BK
TP.HCM

Cryptography and Network Security

Chapter 4 – Part A

Cryptographic Hash Functions

Lectured by
Nguyễn Đức Thái

Outline

- Cryptographic Hash Functions
- Message Authentication
- Attacks on Hash Functions
  - Brute-Force Attacks
  - Cryptanalysis Attacks
- Secure Hash Algorithm (SHA)

Hash functions

- A hash function maps a ***variable-length message*** into a ***fixed-length hash value***, or message digest

- A ***hash function H*** accepts a variable-length block of data as input and produces a fixed-size hash value

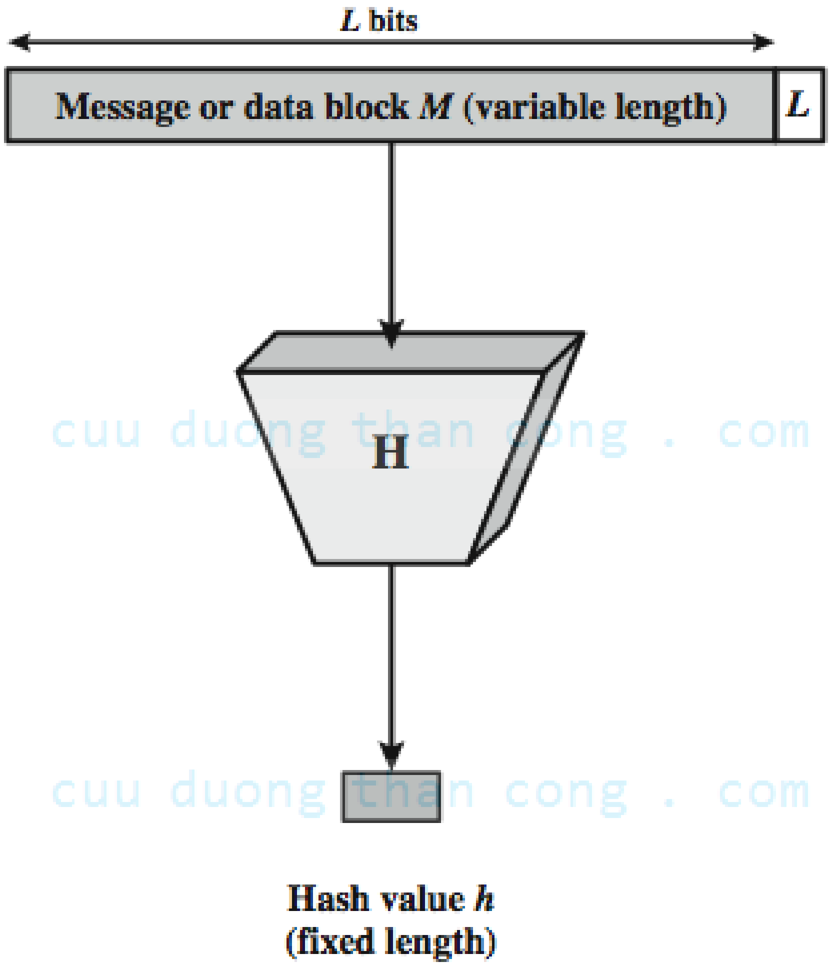
$$h = \mathrm{H}(M)$$

- The ***principal object*** of a hash function is ***data integrity***

Cryptographic Hash functions

- The kind of hash function needed for ***security applications*** is referred to as a ***cryptographic hash function***.
- A ***cryptographic hash function*** is an algorithm for which it is *computationally* ***infeasible***
- Because of these characteristics, hash functions are often used to determine ***whether or not data has changed***

