SWINBURNE UNIVERSITY OF TECHNOLOGY

Theory of Blockchain

Session 2:

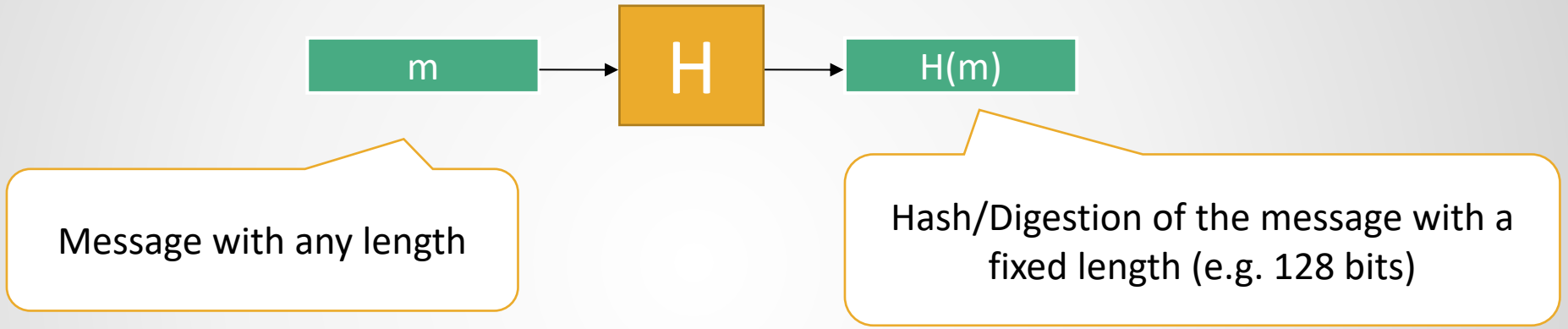
Symmetric Cryptography

Module 3 – Hash Functions and their Applications

Classification of Ciphers

Forget about this classification for a while. Hash functions are not encryption algorithms. They are keyless, but can sometimes be constructed by using block ciphers.

Hash Function



Examples: SHA-1 ،MD5 ...

Hash functions are keyless. A hash function takes any message with any length and turns it into a fixed-length output. This conversion is fast, but calculating the reverse, is computationally hard.

Examples

MD4("The quick brown fox jumps over the lazy dog")
= 1bee69a46ba811185c194762abaeae90
MD4("The quick brown fox jumps over the lazy cog")
= b86e130ce7028da59e672d56ad0113df

MD4 ("a") = bde52cb31de33e46245e05fbdbd6fb24
MD4 ("abc") = a448017aaf21d8525fc10ae87aa6729d
MD4 ("message digest") = d9130a8164549fe818874806e1c7014b
MD4 ("abcdefghijklmnopqrstuvwxyz")
= d79e1c308aa5bbcdeea8ed63df412da9

Properties of a Good Hash Function

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.

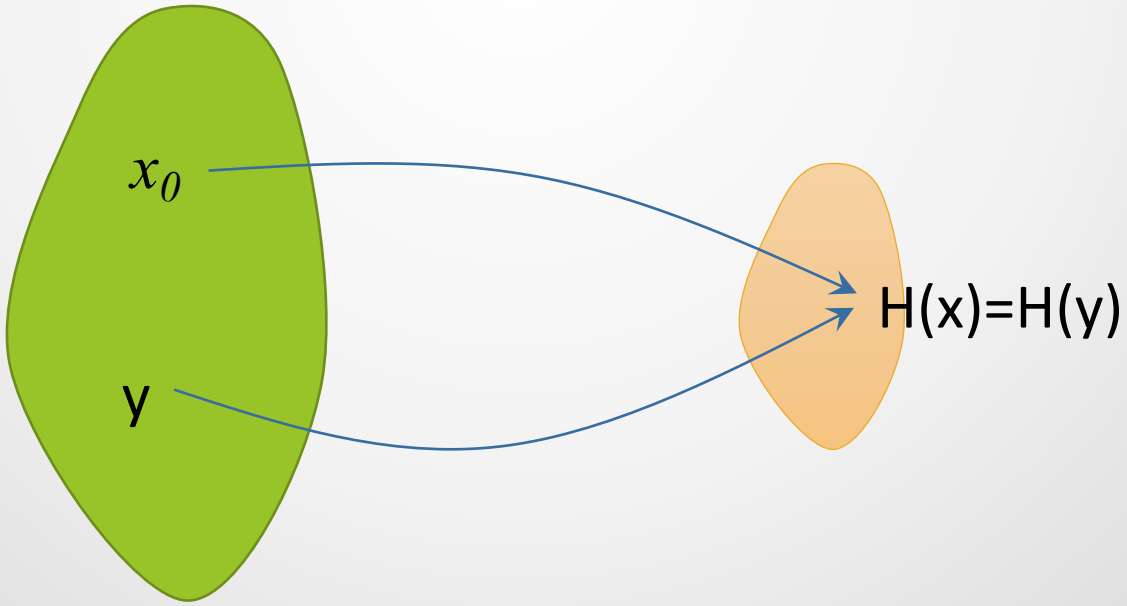
4. For any given code h, it is computationally infeasible to find x such that H(x)=h.

