

Bahan kuliah IF4020 Kriptografi

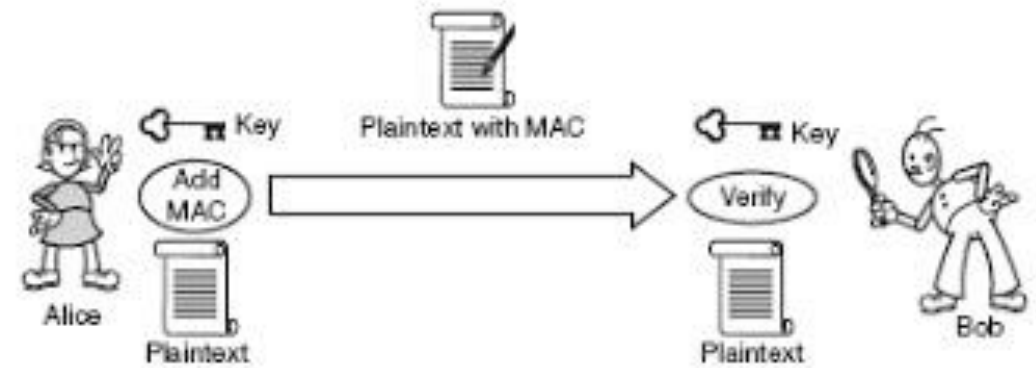


Figure 9.3 Message Authentication Code (MAC)

MAC

(Message Authentication Code)

Oleh: Rinaldi Munir

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
ITB – 2026

Definisi MAC

- MAC (*message authentication code*): kode kecil berukuran tetap (*fixed*) yang dihasilkan dari pesan dan kunci untuk mengotentikasi pengirim dan otentikasi pesan.

$$\text{MAC} = C_K(M)$$

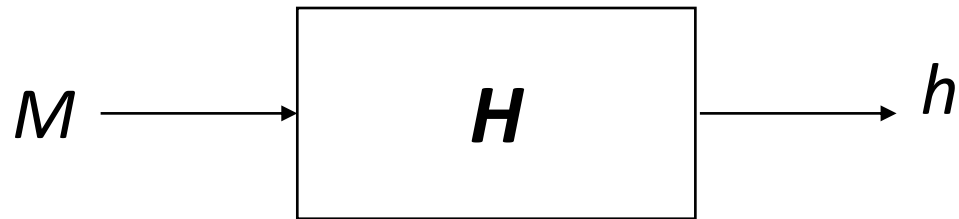
MAC = kode otentikasi

C = algoritma MAC

K = kunci rahasia

- Nama lainnya: *Message Integrity Code* (MIC)

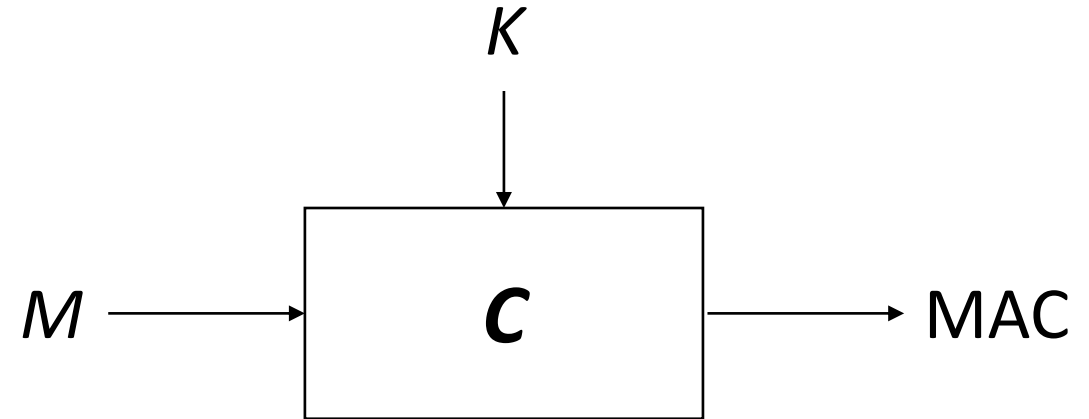
Perbedaan Algoritma MAC dengan Fungsi Hash biasa



$$h = H(M)$$

Message digest dengan fungsi hash

(tidak membutuhkan kunci)



$$\text{MAC} = C_K(M)$$

MAC dengan fungsi hash

(membutuhkan kunci)

