Hash Function

PREPARED BY: DR. REEMA PATEL

Message Integrity

- The cryptography systems that we have studied so far provide secrecy, or confidentiality, but not integrity.

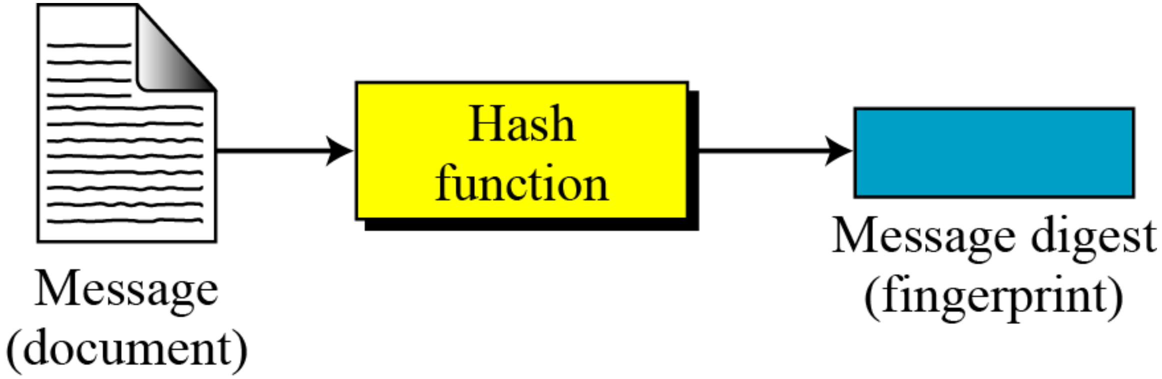
- However, there are occasions where we may not even need secrecy but instead must have integrity.

Document and Fingerprint

- One way to preserve the integrity of a document is through the use of a fingerprint.
- Alice needs to be sure that the contents of her document will not be changed, she can put her fingerprint at the bottom of the document

Message and Message Digest

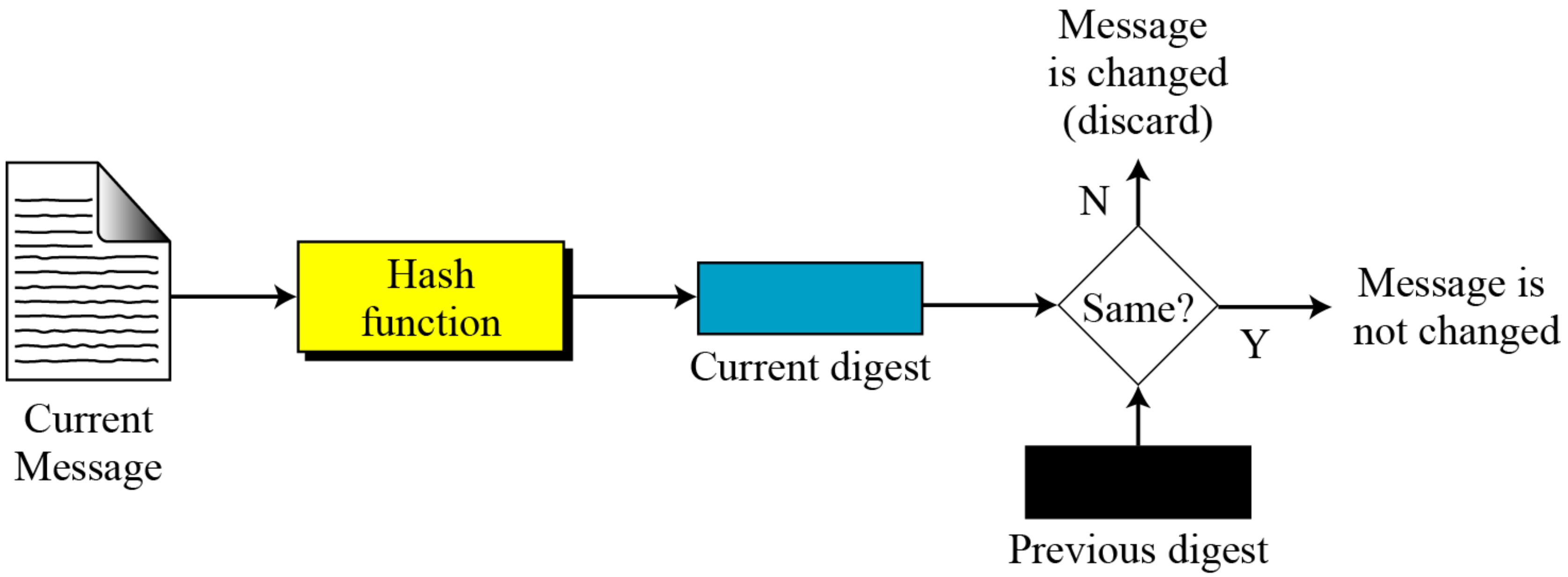
- The electronic equivalent of the document and fingerprint pair is the message and digest pair.



Difference

- The two pairs (document / fingerprint) and (message / message digest) are similar, with some differences.
- The document and fingerprint are physically linked together. The message and message digest can be unlinked separately, and, most importantly, **the message digest needs to be safe from change.**

Checking Integrity



Hash Function

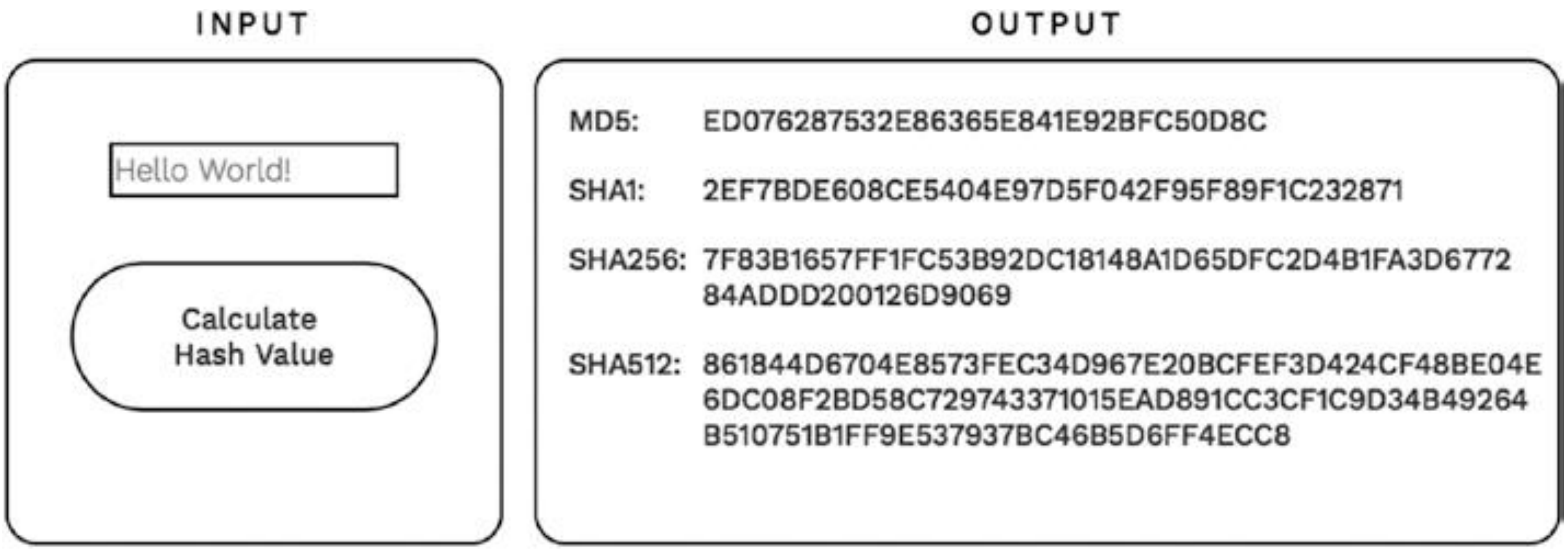
- Hash Function: Takes Message as input and produces a fixed length output.
- Hash functions are functions that compress an input of arbitrary length to a result with a fixed length.

- h = H(M) :- M - Arbitrary length message, h- Fixed length hash code
- An n–bit hash is a map from arbitrary length message to n-bit hash value.
- n-bit hash value referred as a hash-value, hash-code, hash-result, message digest, digital fingerprint or simply hash.
- Hash Functions are used for data integrity and authentication.