(A hash function with this property is referred to as **one-way** or **preimage resistant**)

Properties of a Good Hash Function

5. For any given block x_0, it is computationally infeasible to find $y \neq x_0$ with $H(y)=H(x_0)$.

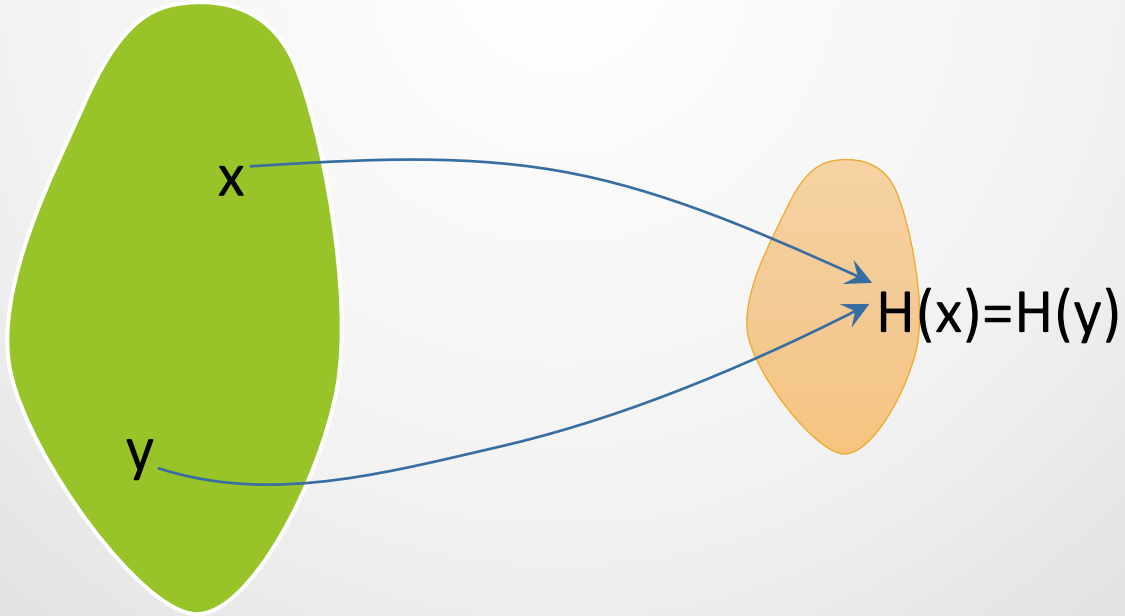
(A hash function with this property is called **second preimage resistant**. This is sometimes referred to as **weak collision resistant**)



Properties of a Good Hash Function

6. It is computationally infeasible to find any pair (x, y) such that $H(x)=H(y)$.

(A hash function with this property is referred to as **collision resistant**. This is sometimes referred to as **strong collision resistant**)



Hash Function Applications

1- OS

2- Message Authentication Codes

3- Blockchain

4- Hash Chains

.

.

.

Hash Application in OS

- User password stored as **H(**password**)**

- When user enters password
  - System computes H(password)
  - Compares with entry in password file

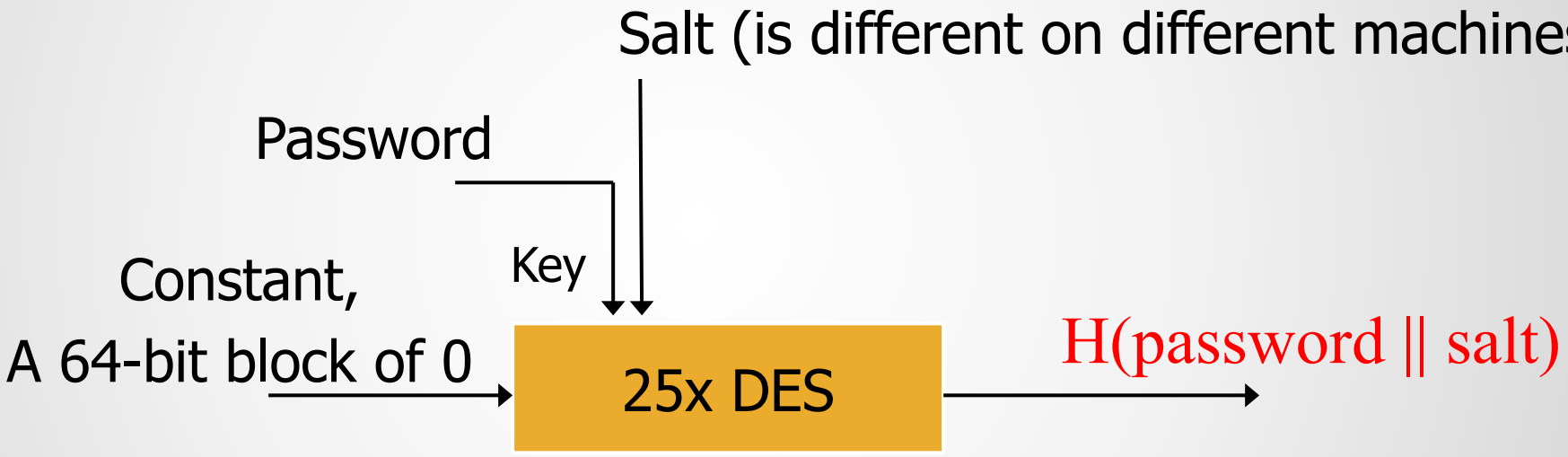
- No passwords stored on disk as plain-text