- MAC dilekatkan (*embed*) pada pesan sebagai “signature”
- MAC digunakan untuk memeriksa integritas (keaslian) pesan dan otentikasi pengirim.
- Jika MAC yang dikirim sama dengan MAC yang dihitung oleh penerima, maka pesan disimpulkan asli dan pengirim juga benar (karena kunci K hanya diketahui oleh pengirim dan penerima pesan saja)

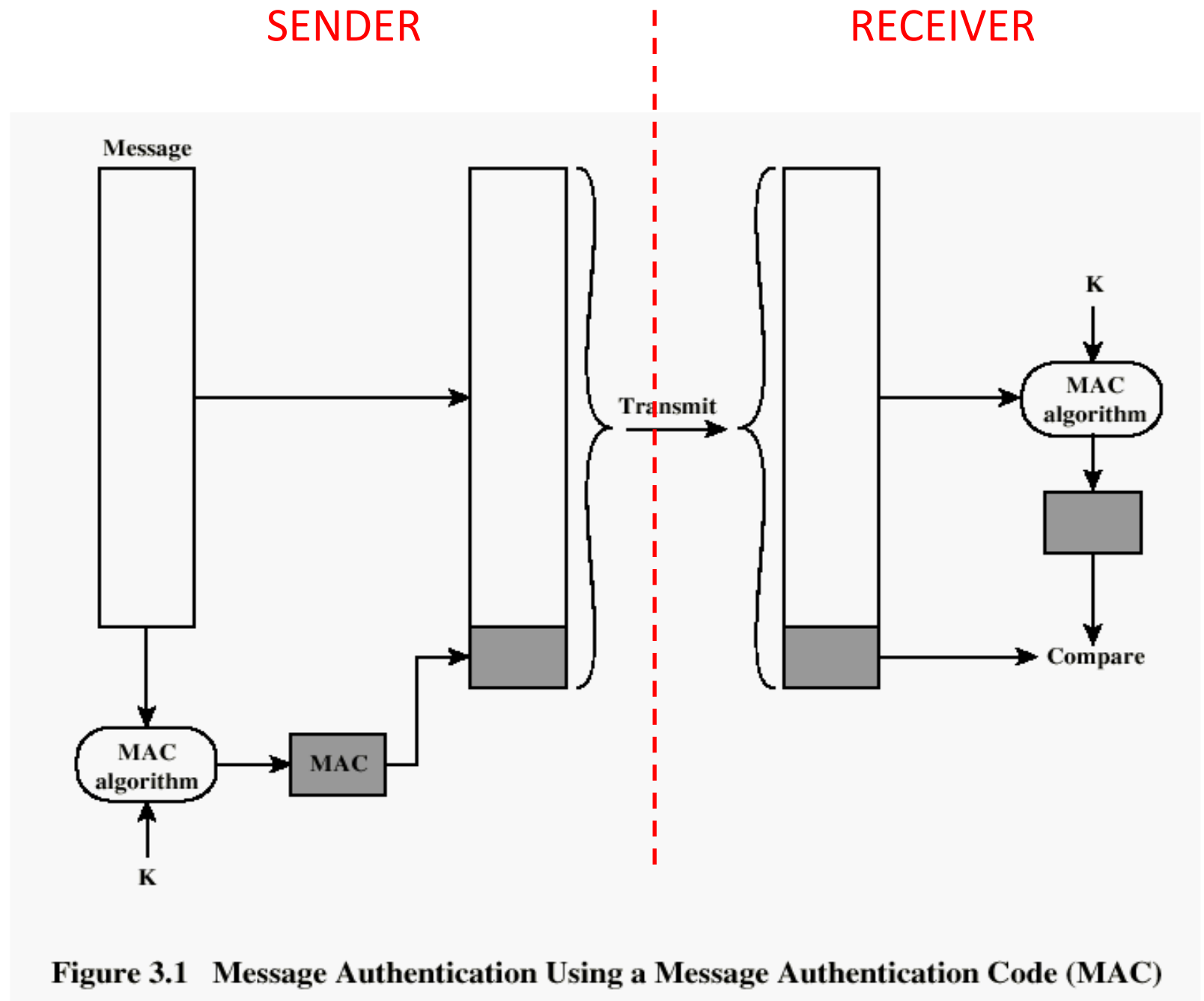


Figure 3.1 Message Authentication Using a Message Authentication Code (MAC)

Kegunaan MAC

1. Seperti *message digest*, MAC digunakan untuk memeriksa keaslian pesan, dokumen, dsb. (*cryptographic checksum*)

→ Menjaga integritas (keaslian) pesan terhadap perubahan oleh pihak lawan, misalnya akibat serangan *hacker*, virus, dsb.

- Jika MAC penerima pesan = MAC pengirim pesan, berarti pesan masih asli.
- Jika pemilik pesan hanya menggunakan fungsi *hash* satu-arah biasa (seperti SHA), maka pihak lawan dapat menghitung *message digest* yang baru dari dokumen yang sudah diubah, lalu mengganti nilai *hash* yang lama dengan nilai *hash* yang baru.

Tetapi, jika digunakan *MAC*, pihak lawan tidak dapat melakukan hal ini karena ia tidak mengetahui kunci yang asli untuk menghitung *MAC*.



Home

Windows

Mac

Linux

Freeware



e.g. Spyware Removal

Windows

Go

Choose Download Location

Norton AntiVirus 2010

You have chosen to download **Norton AntiVirus 2010**. Check the file details to make sure this is the correct program and version, and that your operating system is supported.

Download Details

OPERATING SYSTEMS 7 / XP / VISTA