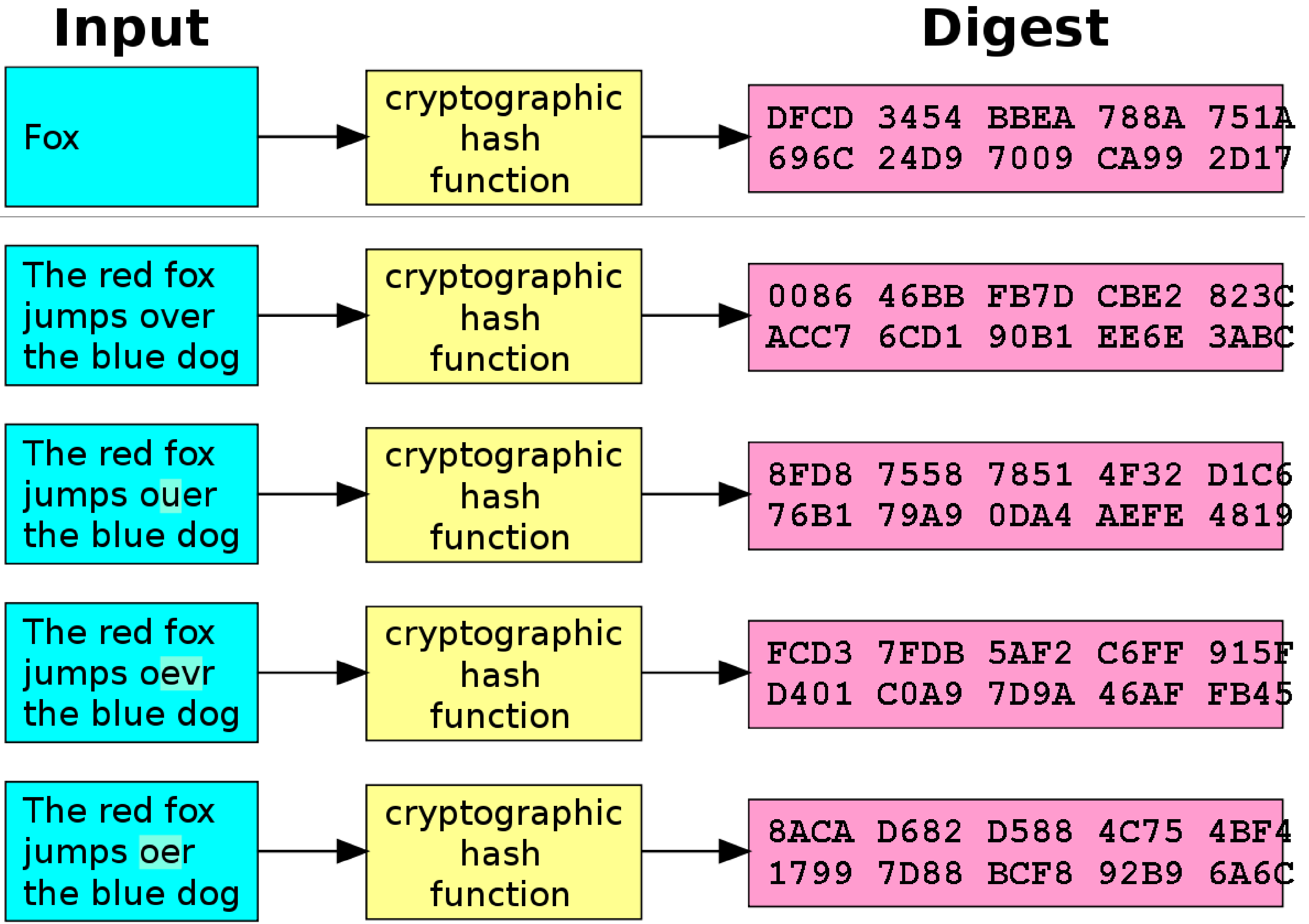
Hash Function

- Hash function:
  - Takes message as input and produces an output referred to as a hash-code, hash-result, hash-value, or simply hash.
  - Maps bit strings of arbitrary finite length to strings of fixed length, say n bits.
  - hash value serves as a compact representative image (called imprint, digital fingerprint, or message digest) of input string.
  - Hash functions are used for data integrity and message authentication.

How It Works


INPUT
Hello World!
Calculate
Hash Value
OUTPUT
MD5: ED076287532E86365E841E92BFC50D8C
SHA1: 2EF7BDE608CE5404E97D5F042F95F89F1C232871
SHA256: 7F83B1657FF1FC53B92DC18148A1D65DFC2D4B1FA3D6772
84ADDD200126D9069
SHA512: 861844D6704E8573FEC34D967E20BCFEF3D424CF48BE04E
6DC08F2BD58C729743371015EAD891CC3CF1C9D34B49264
B510751B1FF9E537937BC46B5D6FF4ECC8

- A small change in the data results in a significant change in the output – called avalanche effect



Hash Function

- Easy to compute
- Almost impossible to reverse
- Security properties:
  - Collision-resistant
  - Hides the original string
  - Almost impossible to get the original string from the output
  - Puzzle friendly

Requirements of Cryptographic Hash Function

- Cryptographic hash functions have the following properties:
  - Providing hash values for any kind of data quickly
  - Being deterministic
  - Being pseudorandom
  - Being one-way functions (preimage resistant)
  - Being collision resistant

Requirements of Cryptographic Hash Function

- **Deterministic**
  - hash function yields identical hash values for identical input data.

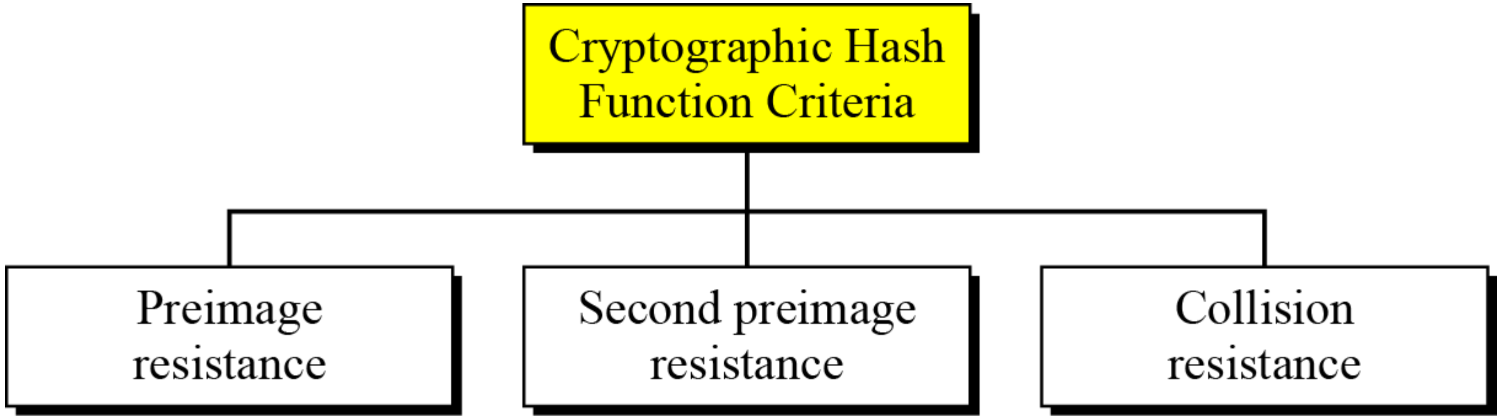
- **Pseudorandom**
  - Being pseudorandom means that the hash value returned by a hash function changes unpredictably when the input data are changed.
  - Even if the input data were changed only a little bit, the resulting hash value will differ unpredictably.
  - It should not be possible to predict the hash value based on the input data

Requirements of Cryptographic Hash Function

- **One-Way Function**
  - it is impossible to recover the original input data based on the hash value.

Cryptographic Hash Function Criteria

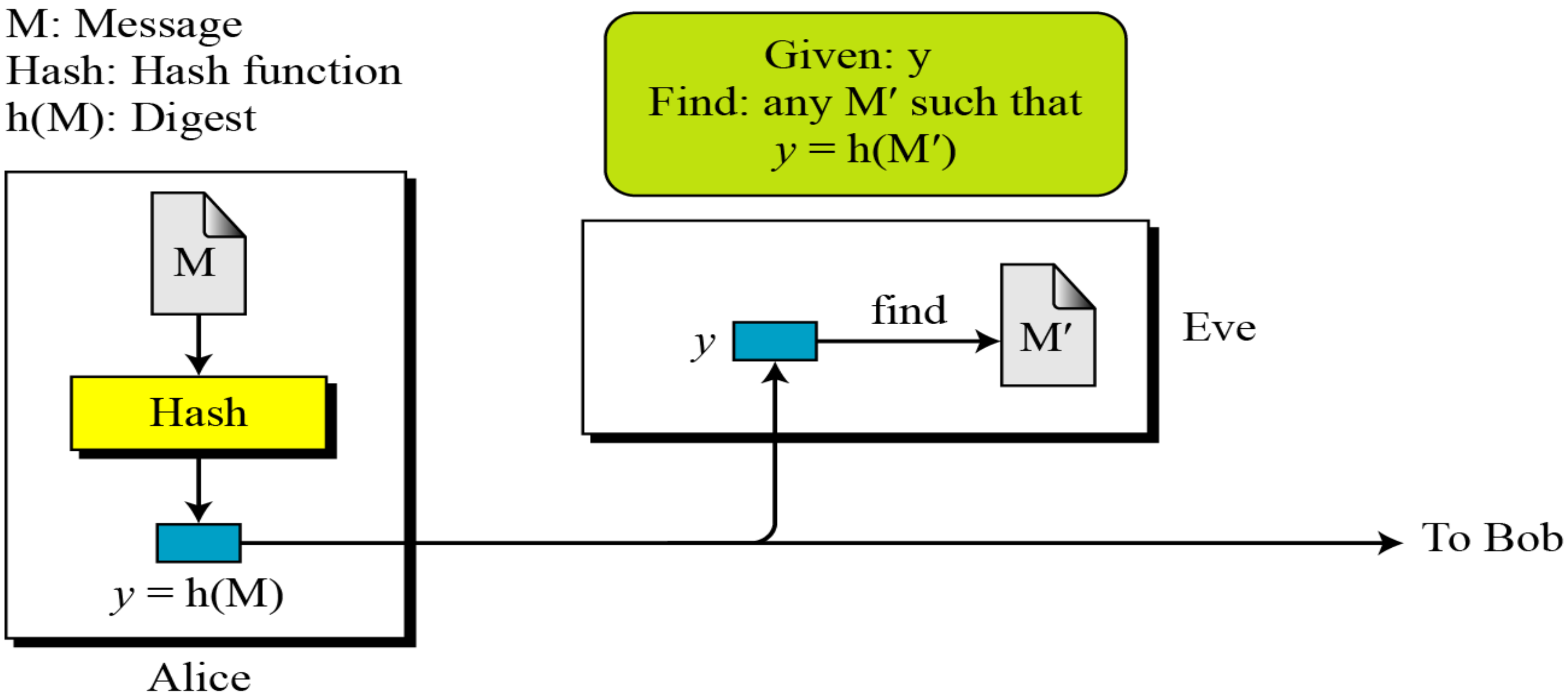
- A cryptographic hash function must satisfy three criteria: preimage resistance, second preimage resistance, and collision resistance.



Preimage Resistance

Preimage Attack

Given: y = h(M) **Find: M′ such that y = h(M′)**



Example

1. Can we use a conventional lossless compression method such as *StuffIt* as a cryptographic hash function?

- Solution
- We cannot. A lossless compression method creates a compressed message that is reversible.