Example: **Unix/Linux password system**

- Hash function is 25xDES
  - 25 rounds of DES-variant encryptions
- Any user can try “dictionary attack”
- **“Salt”** makes dictionary attack harder

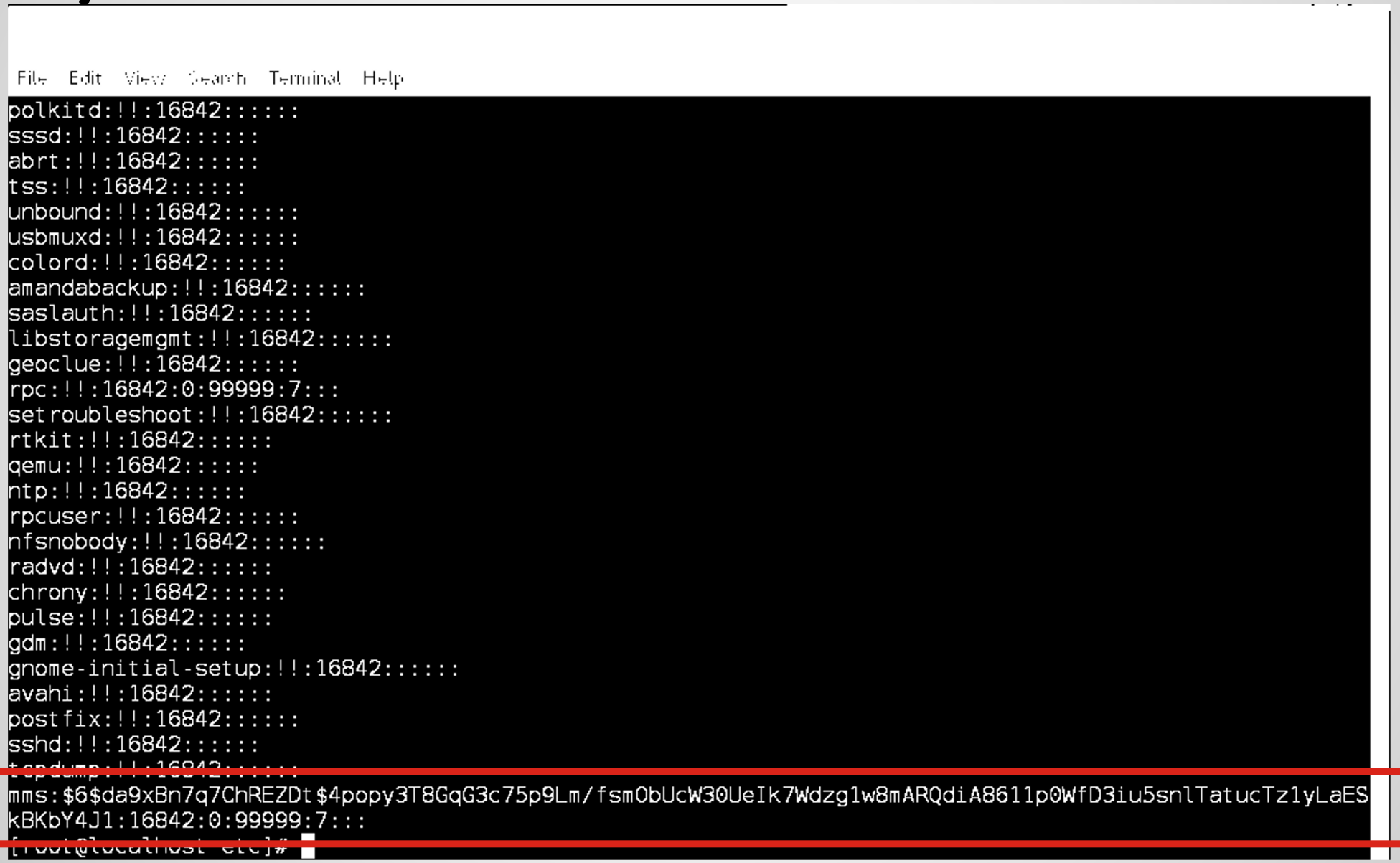
R.H. Morris and K. Thompson, Password security: a case history, Communications of the ACM, November 1979

Unix Password Storage System



When password is set, salt is chosen randomly. 12-bit salt makes dictionary attacks more costly (by a factor of 2^{12}).

