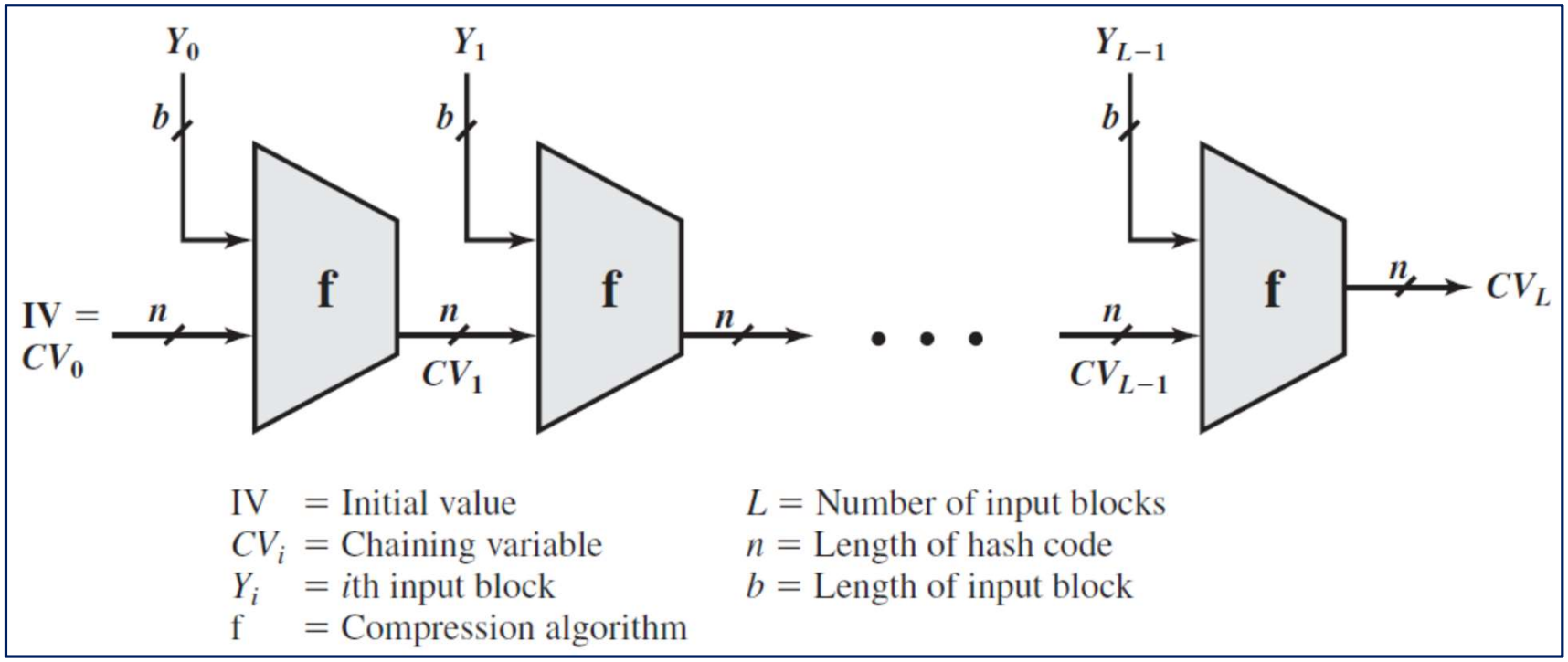
2. Collision Resistant Attacks:

- For a collision resistant attack, an adversary wishes to find two messages or data blocks x and y with $x \neq y$ that yield the same hash function $H(x) = H(y)$.
- The effort required is explained by a mathematical result referred to as the ***birthday paradox***.
- In essence, if we choose random variables from a uniform distribution in the range of $0 \leq x \leq N - 1$, then the probability that a repeated element is encountered exceeds 0.5 after $\sqrt{N}$ choices have been made.
- For a n-bit hash value, the **collision is expected to occur** within $2^{\frac{n}{2}}$ attempts.