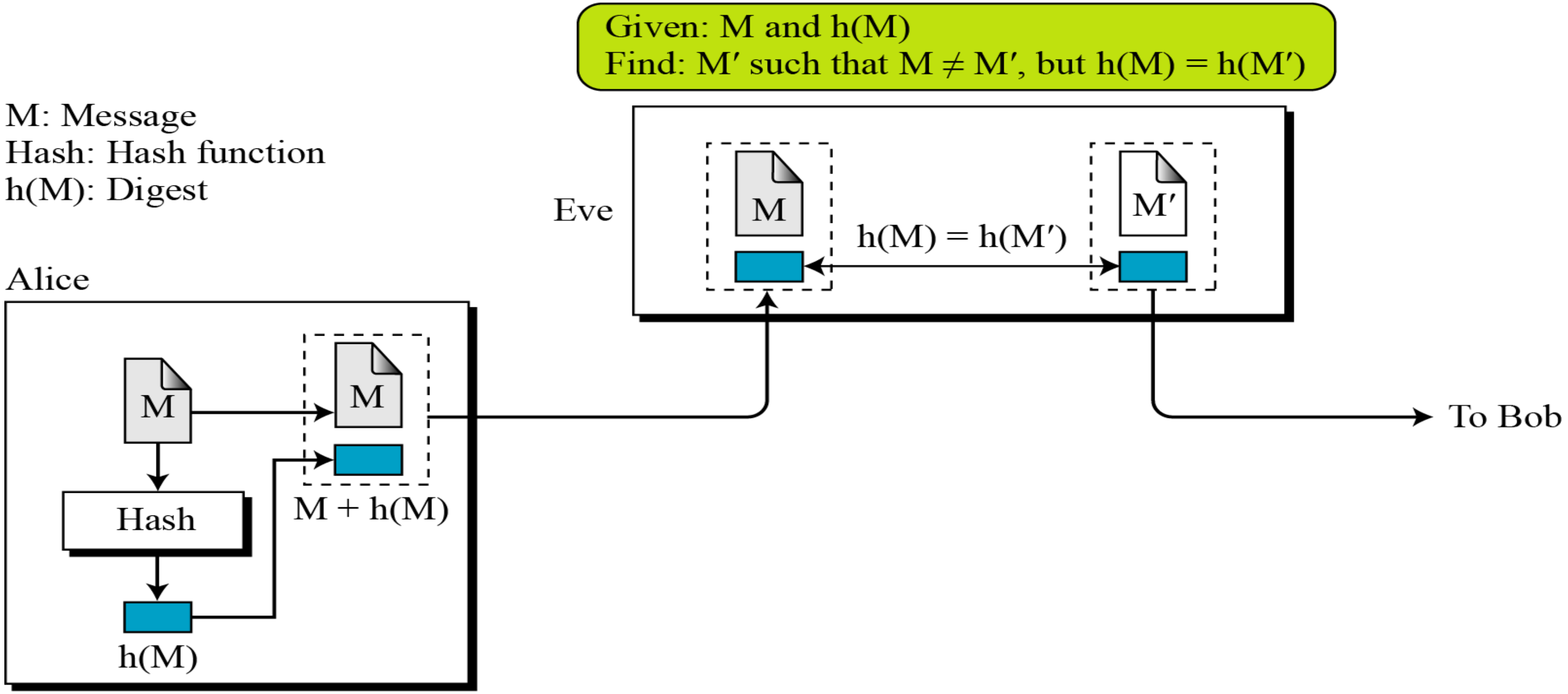
2. Can we use a checksum function as a cryptographic hash function?

- Solution
- We cannot. A checksum function is not preimage resistant, Eve may find several messages whose checksum matches the given one.

Second Preimage Resistance

Second Preimage Attack

Given: M and h(M) **Find: M′ ≠ M such that h(M) = h(M′)**



Collision Resistance

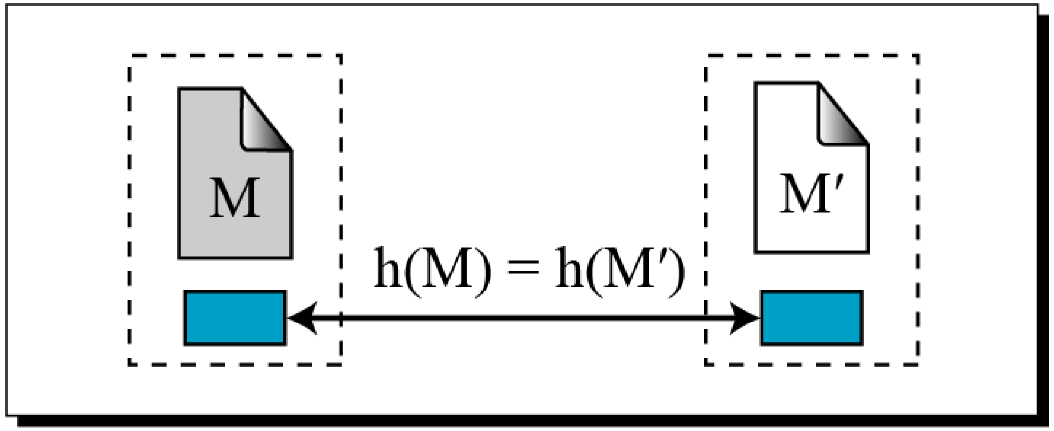
Collision Attack

Given: none **Find: $M' \neq M$ such that $h(M) = h(M')$**

M: Message
Hash: Hash function
h(M): Digest

Find: M and M' such that $M \neq M'$, but $h(M) = h(M')$

Eve



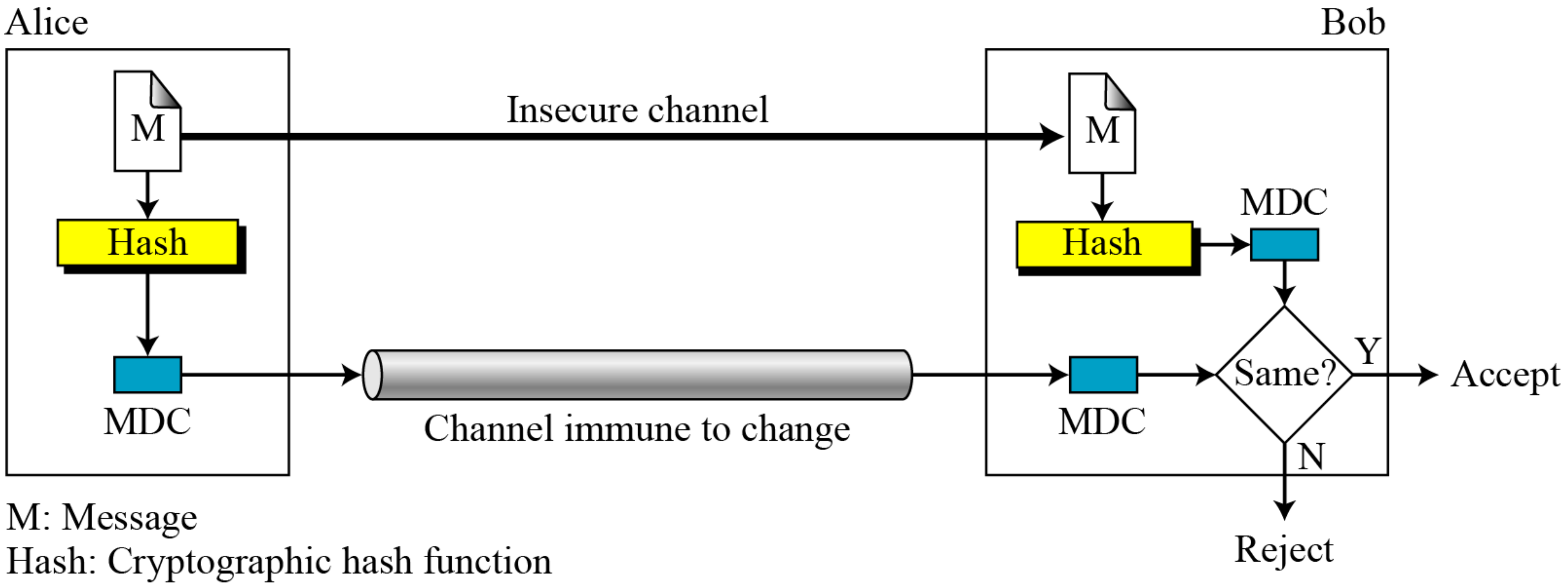
Modification Detection Code (MDC)

- A message digest does not authenticate the sender of the message.
- To provide message authentication, Alice needs to provide proof that it is Alice sending the message and not an impostor.
- The digest created by a cryptographic hash function is normally called a modification detection code (MDC).
- What we need for message authentication is a message authentication code (MAC).

MDC

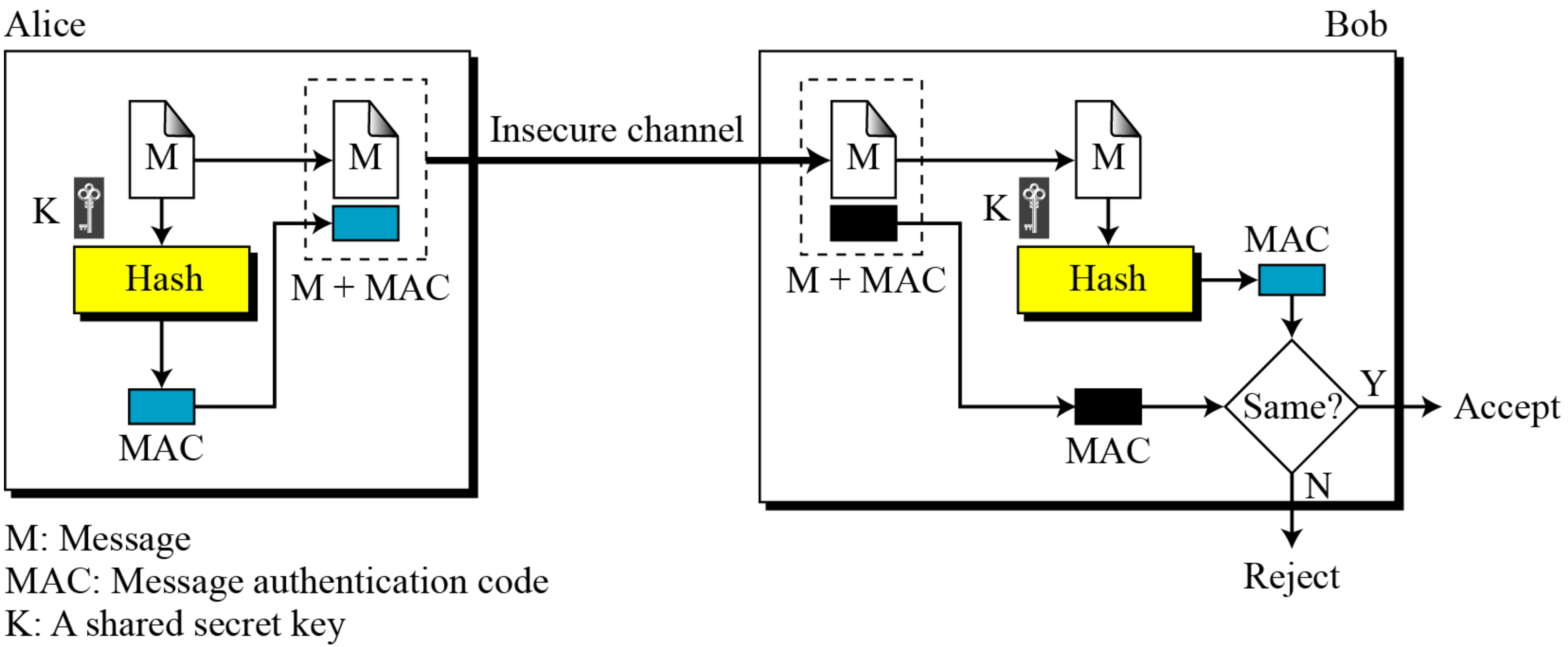
- A modification detection code (MDC) is a message digest that can prove the integrity of the message: that message has not been changed.
- If Alice needs to send a message to Bob and be sure that the message will not change during transmission
- Alice can create a message digest, MDC, and send both the message and the MDC to Bob.
- Bob can create a new MDC from the message and compare the received MDC and the new MDC.
- If they are the same, the message has not been changed.