/etc/shadow on Centos 7

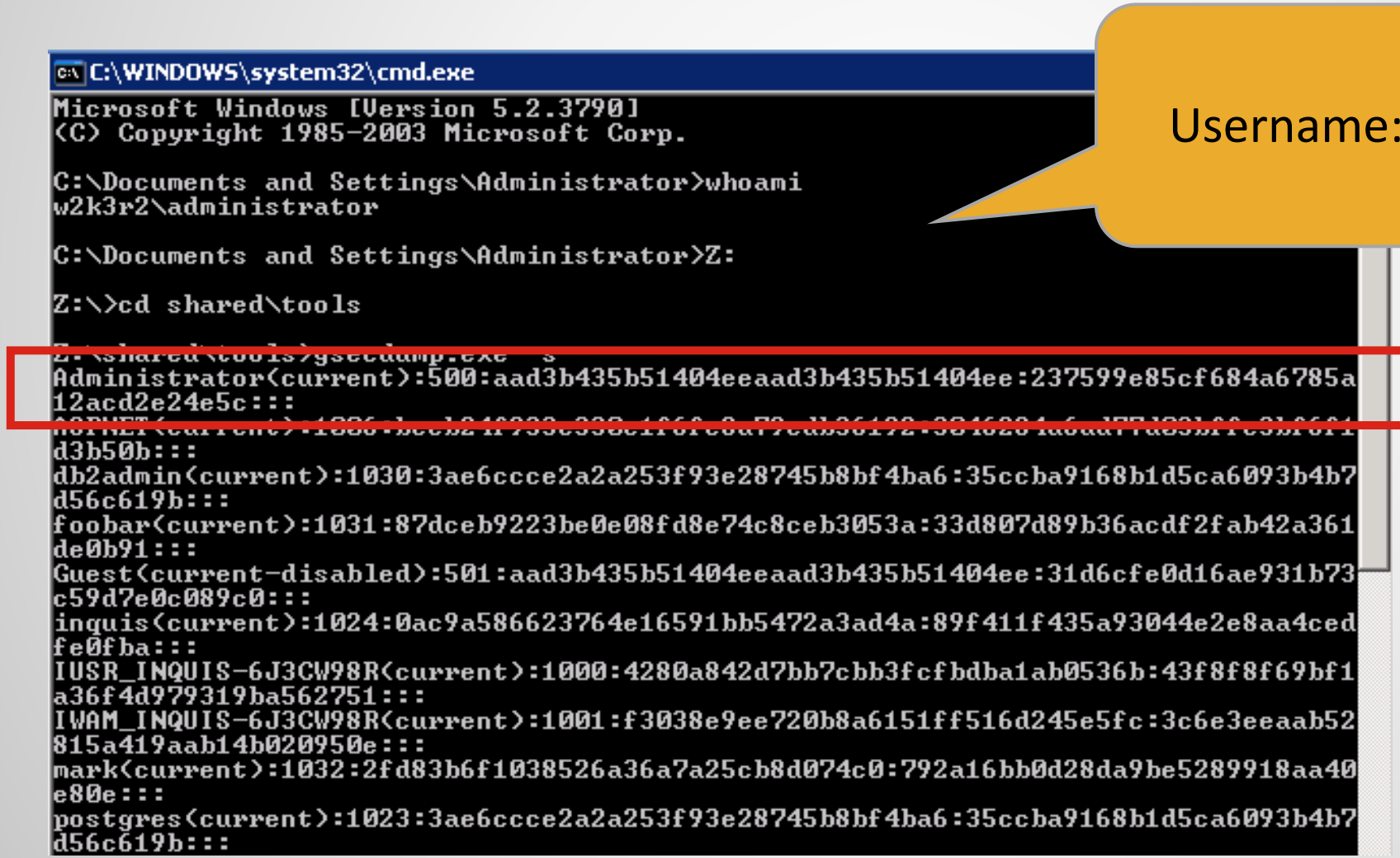


Recent Linux Distributions

- Salt is chosen for each user ID upon creation of that user.
- Salt is not limited to 12 bits anymore.
- Hash function is not DES-based anymore. SHA-512 and MD5 are more dominant.

Windows is no different!

- Hash of the passwords are stored in SAM file C:\windows\system32\config\sam



Old LM-hash vs the new NTLM hash

LM hash is used only for backward compatibility. It's been broken too.

LM hash is computed as follows:

1. The password is padded with NULL bytes to reach 14 characters.
2. The password is converted to all uppercase.
3. The password is split into two 7-byte (56-bit) keys.
4. Each key is used to encrypt a fixed string (that is "KGS!@#$%") by using DES.
5. The two results from step 4 are concatenated and stored as the LM hash.

No Salt in Windows hashes

NT hash is simply a hash.