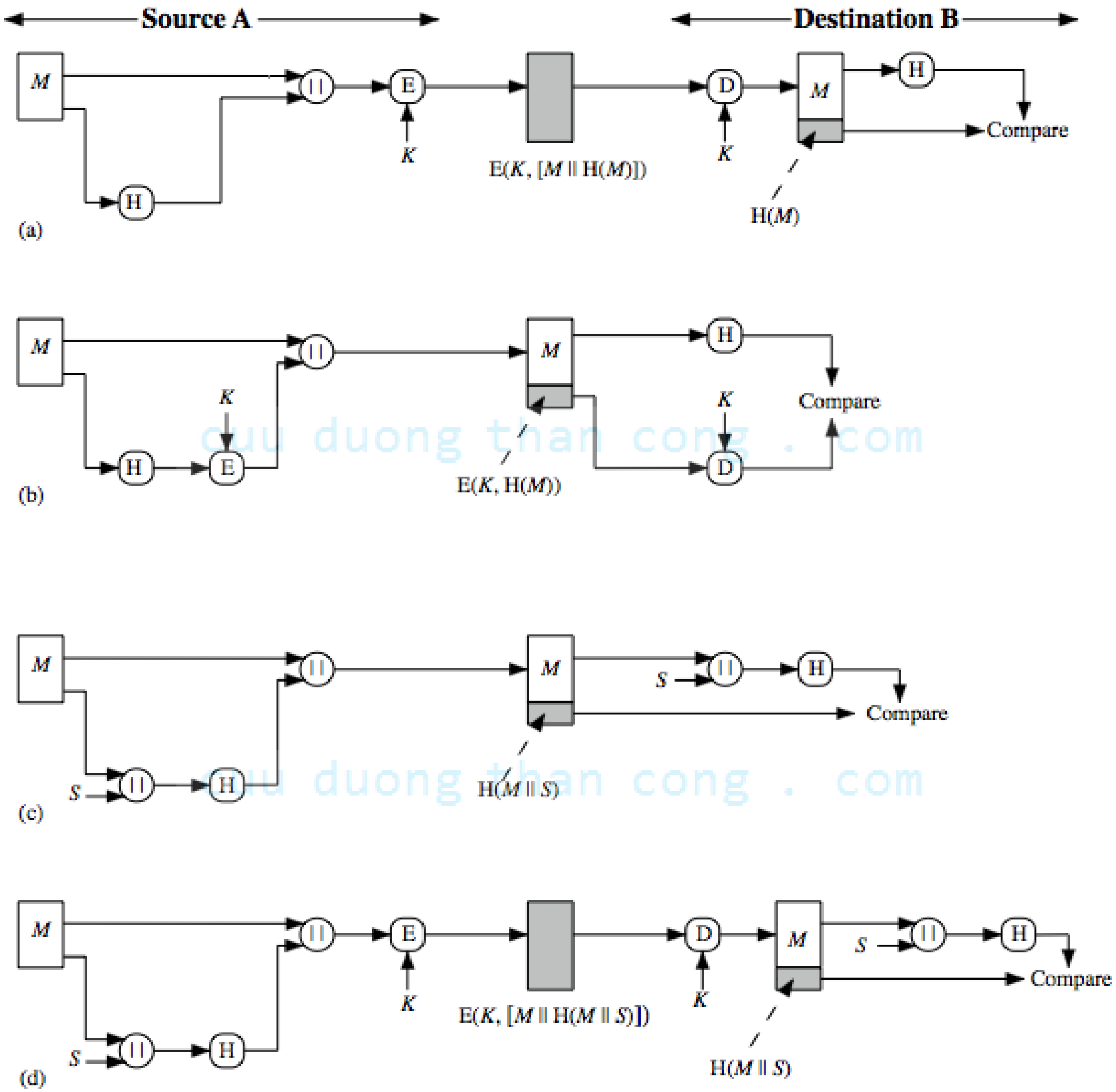
Cryptographic Hash functions



Message Authentication

- Message authentication is a ***mechanism*** or ***service*** used to verify the ***integrity of a message***.
- Message authentication assures that data received are ***exactly*** as sent (i.e., contain no modification, insertion, deletion, or replay).
- When a hash function is used to provide message authentication, the hash function value is often referred to as a ***message digest***.