FILE NAME NAV60TMD.exe

MD5 HASH CE0F5F1BF0F165465BE97BAEB4BD940C

FILE SIZE 85.03 MB

In order to make the download process as fast for you as possible, this file exists on several Tucows Downloads servers around the world. Please choose the location closest to you from which to download the file.

Hacker bisa mengganti file dengan file lain, mengganti nilai *hash* (MD5) semula dengan nilai *hash* yang baru. Pengunduh file tidak dapat menyadarinya.

2. Mengotentikasi pengirim pesan

- Selain memeriksa keaslian pesan, MAC dapat digunakan untuk mengotentikasi pengirim pesan, bahwa pesan memang berasal dari pengirim yang sesungguhnya (asli).
- Hal ini karena hanya pengirim dan penerima pesan yang mengetahui kunci K.
- Jika K pengirim pesan tidak sama dengan K penerima pesan, maka MAC yang dihitung oleh penerima pesan pasti tidak sama dengan MAC yang melekat pada pesan yang diterimanya.

Kapan menggunakan MAC?

- MAC digunakan jika hanya otentikasi pesan dan otentikasi pengirim saja yang diperlukan, sedangkan kerahasiaan pesan tidak ditekankan.
- Namun, jika dikombinasikan dengan enkripsi pesan, maka enkripsi pesan dan perhitungan MAC menggunakan kunci yang terpisah.
- MAC dapat dihitung setelah pesan dienkripsi atau sebelum pesan dienkripsi. Umumnya lebih baik MAC dihitung sebelum pesan dienkripsi.
- Perlu dicatat bahwa MAC tidak sama dengan “*digital signature*” karena ia tidak menyediakan layanan nir-penyangkalan (*non-repudiation*).
- *Digital signature* akan dibahas pada pokok bahasan berikutnya