1. The password is hashed by using the MD4 algorithm and stored.

https://learn.microsoft.com/en-us/windows-server/security/kerberos/passwords-technical-overview

Message Authentication Code (MAC)

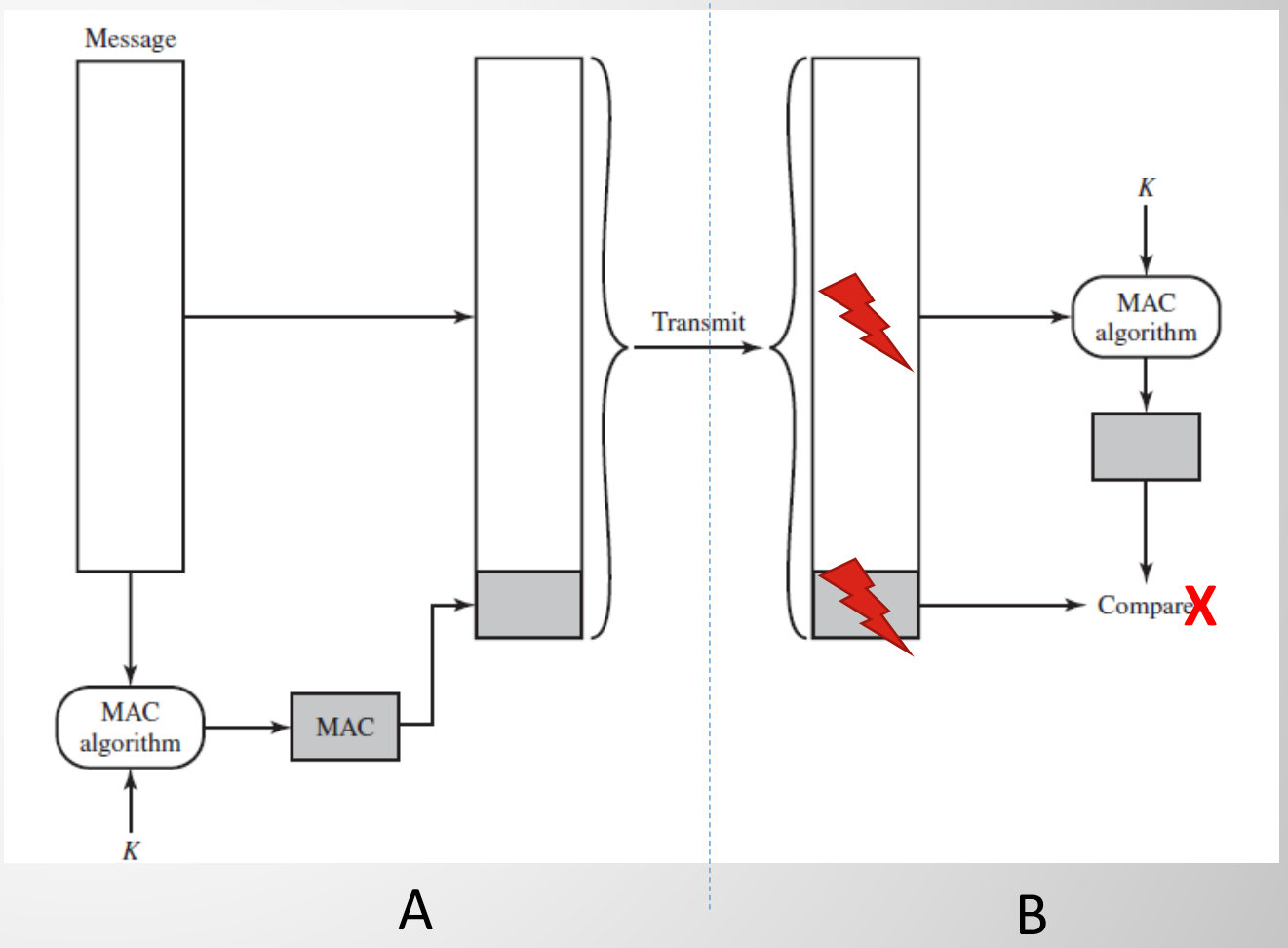
- MAC provides Authentication + Integrity services.
  - But not Confidentiality
- It is sometimes called Message Integrity Code (MIC) to differentiate it from Medium Access Control (MAC).

Message Authentication Code (MAC)

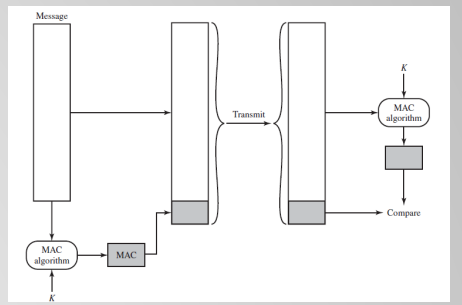
This technique assumes that two communicating parties, say A and share a common secret key K_{AB}.