Hash Functions & Msg Authentication



Message Authentication – Picture a)

- The message plus concatenated hash code is encrypted using ***symmetric encryption***.
- Because only A and *B* ***share the secret key***, the message must have come from A and has not been altered.
- The hash code provides the structure or redundancy required to achieve authentication.
- Because encryption is applied to the entire message plus hash code, ***confidentiality is also provided***

Message Authentication – Picture b)

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

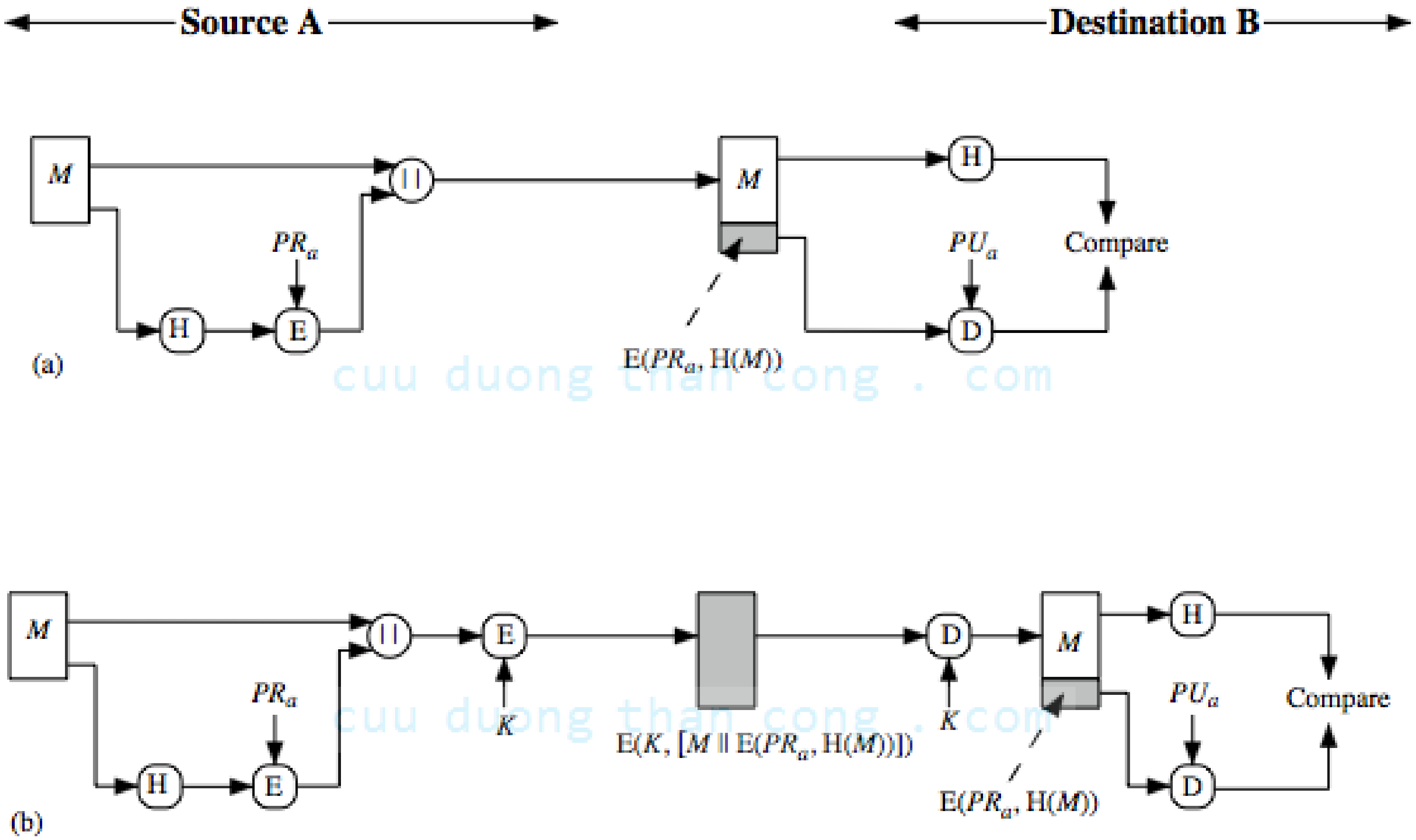
Message Authentication – Picture c)