Algoritma MAC

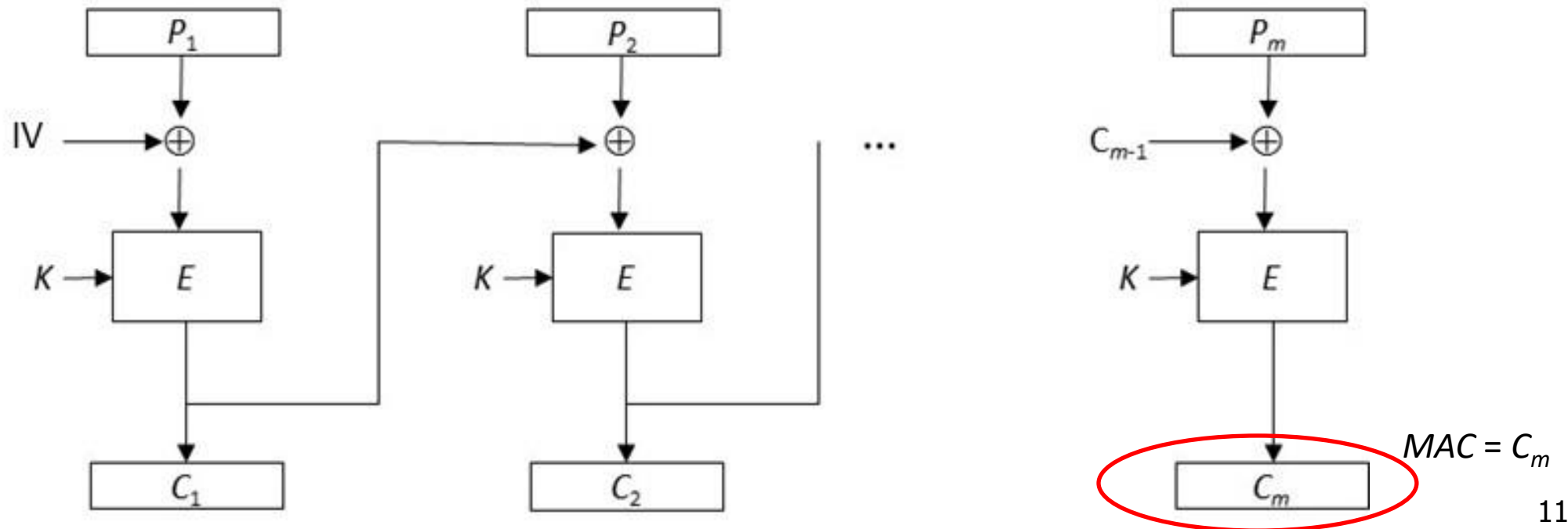
- Sebagaimana fungsi *hash*, MAC dihasilkan dengan oleh algoritma yang melakukan kompresi pesan yang bersifat *irreversible*.
- Algoritma MAC adalah fungsi banyak-ke-satu (*many-to-one function*), artinya potensial banyak pesan memiliki MAC yang sama
- Namun menemukan dua atau lebih pesan memiliki MAC yang sama adalah sangat sulit.
- Persyaratan algoritma MAC:
 1. Diberikan sebuah pesan dan MAC, maka sangat sukar menemukan pesan lain yang memiliki MAC yang sama.
 2. MAC dari berbagai pesan seharusnya terdistribusi secara *uniform*
 3. MAC seharusnya bergantung merata pada semua bit-bit pesan

- MAC dapat dihasilkan dengan menggunakan:
 1. Algoritma MAC berbasis *block cipher*
 2. Algoritma MAC berbasis fungsi *hash* satu arah yang sudah ada