MDC



M: Message
Hash: Cryptographic hash function
MDC: Modification detection code

Message Authentication Code



The security of a MAC depends on the security of the underlying hash algorithm.

Hash Function – Classification and Framework

- Definition:

  A hash function is a function h which has, as a minimum, the following two properties:

1. Compression- h maps an input x of arbitrary finite bit length, to an output h(x) of fixed bitlength n.

2. ease of computation- given h and an input x, h(x) is easy to compute.

Hash Function – Classification and Framework

- May be split into two classes:
  - Unkeyed hash functions
    - a single input parameter(a message)
  - keyed hash functions
    - two distinct input parameters, a message and a secret key

Hash Function – Classification and Framework

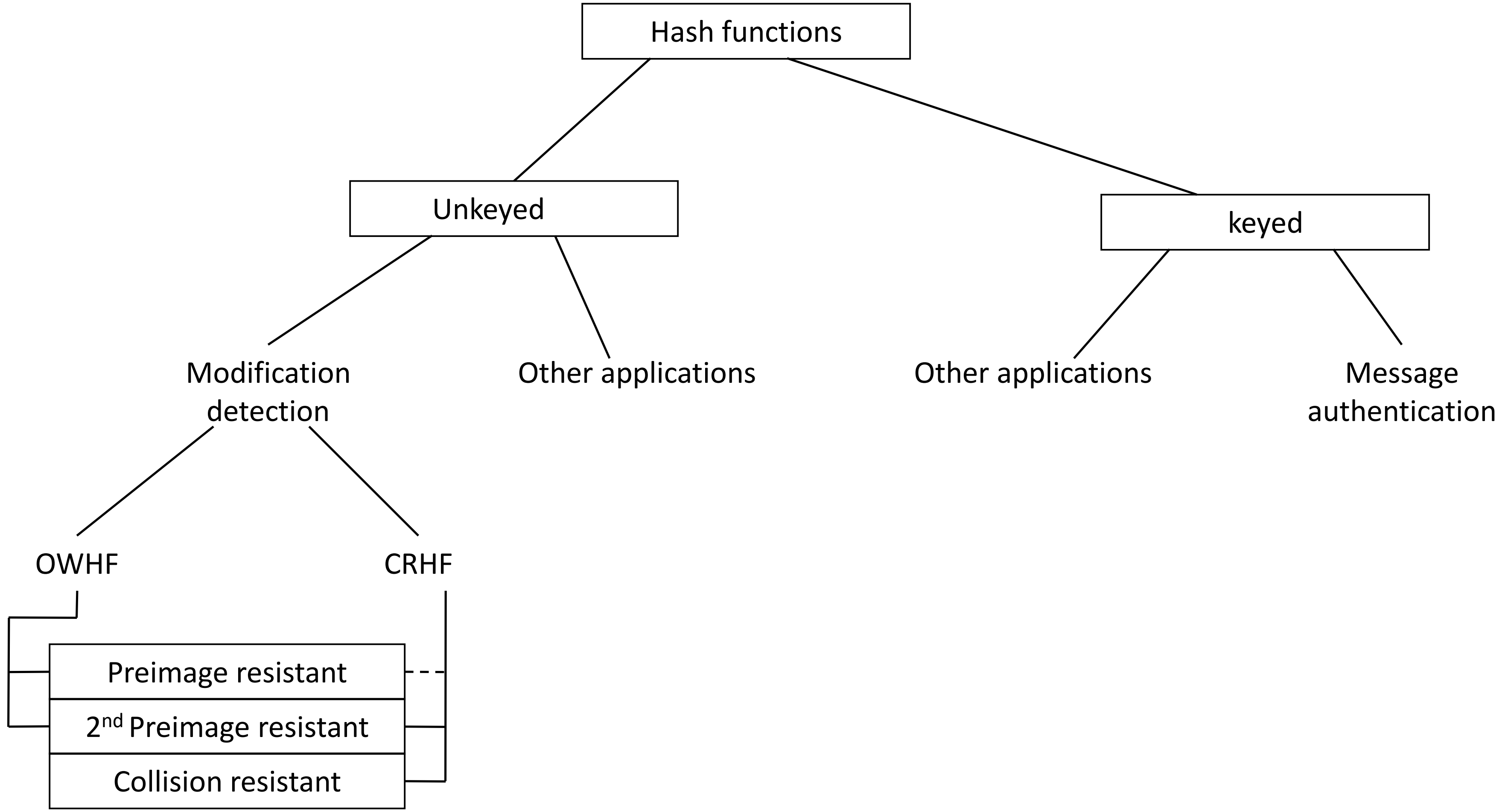
1. Modification detection codes (MDCs)
   - to facilitate data integrity assurance
   - Subclasses of unkeyed hash functions
   - Further classified into
     1. One-way hash functions(OWHFs):
        - Finding an input which hashes to a pre-specified hash value is difficult
     2. Collision resistant hash functions(CRHFs):
        - Finding any two inputs having the same hash-value is difficult

Hash Function – Classification and Framework

Message authentication codes(MACs)

- To facilitate assurance regarding both the source of a message and its integrity
- Subclass of keyed hash functions

Hash Function – Classification and Framework



Security objective and basic attack

Hash type	Design goal	Ideal strength	Adversary's goal
OWHF	preimage resistance; 2nd-preimage resistance	2^n 2^n	produce preimage; find 2nd input, same image
CRHF	collision resistance	$2^{n/2}$	produce any collision
MAC	key non-recovery; computation resistance	2^t $P_f = \max(2^{-t}, 2^{-n})$	deduce MAC key; produce new (msg, MAC)

Table 9.2: *Design objectives for n-bit hash functions (t-bit MAC key). P_f denotes the probability of forgery by correctly guessing a MAC.*

General model for iterated hash functions

- hashing by processing successive fixed size blocks of the input
- the hash function with input x = x1x2…xt can be modeled as,

H_0 = IV

$H_i = f(H_i\text{-}1, x_i)$, 1<=i<=t;

$H(x) = g(H_t)$

- H_{i-1} → n-bit chaining variable between stage i-1 and stage i

General model for iterated hash functions

- output function to map n-bit chaining variable to m-bit result
- Often the identity mapping
  - $g(H_t) = H_t$

General model fo iterated hash functions

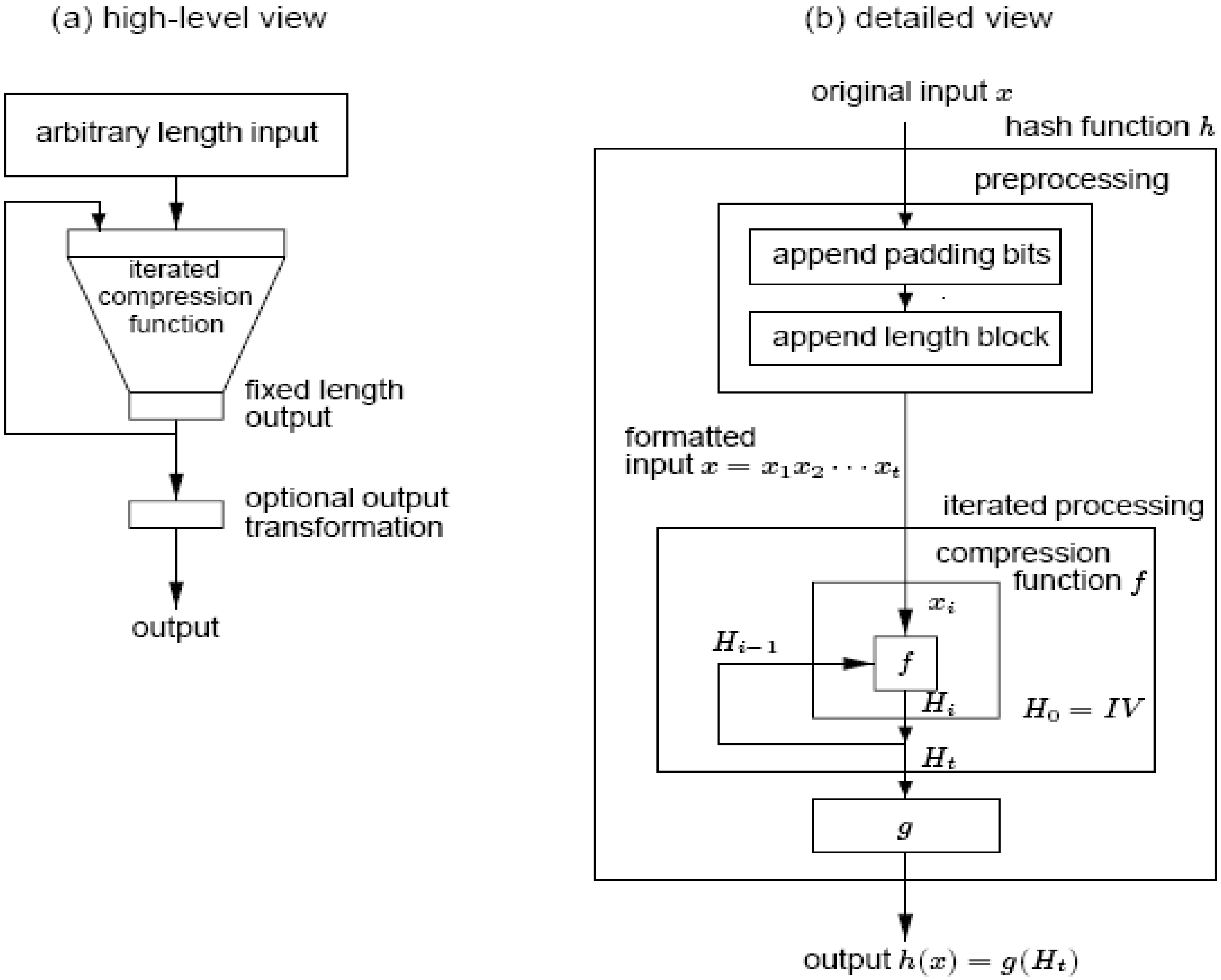


Figure 9.2: *General model for an iterated hash function.*

MD4 - Message Digest

- One-Way hash function designed by Ron Rivest
- MD stands for Message Digest,
- Produces 128-bit hash
- Specified as Internet standard RFC1320
- Design goals:
  - Security :
    - Its is computationally infeasible to find two messages that hash to same value
    - No attack is more efficient than brute force
  - Direct Security:
    - Not based on any assumption