- It is possible to use a hash function but ***no encryption*** for message authentication.
- The technique assumes that 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.
- Because B possesses, it can recompute the hash value to verify.
- Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.

Message Authentication – Picture d)

- ***Confidentiality can be added*** to the approach of method (c) by ***encrypting*** the entire message plus the hash code

Hash Functions & Digital Signatures



Hash Functions & Dig. Signatures – a)

- The hash code is ***encrypted***, using public-key encryption with the sender's private key.
- It also provides a ***digital signature***, because only the sender could have produced the encrypted hash code.
- In fact, this is the essence of the digital signature technique.

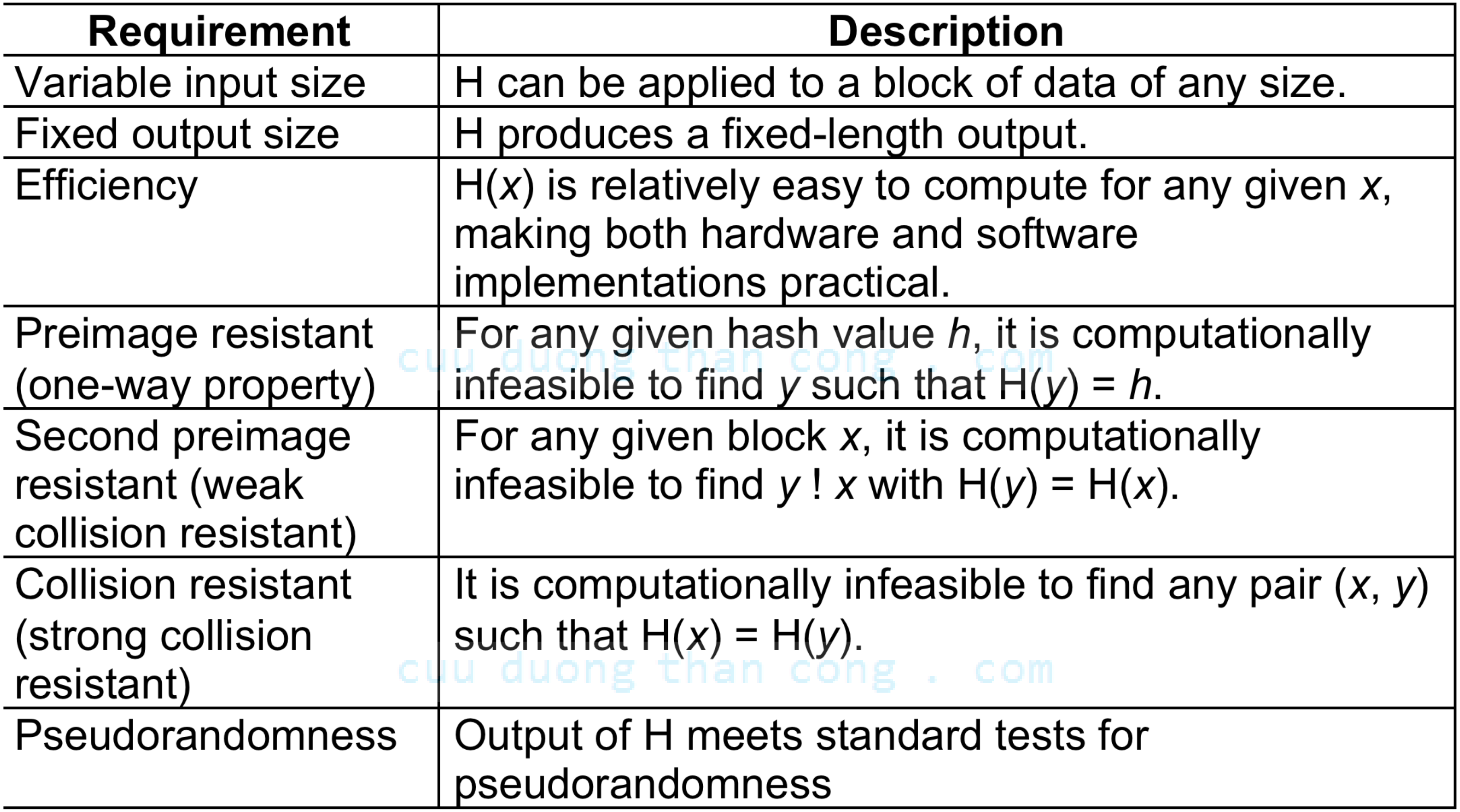
Hash Functions & Dig. Signatures – b)