When A has a message to send to B, it calculates the message authentication code as a function of the message and the key:

$MAC_M = F(K_{AB}, M)$



Points



- Since the key "K_{AB}" is only shared between A and B, if the MAC verifies, the receiver knows it the message must have come from the other party (This means message authentication).

- Verification of MAC implies that the message has not been tampered with. Because if either the message or the MAC part is touched, the receiver will know. Since no one (except A and B) knows the key, it is impossible to change both the message and the MAC so that they match at the receiver.

MAC Algorithm

What is Inside the MAC Box?

It has two parts

1- Message Digestion (i.e. hash)

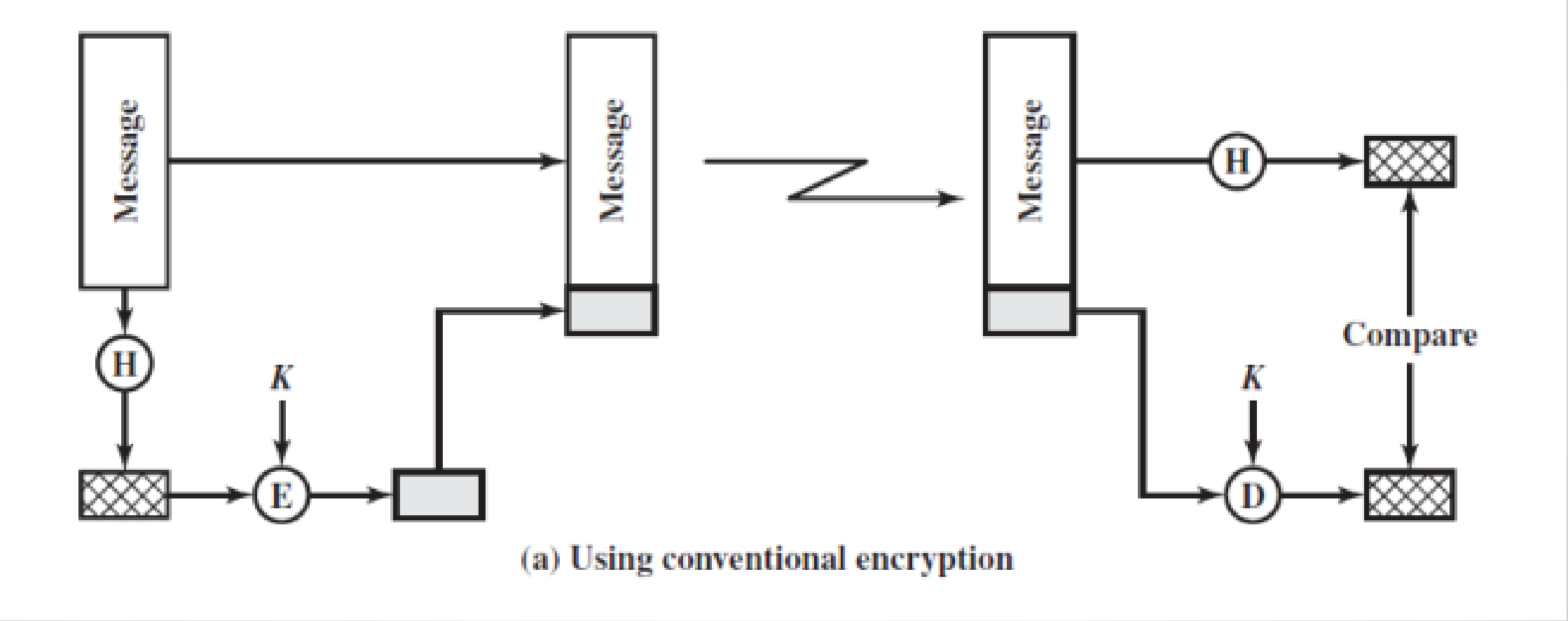
2- Combination of the digestion with the secret key

Because applying the key on the whole message is costly.

It is possible to create a MAC that everyone can verify.