Cryptanalysis on Hash Function

- An attack based on weaknesses in a particular cryptographic algorithm.
- Cryptanalysis on hash function seeks to **exploit some properties of the algorithm** to perform some attacks other than an exhaustive search.
- To understand how cryptanalysis works, we need to **understand how the internal structures of certain hash function**. In this case, we consider the general structure of **secure hash code,** an iterated hash function proposed by **Ralph Merkle in 1989**.

Cryptanalysis on Secure Hash Code



Cryptanalysis on Secure Hash Code

- 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. Either the opponent must find two messages of equal length that hash to the same value or two messages of differing lengths that, together with their length values, hash to the same value.

Cryptanalysis on Secure Hash Code

- The hash algorithm involves repeated use of a **compression function f**, that takes two inputs (an n-bit input from the previous step, called the chaining variable, and a b-bit block) and produces an n-bit output.
- At the start of hashing, the chaining variable has an initial value that is specified as part of the algorithm.
- The final value of the chaining variable is the hash value.

Cryptanalysis on Hash Function Secure Hash Code

- The motivation for this iterative structure is that if the compression function is collision resistant, then so is the resultant iterated hash function.
- The problem of designing a secure hash function **reduces to that of designing a collision-resistant compression function** that operates on inputs of some fixed size.
- The cryptanalysis is then focuses on the **internal structure of f** and is on attempts to find efficient techniques for producing collisions for a single execution of f. After that, the attack must consider the fixed value of IV.

TYPES OF
PRACTICAL
HASH FUNCTIONS

Practical Hash Function Based on Cipher Block Chaining (CBC)

- Several proposals have been made for hash functions based on using a cipher block chaining (CBC) technique, but without using the secret key.
- One of the first proposals was due to Rabin that divide a message M into fixed-size blocks $M_1, M_2, \ldots, M_N$ and use a symmetric encryption system such as DES to compute the hash code G as

$$H_0 = \text{Initial Value}$$
$$H_1 = \text{Enc}(M_i, H_{i-1})$$
$$G = H_N$$

Practical Hash Function Based on Cipher Block Chaining (CBC)

- The operation is similar to the CBC technique, but there is no secret key involved here.
- As with any hash code, this scheme is subject to the **birthday attack**. If the encryption algorithm is DES and only a 64-bit hash code is produced, the system is **vulnerable to meet-in-the-middle attack**, for instance.
  - Another version of the birthday attack used even if the opponent has access to only one message and its valid signature and cannot obtain multiple signings.

Practical Hash Function Based on Cipher Block Chaining (CBC)

- It can be shown that some form of birthday attack will succeed against any hash scheme involving the use of cipher block chaining without a secret key, provided that either the resulting hash code is small enough or that a larger hash code can be decomposed into independent subcodes.

Practical Hash Function – Message Digest Algorithm 5 (MD5)

- The Message Digest Algorithm 5 (MD5) was developed by Ron Rivest in 1991 that generates 128-bit hash values.
- It was standardised by IETF in RFC1321 and commonly used to authenticate files
- It is **no longer recommended to use** since collision and other attacks possible.
- However, if one simply **copying a file from one place to another**, MD5 will still do the job.

Practical Hash Function – Secure Hash Algorithm (SHA)

- **Secure Hash Algorithm (SHA)** was originally designed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993. It was then revised as SHA-1 in 1995.
- SHA is designed based on the **hash function MD4** and produces **160-bit hash values**.
- In 2002 NIST produced a revised version of the standard that defined three new versions of SHA with **hash value lengths of 256, 384, and 512**, collectively known as **SHA-2**.

Practical Hash Function – Secure Hash Algorithm (SHA)

Table 11.3 Comparison of SHA Parameters

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Note: All sizes are measured in bits.

Practical Hash Function – Secure Hash Algorithm (SHA)

- The real practical SHA algorithm is very complex and beyond our syllabus, hence we will not cover them here.
- If you are interested to learn in details on how SHA-512 works, you can refer to Stallings' book, Page 343 – 352, SHA-512 Logic for its elegant design and architecture.

Main References for Chapter 10

1. Stallings, W. "***Cryptography and Network Security: Principles and Practices***", Fifth Edition, Pearson Prentice Hall, 2011.

~ THE END ☺ ~