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

Other Hash Functions Uses

- **Hash functions are commonly used to *create a one-way password file*.**
  - Thus, the actual password is not retrievable by a hacker who gains access to the password file.
  - This approach to password protection is used by most operating systems.
- **Hash functions can be used for *intrusion detection* and *virus detection*.**
  - Store H(F) for each file on a system and secure the hash values (e.g., on a CD-R that is kept secure).
  - One can later determine if a file has been modified by recomputing H(F).
  - An intruder would need to change F without changing H(F).
- **Can be used to construct a pseudorandom function (PRF) or a pseudorandom number generator (PRNG).**

Hash Functions Requirements



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* ! *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

Attacks on Hash Functions

- Brute-Force attacks
  - Preimage and second preimage attacks
  - Collision resistant attacks

- Cryptanalysis attacks

Brute-Force Attacks

- A brute-force attack ***does not depend on the specific algorithm*** but ***depends only on bit length***.
- In the case of a hash function, ***a brute-force attack depends only on the bit length of the hash value***.
- A cryptanalysis, in contrast, is an attack based on ***weaknesses in a particular cryptographic algorithm***.

Preimage & Second Preimage Attacks

- For a preimage or second preimage attack, an adversary wishes to find a value such that H(*y*) is equal to a given hash value.
- The brute-force method is to pick values of *y* at random and try each value until a collision occurs.
- For an *m*-bit hash value, the level of effort is proportional to 2^m
- Specifically, the adversary would have to try, on average, **2^{m-1} values** of *y* to find one that generates a given hash value h.