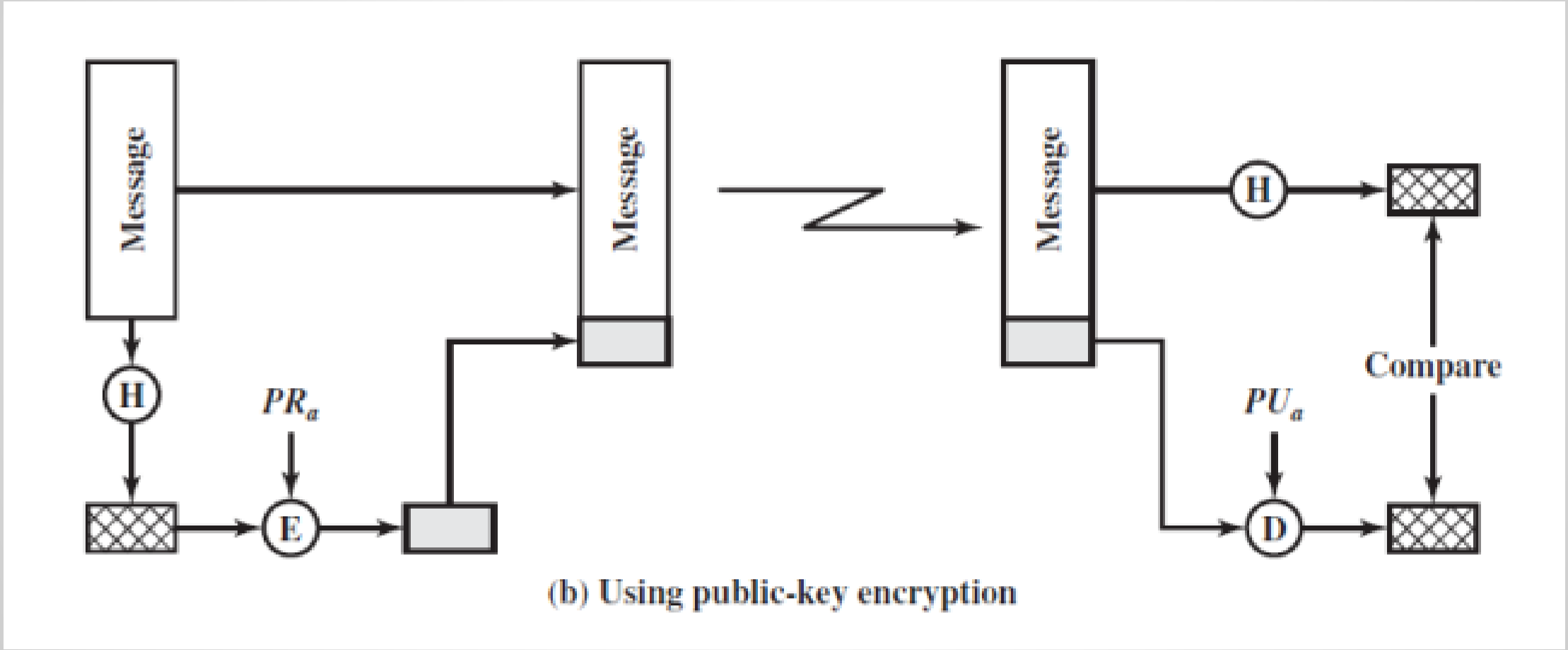
But if everyone has the key, security becomes meaningless. So the key for creation of MAC and verification of MAC must be different.