MD4

- Speed :
  - Suitable for high-speed software implementations
  - Based on simple set of bit manipulations on 32-bit operands

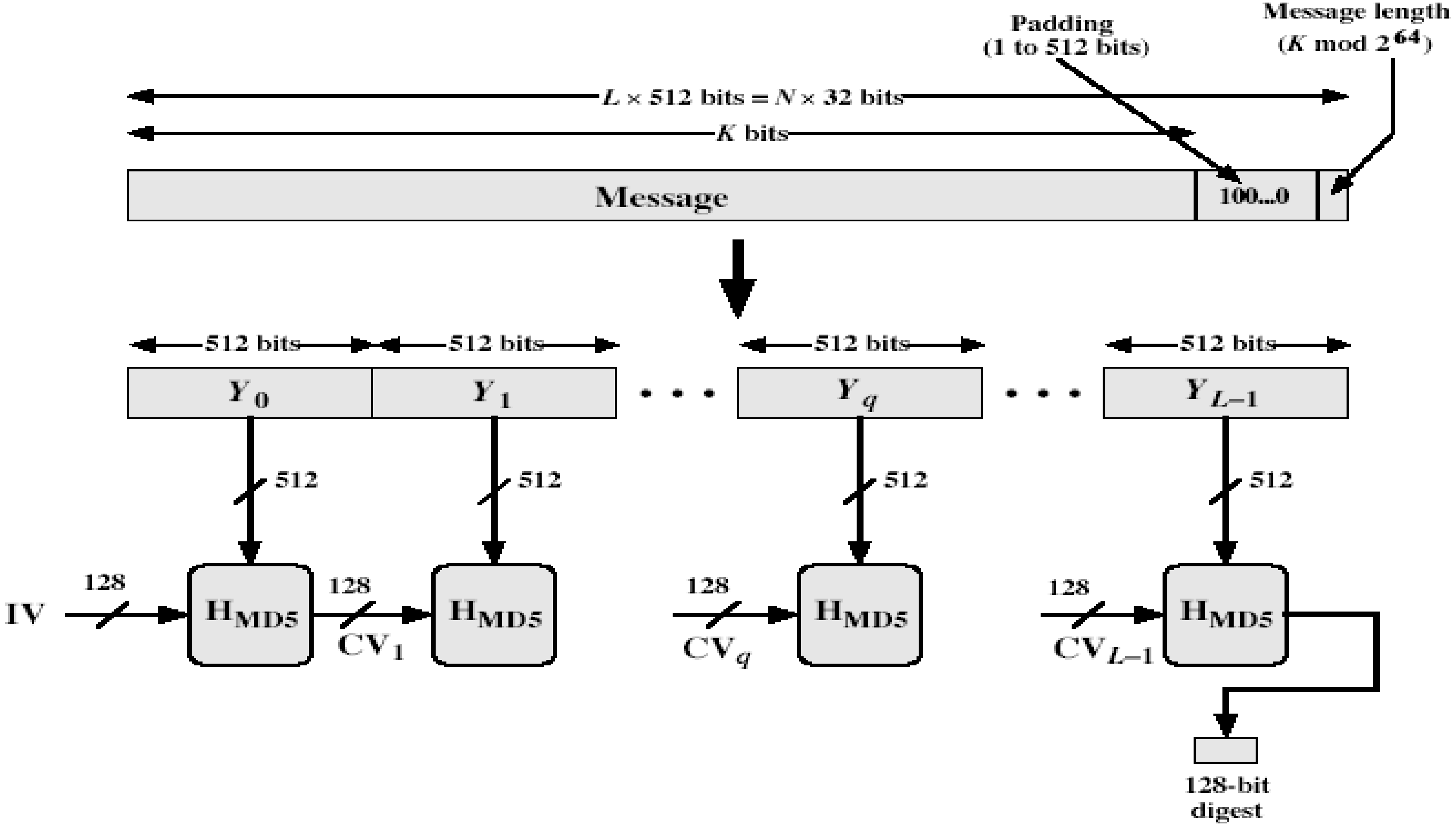
- Simplicity and compactness:
  - MD4 is as simple as possible, without large data structures or a complicated program.

- Favor Little-Endian Architectures
  - MD4 is optimized for microprocessor architectures (specifically Intel microprocessors); larger and faster computers make any necessary translations

MD5

- Improved version of MD4
- MD5 is designed by well-known cryptographer Ronald Rivest in 1991
- The MD5 function is a cryptographic algorithm that takes an input of arbitrary length and produces a *message digest* that is **128 bits** long.
- Specified as Internet standard RFC1321

MD5



Padding
(1 to 512 bits)
Message length
(K mod 2 64)
L × 512 bits = N × 32 bits
K bits
Message
100...0
512 bits
512 bits
512 bits
512 bits
Y 0
Y 1
Y q
Y L–1
512
512
512
512
IV
128
128
128
128
HMD5
HMD5
HMD5
HMD5
CV1
CVq
CVL–1
128-bit
digest

MD5

- **Preparing the input**
  - The MD5 algorithm first divides the input in **blocks** of 512 bits each.
  - 64 Bits are appended at the end of the last block. These 64 bits represent the length of the original input.
  - If the last block is less than 512 bits, the message is padded (1 followed by 0s) such that its length ≡ 448 mod 512
  - Next, each **block** is divided into 16 **words** of 32 bits each. These are denoted as M_0 ... M_{15}.

MD5

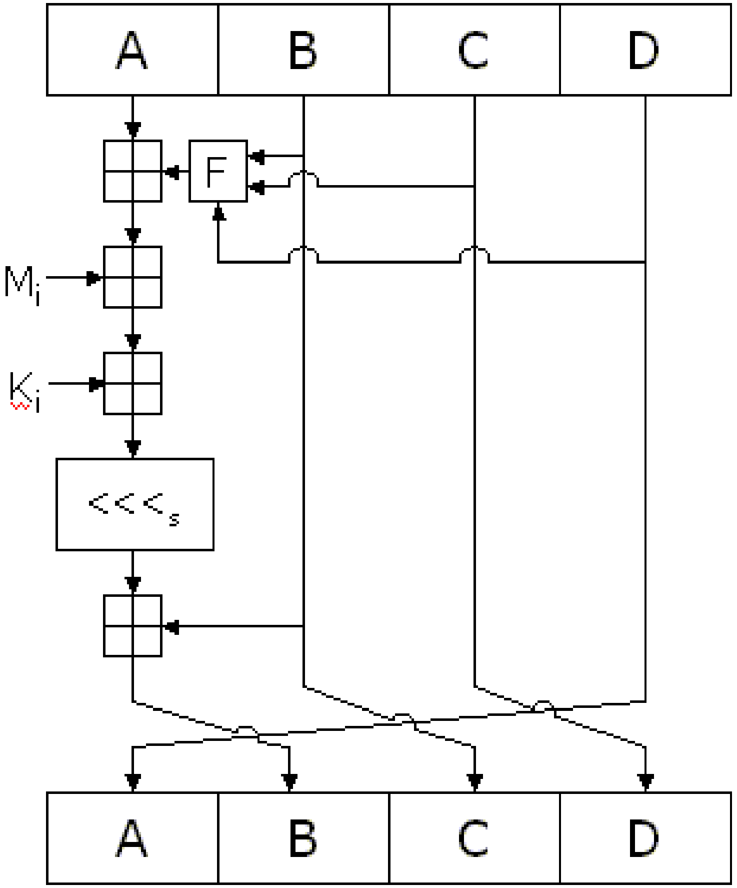
- **MD5 helper functions**
  - MD5 uses a buffer that is made up of four **words** that are each 32 bits long. These words are called A, B, C and D. They are initialized as
    - A = 01 23 45 67
    - B = 89 AB CD EF
    - C = FE DC BA 98
    - D = 76 54 32 10
  - These words are called chaining variables
- Ki = Constant Value derived from sin function
- Int(abs(sin(i)) * 2^{32}) $0<i<65$

MD5

- **Four auxiliary functions**
  - In addition MD5 uses four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.
  - They apply the logical operators and, or, not and xor to the input bits.
  - The message is processed in 16-word (512-bit) chunks, using 4 rounds of 16 steps each

$$F(x,y,z) = (x \wedge y) \vee (\sim x \wedge z)$$
$$G(x,y,z) = (x \wedge z) \vee (y \wedge \sim z)$$
$$H(x,y,z) = x \oplus y \oplus z$$
$$I(x,y,z) = y \oplus (x \wedge \sim z)$$

MD5



MD5

- Round 1
  - FF(a,b,c,d,M_j,s,t_i) denotes $a = b + ((a + \mathbf{F(b,c,d\,)} + M_j + t_i) <<< s)$
    - 16steps:
      - FF (a, b, c, d, M_0, 7, 0xd76aa478)
      - FF (d, a, b, c, M_1, 12, 0xe8c7b756)

        ....
      - FF (b, c, d, a, M_{15}, 22, 0x49b40821)

MD5

- Round2
  - GG(a,b,c,d,M_j,s,t_i) denotes $a = b + ((a + \mathbf{G(b,c,d)} + M_j + t_i) <<< s)$
    - GG (a, b, c, d, M_1, 5, 0xf61e2562) ..... Upto 16steps
- Round3
  - HH(a,b,c,d,M_j,s,t_i) denotes $a = b + ((a + \mathbf{H(b,c,d)} + M_j + t_i) <<< s)$
    - HH (a, b, c, d, M_5, 4, 0xfffa3942) 16steps
- Round4
  - II(a,b,c,d,M_j,s,t_i) denotes $a = b + ((a + \mathbf{I(b,c,d)} + M_j + t_i) <<< s)$
    - II (a, b, c, d, M_0, 6, 0xf4292244) .....16 steps

MD5

- After all of this, *a, b, c,* and *d* are added to *A, B, C, D,* respectively, and the algorithm continues with the next block of data.
- The final output is the concatenation of *A, B, C,* and *D*.