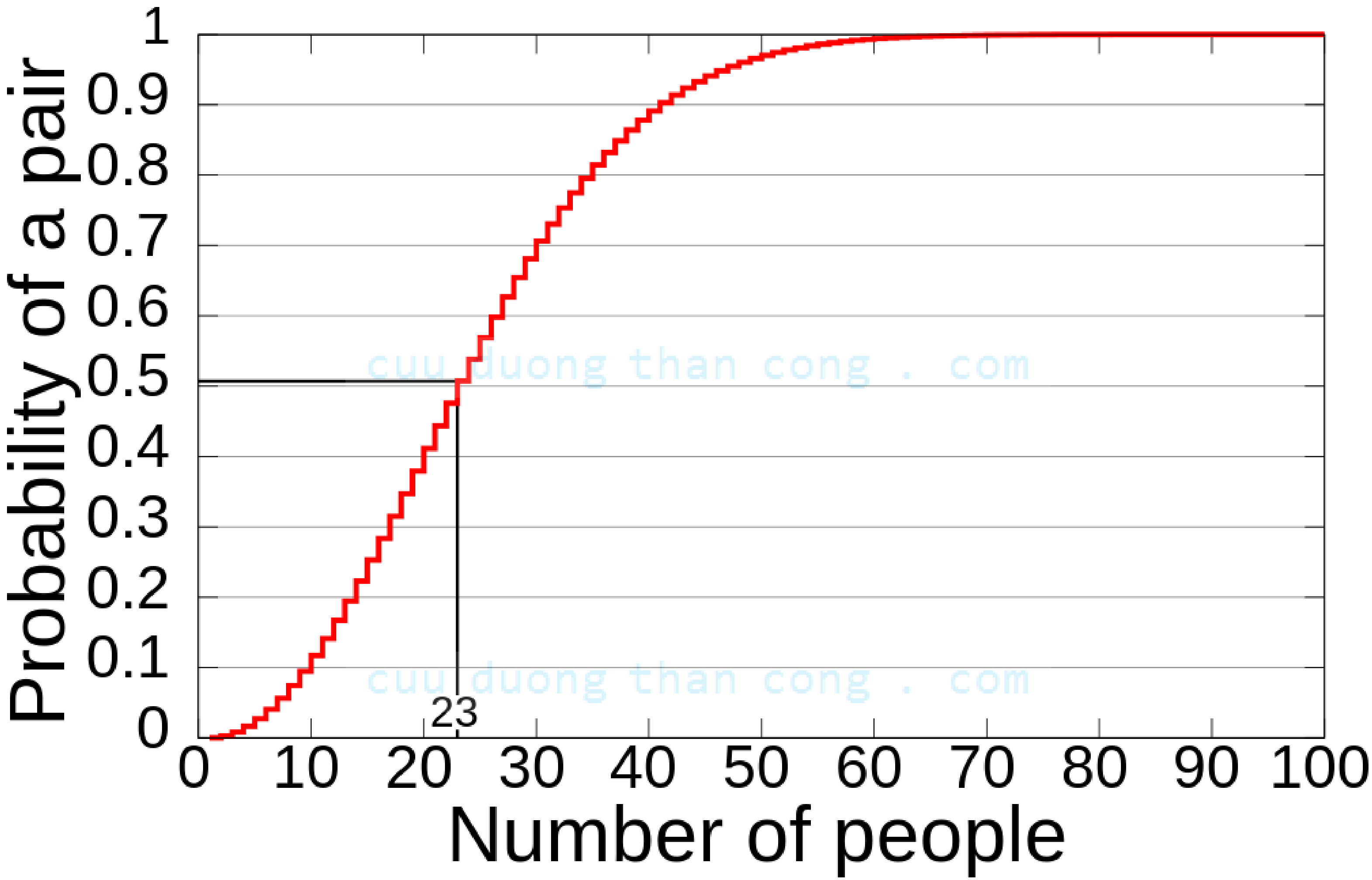
Collision Resistant Attacks

- For a collision resistant attack, an adversary wishes ***to find two messages*** or data blocks, x and y, that yield the same hash function: H(x) = H(y).
- In essence, if we choose random variables from a uniform distribution in the range 0 through N – 1, then the probability that a repeated element is encountered exceeds 0.5 after $N^{1/2}$ choices have been made
- Thus, for an ***m-bit*** hash value, if we pick data blocks at random, we can expect to find two data blocks with the same hash value **within $2^{m/2}$ attempts**