Making a MAC - Method 1



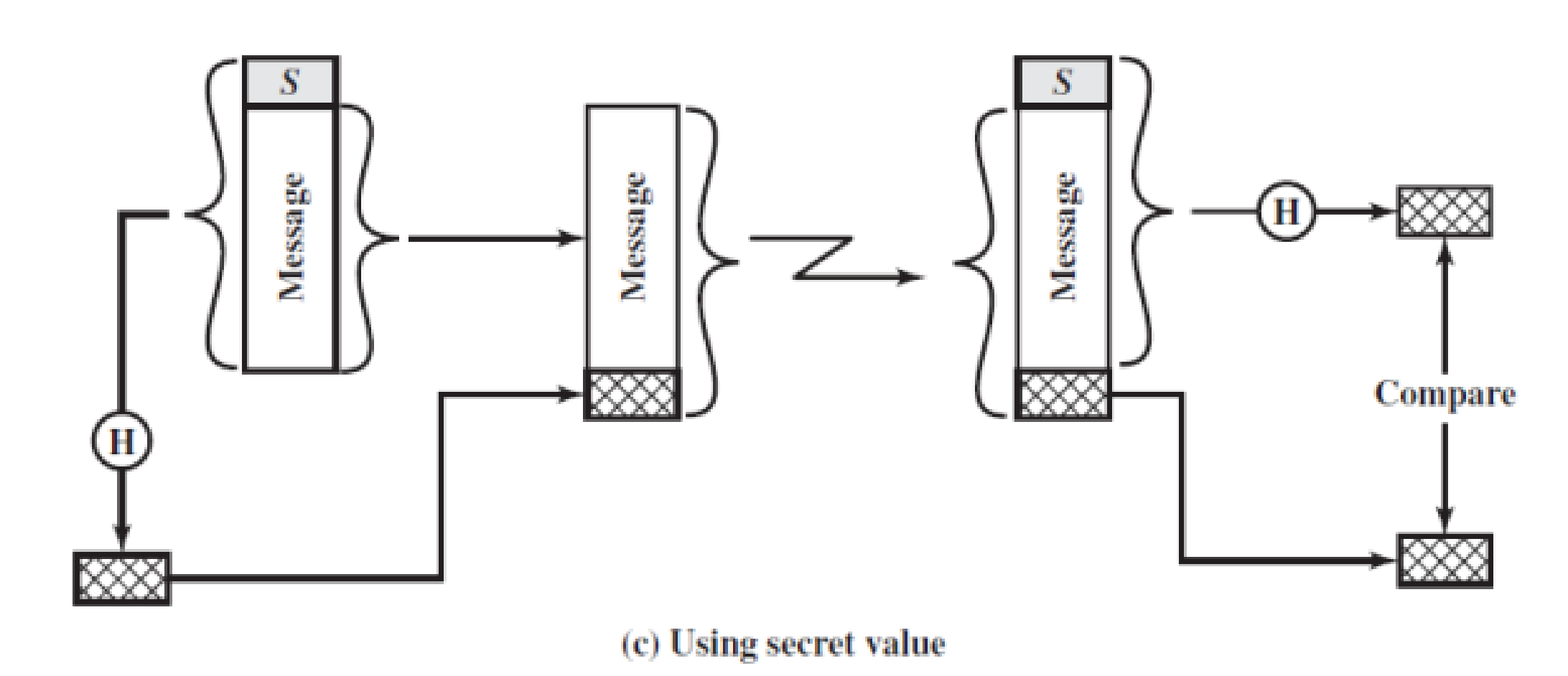
(a) Using conventional encryption

Making a MAC - Method 2 (Dig. Sig.)



(b) Using public-key encryption

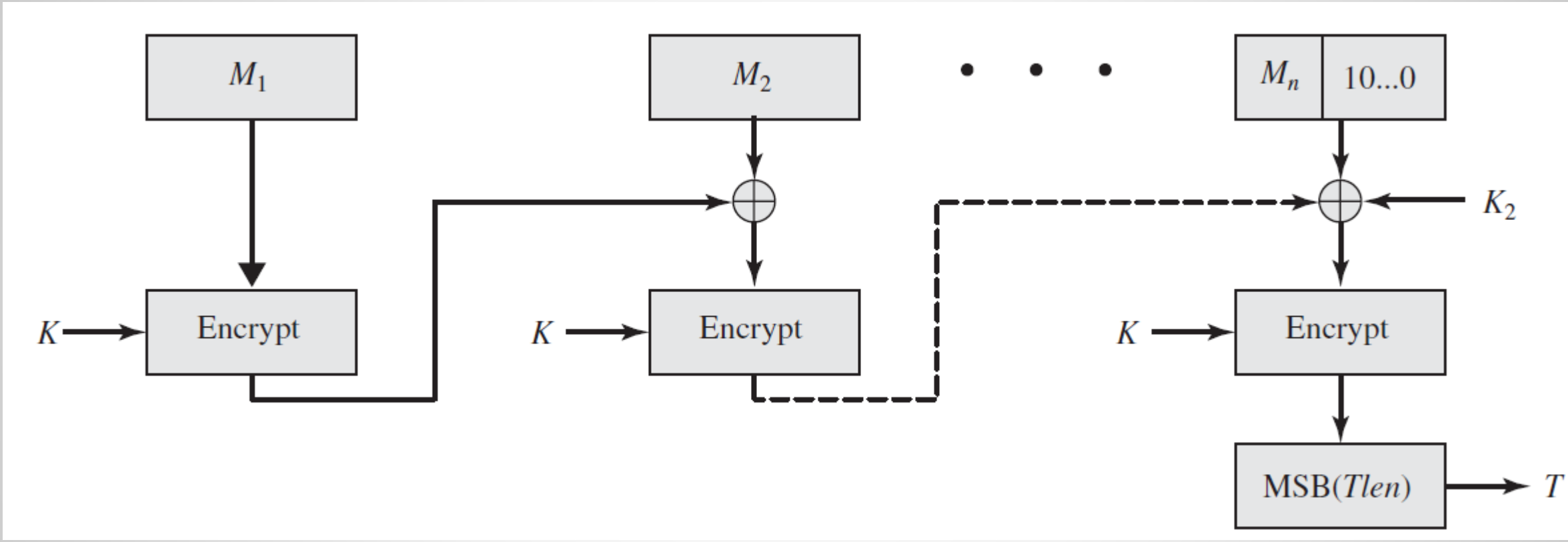
Making a MAC - Method 3



(c) Using secret value

Secret value can be the key

How to make a MAC using a block cipher



$M_1 \ldots M_{n-1}$ are plain text pieces. K and K_2 are encryption and MAC keys, respectively. MSB(Tlen) takes a desired length from output (from the most significant bit).

Famous Hash Functions

- MD4 ،MD5

  MD4 was proposed by Ronald Rivest in 1990. The digest length is 128 bits. It was broken later but it influenced the design of MD5 (RFC1321), SHA-1 and RIPEMD hashing algorithms.

- SHA (1), SHA-2 … SHA-3

  SHA was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993. SHA used MDx designs.

Bitcoin uses SHA-256 (SHA-2), and Ethereum uses Keccak-256 (the base of SHA-3).

What Comes Next …

- We learned about hash functions and their applications.
- We learned how Message Authentication Codes are generated.
- In the next video, we introduce some asymmetric encryption algorithms and use the subjects you learned here to create digital signatures.

See you in the next video …