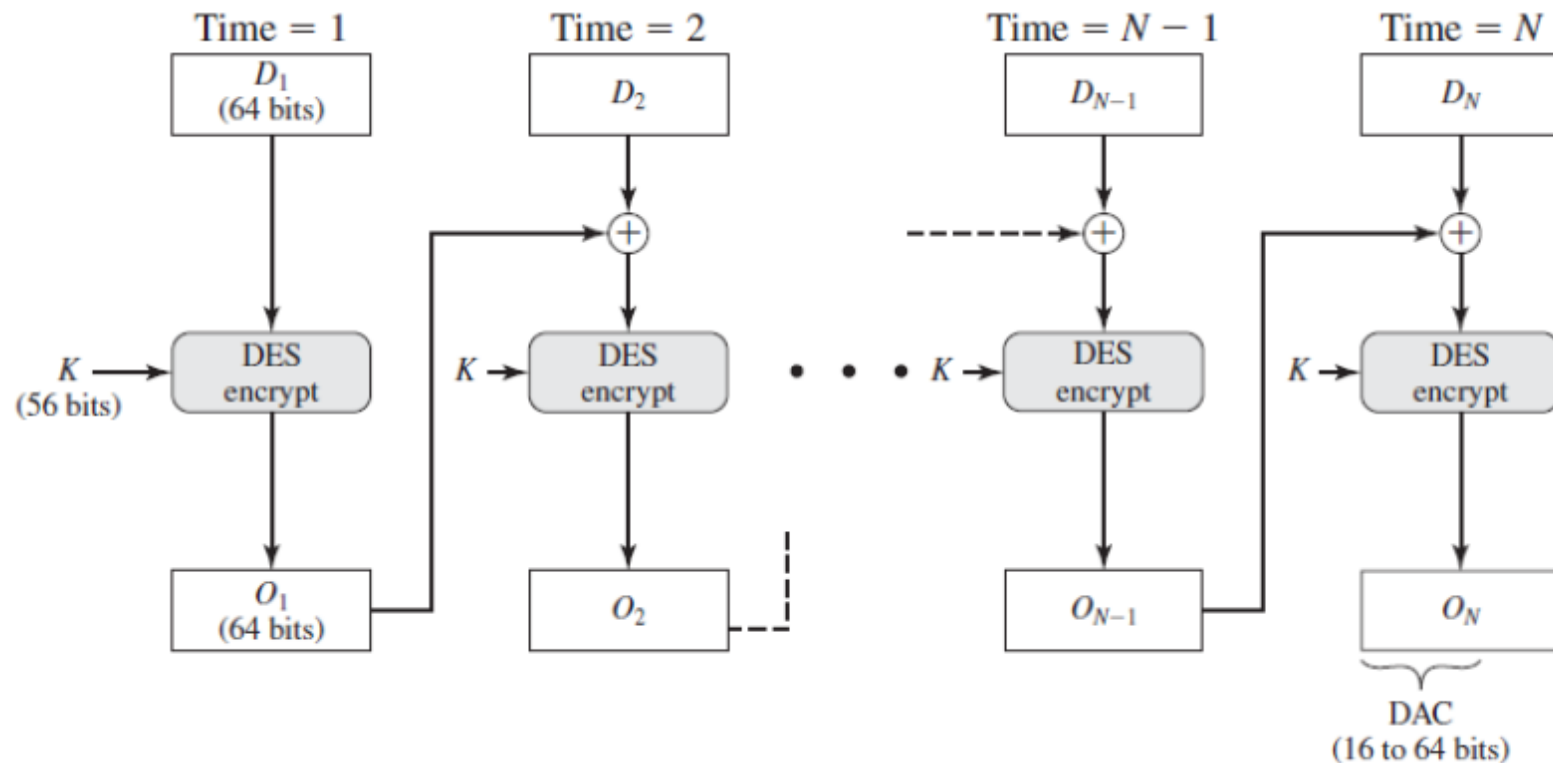
Algoritma MAC

(a) Algoritma MAC berbasis *block cipher*

- *MAC* dibangkitkan dari *block cipher* dengan mode *CBC* atau *CFB*.
- Nilai *hash*-nya (yang menjadi *MAC*) adalah hasil enkripsi blok terakhir.