Security of MD5

- Size of message digest 128bit is too small.

Cryptanalysis of MD5

- Collisions for the compression function of MD5 have been demonstrated, though collisions for the full MD5 have not yet been achieved

- Existing signatures formed using MD5 are not at risk and while MD5 is still suitable for a variety of applications (namely those which rely on the one-way property of MD5 and on the random appearance of the output) as a precaution it should not be used for future applications that require the hash function to be collision-resistant.

Secure Hash Algorithm (SHA)

- SHA originally designed by NIST (National Institute of standards and technology) and published as a Federal Information Processing Standard (FIPS 180) in 1993.
- Was revised in 1995 as FIPS 180-1 and referred to as SHA-1, also Internet RFC3174
- Three generations of Secure Hash Algorithm.

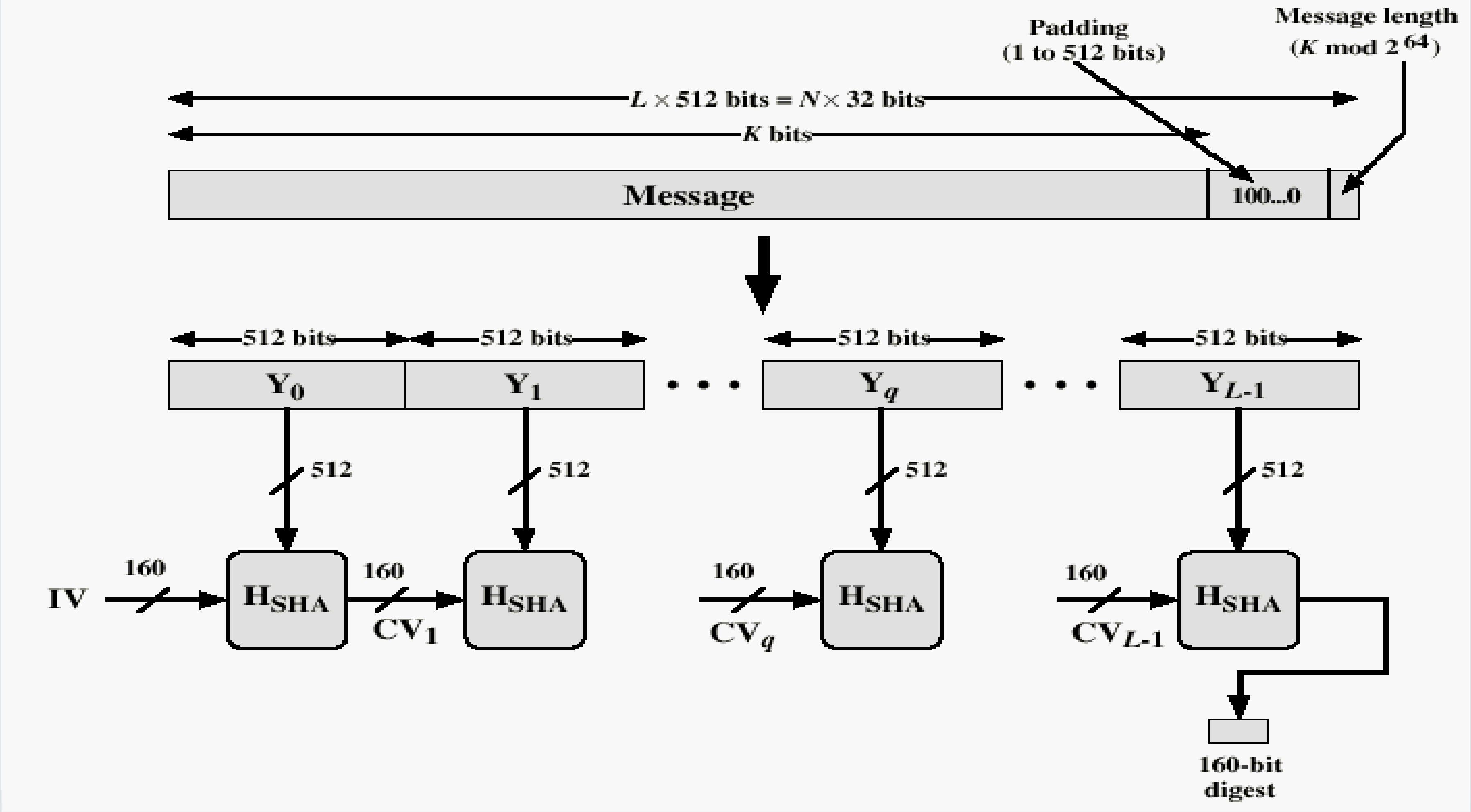
Secure Hash Algorithm (SHA)

SHA-Generation	SHA-1	SHA-2			SHA-3
	SHA-1	SHA-256	SHA-384	SHA-512	future hash function standard still in development
Message digest size	160	256	384	512	
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$	
Block size	512	512	1024	1024	
Word Size	32	32	64	64	
Number of Steps	80	64	80	80	

Secure Hash Algorithm (SHA) - 1

- SHA-1 produces a 160-bit digest from a message with a maximum length of $(2^{64} - 1)$ bits.
- SHA-1 is based on principles similar to those used by Ronald L. Rivest of MIT in the design of the MD4 and MD5 message digest algorithms.
- Preprocessing : exactly same as MD5

Secure Hash Algorithm (SHA)



Secure Hash Algorithm (SHA)

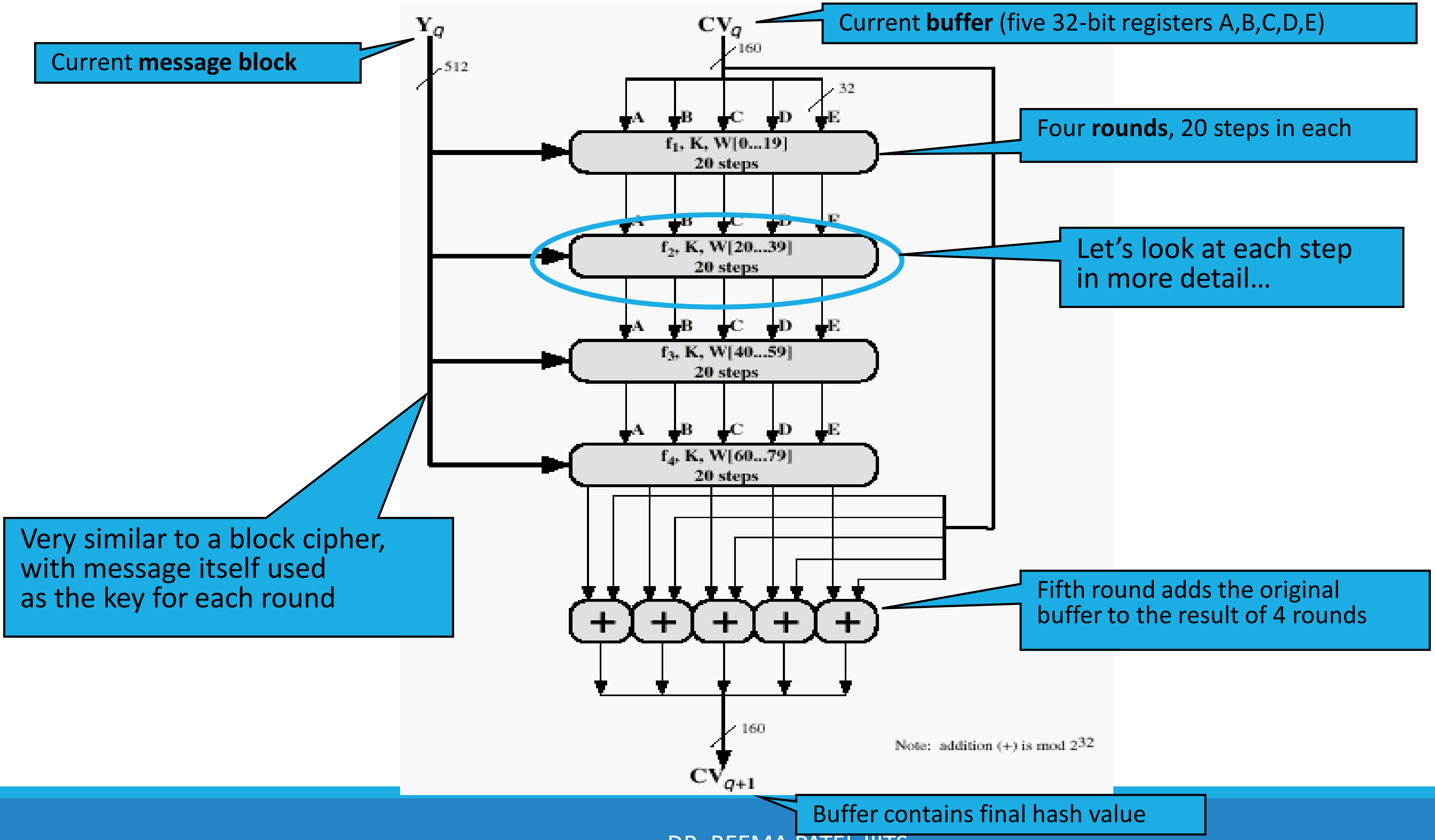
- Five chaining variables
  - A = 67452301
  - B = efcdab89
  - C = 98badcfe
  - D = 10325476
  - E = c3d2e1f0
- Main loop : Four rounds of 20 operations each
  - $ft(X,Y,Z) = (X \land Y) \lor ((\neg X) \land Z)$, for t=0 to 19
  - $ft(X,Y,Z) = X + Y + Z$, for t = 20 to 39
  - $ft(X,Y,Z) = (X \land Y) \lor (X \land Z) \lor (Y \land Z)$, for t=40 to 59
  - $ft(X,Y,Z) = X + Y + Z$, for t = 60 to 79
- + = XOR