Birthday Attacks

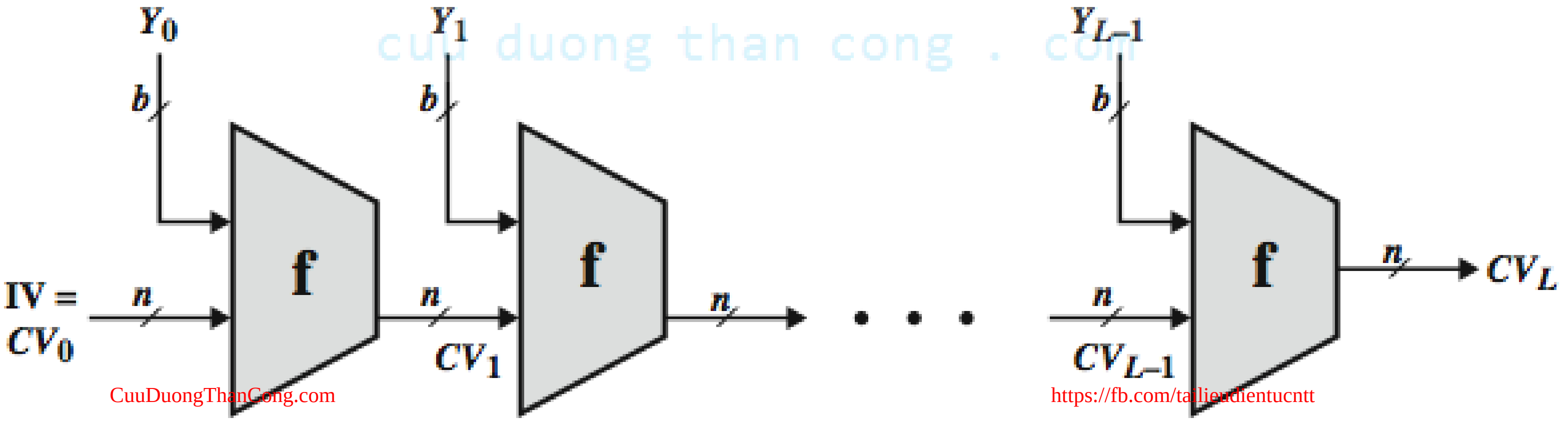
- **might think a 64-bit hash is secure**
- **but by Birthday Paradox is not**
- **birthday attack works thus:**
  - given user prepared to sign a valid message x
  - opponent generates $2^{m/2}$ variations x’ of x, all with essentially the same meaning, and saves them
  - opponent generates $2^{m/2}$ variations y’ of a desired fraudulent message y
  - two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)
  - have user sign the valid message, then substitute the forgery which will have a valid signature
- **conclusion is that need to use larger MAC/hash**

Birthday Attacks



Cryptanalysis Attacks

- As with encryption algorithms, cryptanalytic attacks on hash functions seek ***to exploit some property of the algorithm to perform some attack*** other than an exhaustive search.
- The hash algorithm involves repeated use of a compression function, f, that takes two inputs (an -bit input from the previous step, called the *chaining variable*, and a -bit block) and produces an -bit output



Block Cipher as Hash Functions

- A number of proposals have been made for hash functions based on using a cipher block chaining technique, but without using the secret key.
- ***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 has
  - H_0 = initial value
  - $H_i = E(M_i, H_i\text{-}1)$
  - $G = H_N$
- use final block as the hash value

Secure Hash Functions (SHA)

- **SHA originally designed by NIST & NSA in 1993**
- **was revised in 1995 as SHA-1**
- **US standard for use with DSA signature scheme**
  - standard is FIPS 180-1 1995, also Internet RFC3174
  - Note that, the algorithm is SHA, the standard is SHS
- **based on design of MD4 with key differences**
- **produces 160-bit hash values**
- **recent 2005 results on security of SHA-1 have raised concerns on its use in future applications**

Revised Secure Hash Standard

- NIST issued revision FIPS 180-2 in 2002
- adds 3 additional versions of SHA
  - SHA-256, SHA-384, SHA-512
- designed for compatibility with increased security provided by the AES cipher
- structure & detail is similar to SHA-1
- hence analysis should be similar
- but security levels are rather higher

SHA Versions

	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

Summary

- Cryptographic Hash Functions
- Message Authentication
- Attacks on Hash Functions
  - Brute-Force Attacks
  - Cryptanalysis Attacks
- Secure Hash Algorithm (SHA)

References

1. Cryptography and Network Security, Principles and Practice, William Stallings, Prentice Hall, Sixth Edition, 2013
2. Computer Networking: A Top-Down Approach 6th Edition, Jim Kurose, Keith Ross, Pearson, 2013