- Misalkan *DES* digunakan sebagai *cipher* blok, maka MAC = ukuran blok = 64 bit, dan kunci rahasia *MAC* adalah kunci DES yang panjangnya 56 bit.
- *Data Authentication Algorithm (DAA)* adalah algoritma MAC berbasis *DES-CBC* yang digunakan secara luas:



(b) Algoritma MAC berbasis fungsi *hash* satu-arah

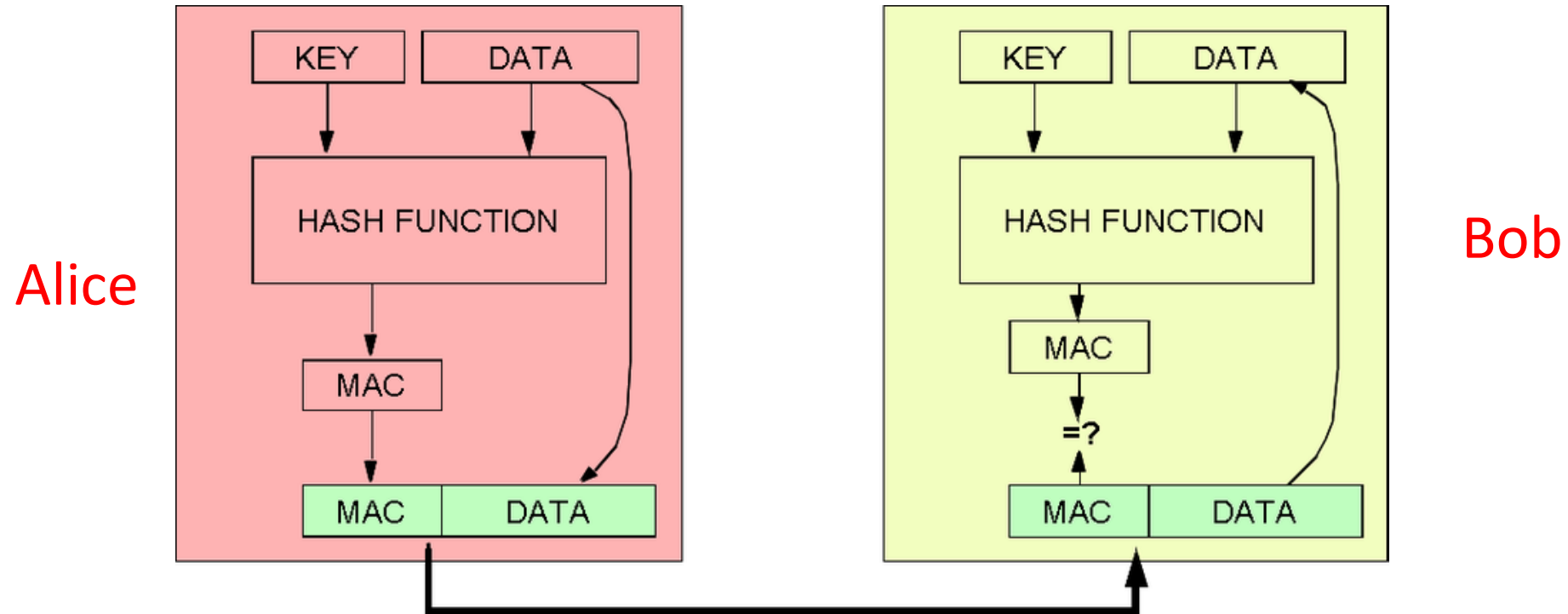
- Fungsi *hash* seperti *MD5* dan *SHA* dapat digunakan sebagai *MAC*
- Pesan *M* disambung (*concat*) dengan kunci *K*, lalu dihitung nilai hash dari hasil penggabungan tersebut dengan fungsi hash *H* seperti *MD5* atau *SHA*

$$\text{MAC} = H(M\|K)$$

ket: ' $\|$ ' adalah simbol *concatenation*

- Panjang *MAC* tergantung dari fungsi *hash* yang digunakan. Jika menggunakan fungsi *SHA-1*, maka *MAC* yang dihasilkan adalah 160 bit