Secure Hash Algorithm (SHA)

- Four constants are used :
  - K_t = 0x5a827999, for t = 0 to 19
  - K_t = 0x6ed9eba1, for t = 20 to 39
  - K_t = 0x8f1bbcdc, for t = 40 to 59
  - K_t = 0xca62c1d6, for t = 60 to 79

- Message block is transferred from 16 blocks to 80 blocks:
  - $W_t = M_t$, for t=0 to 15
  - $W_t = (W_{t-3} + W_{t-8} + W_{t-14} + W_{t-16}) <<< 1$, for t=16 to 79

Secure Hash Algorithm (SHA)



Secure Hash Algorithm (SHA)

- If t is the operation number (from 0 to 79), W_t represents the t th sub-block of the expanded message, and <<< s represents a left circular shift of s bits, then the main loop looks like:

For t=0 to 79

TEMP = (a <<< 5) + f_t(b,c,d) + e + W_t + K_t

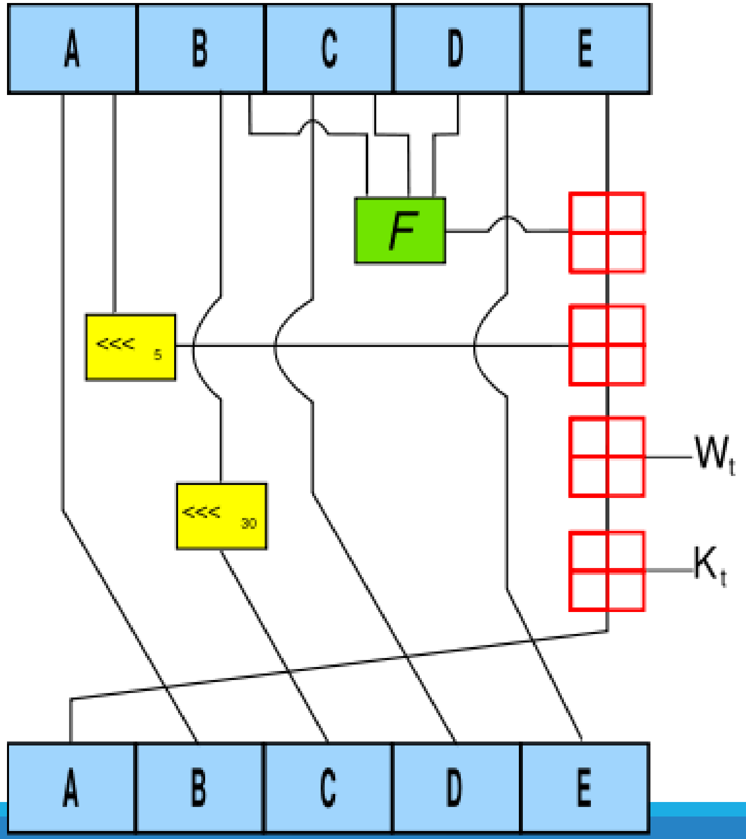
e = d

d=c

c=b <<< 30

b = a

a = TEMP



Security of SHA-1

- “a fourth round is added “ SHA does this too. However in SHA, the fourth round uses the same f function as the second round.

- “Each step now has unique additive constant” True for SHA where it reuses the constants for each group of 20 rounds

- “Faster avalanche effect” True for SHA. Addition of fifth variable to make Boer-Bosselaers attack against MD5 impossible against SHA.

Security of SHA-1

- “The order in which message sub-blocks are accessed in rounds 2 and 3 is changed “. SHA is completely different.

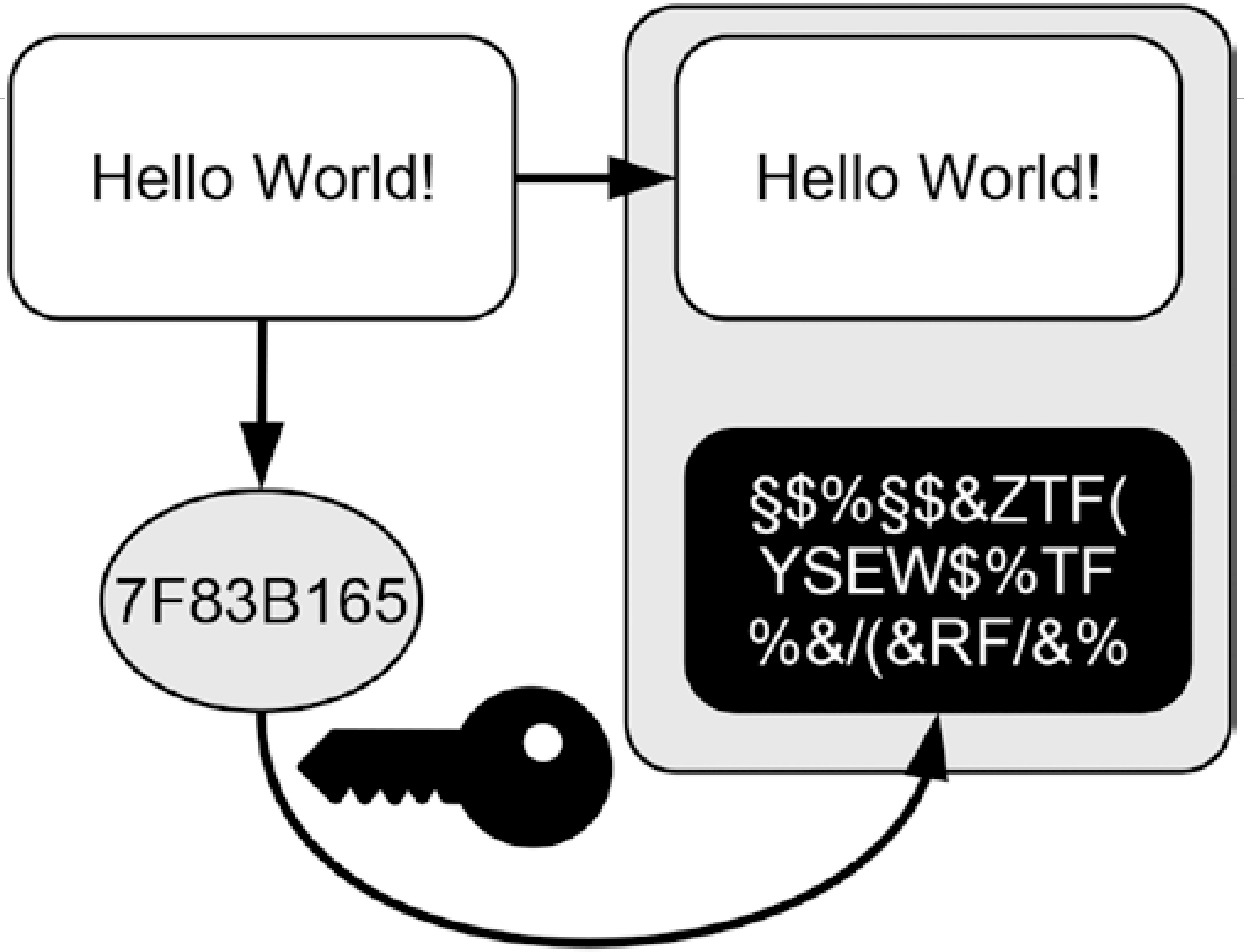
- “The left circular shift… to yield faster avalanche effect.” SHA uses a constant shift amount in each round. This amount is relatively prime to the word size, as in MD4

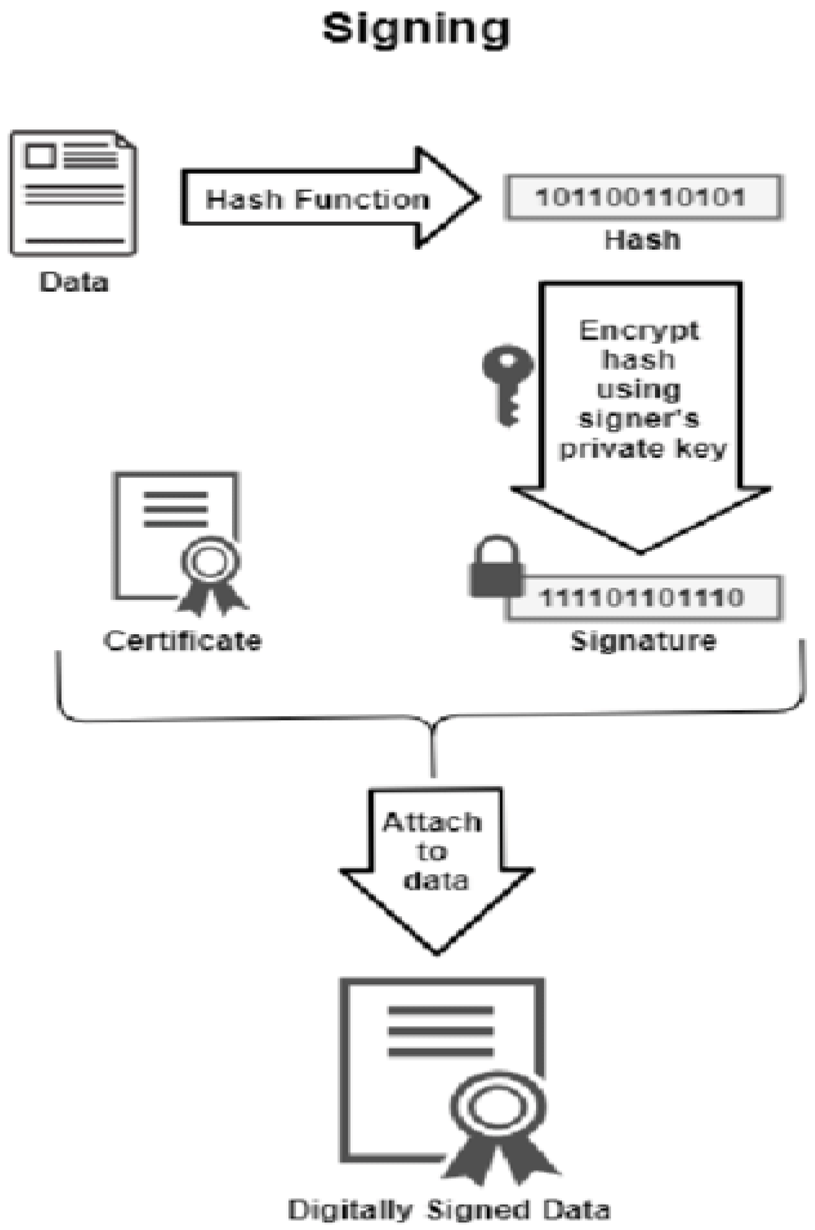
- *SHA is MD4 with the addition of an expand transformation, an extra round, and better avalanche effect; MD5 is MD4 with improved bit hashing, an extra round, and better avalanche effect*

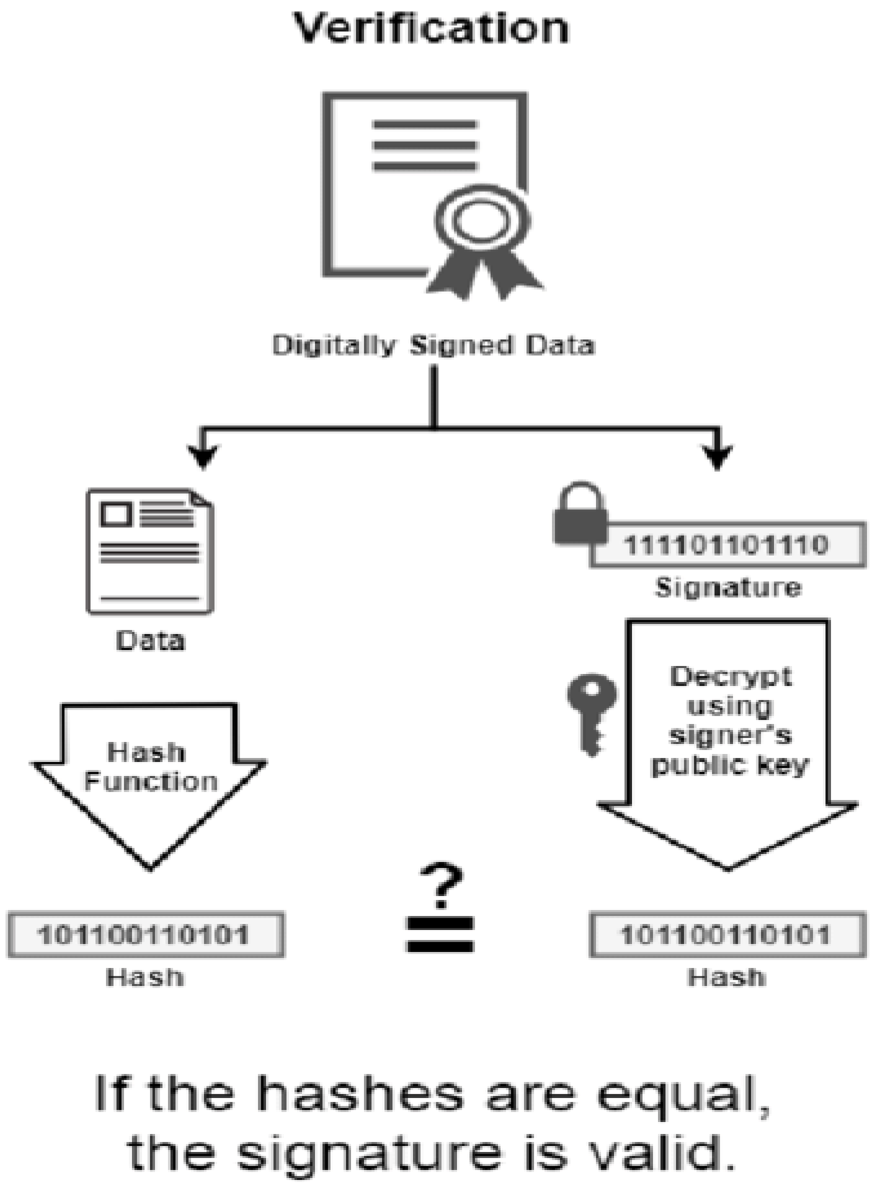
Digital Signature

- Digital signatures are the equivalent of handwritten signatures.
- They utilize cryptographic hashing and the private-to-public information flow of asymmetric cryptography.
- Three major elements of digital signatures:
  - Creating a signature
  - Verifying data by using the signature
  - Identifying fraud by using the signature

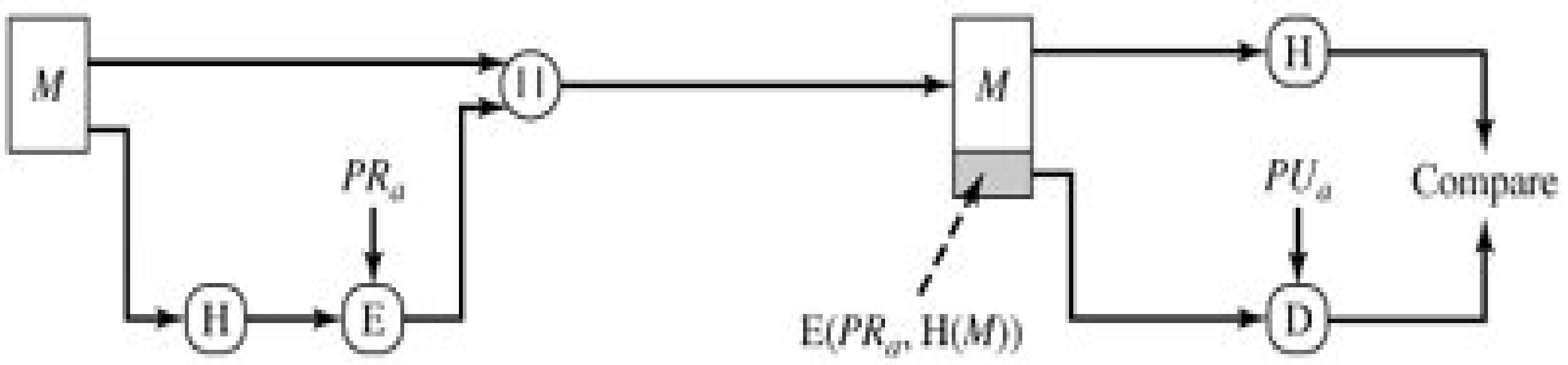
Digital Signature




Signing
Data
Hash Function
101100110101
Hash
Encrypt hash using signer's private key
Certificate
111101101110
Signature
Attach to data
Digitally Signed Data

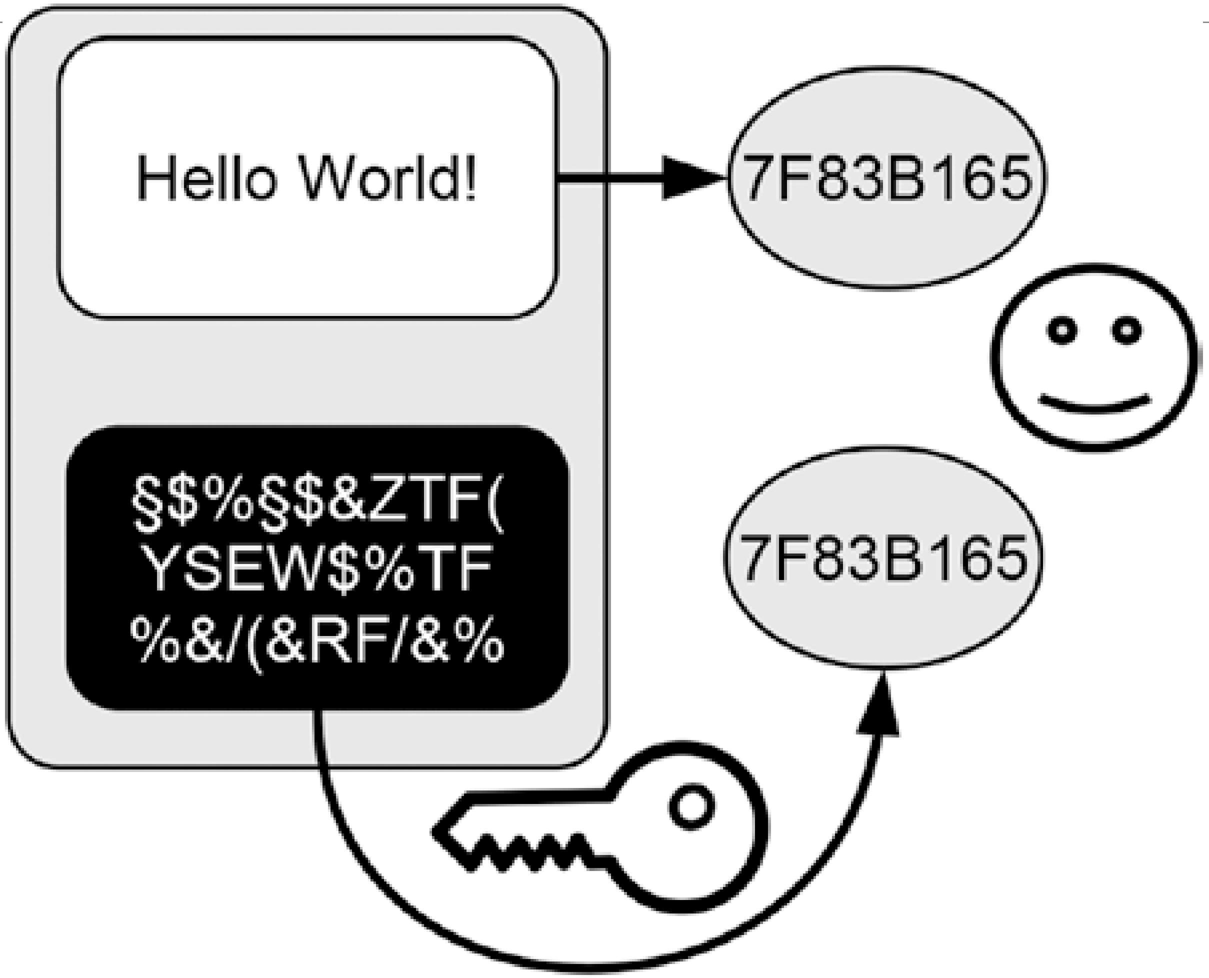

Verification
Digitally Signed Data
Data
111101101110
Signature
Hash Function
Decrypt using signer's public key
101100110101
Hash
?
=
101100110101
Hash
If the hashes are equal,
the signature is valid.

Digital Signature



Digital Signature

- Verifying Data by Using the Signature



Identifying Fraud by Using the Digital Signature