Misalkan Alice dan Bob akan saling bertukar DATA. Alice dan Bob telah berbagi sebuah kunci rahasia *KEY*.



HMAC

- HMAC adalah algoritma pembangkitan MAC menggunakan fungsi *hash* dan kunci.
- Dinotasikan dengan HMAC-x, x adalah nama fungsi hash yang digunakan, misalnya HMAC-SHA1, HMAC-MD5, HMAC-SHA2, dsb.
- Defenisi dari RFC 2014:

$$\text{HMAC}(K, m) = H \left((K' \oplus \text{opad}) \parallel H \left((K' \oplus \text{ipad}) \parallel m \right) \right)$$
$$K' = \begin{cases} H(K) & \text{if } K \text{ is larger than block size} \\ K & \text{otherwise} \end{cases}$$

where

H is a cryptographic hash function.

m is the message to be authenticated.

K is the secret key.

K' is a block-sized key derived from the secret key, K ; either by padding to the right with 0s up to the block size, or by hashing down to less than or equal to the block size first and then padding to the right with zeros.

$\parallel$ denotes [concatenation](#).

$\oplus$ denotes bitwise [exclusive or](#) (XOR).

opad is the block-sized outer padding, consisting of repeated bytes valued 0x5c.

ipad is the block-sized inner padding, consisting of repeated bytes valued 0x36.^[3]

Sumber: Wikipedia

```

function hmac is
  input:
    key:      Bytes    // Array of bytes
    message:  Bytes    // Array of bytes to be hashed
    hash:     Function // The hash function to use (e.g. SHA-1)
    blockSize: Integer // The block size of the hash function (e.g. 64 bytes for SHA-1)
    outputSize: Integer // The output size of the hash function (e.g. 20 bytes for SHA-1)

  // Compute the block sized key
  block_sized_key = computeBlockSizedKey(key, hash, blockSize)

  o_key_pad ← block_sized_key xor [0x5c blockSize] // Outer padded key
  i_key_pad ← block_sized_key xor [0x36 blockSize] // Inner padded key

  return hash(o_key_pad || hash(i_key_pad || message))

function computeBlockSizedKey is
  input:
    key:      Bytes    // Array of bytes
    hash:     Function // The hash function to use (e.g. SHA-1)
    blockSize: Integer // The block size of the hash function (e.g. 64 bytes for SHA-1)

  // Keys longer than blockSize are shortened by hashing them
  if (length(key) > blockSize) then
    key = hash(key)

  // Keys shorter than blockSize are padded to blockSize by padding with zeros on the right
  if (length(key) < blockSize) then
    return Pad(key, blockSize) // Pad key with zeros to make it blockSize bytes long

  return key

```

COM

Search tools...

Free Online Tools For Developers



Encoder / Decoder / Hash / HMAC Generator / Tester Tool

Encoders - Cryptography / HMAC Generator

Computes a Hash-based message authentication code (HMAC) using a secret key. A HMAC is a small set of data that helps authenticate the nature of message; it protects the integrity and the authenticity of the message.

The secret key is a unique piece of information that is used to compute the HMAC and is known both by the sender and the receiver of the message. This key will vary in length depending on the algorithm that you use.

I use [Bouncy Castle](#) for the implementation.

Copy-paste the string here

Kita bikin romantis
Bikin paling romantis

Secret key

MALIQ

Digest algorithm

SHA256

Compute HMAC

-Computed HMAC-

4f1e8af74ed79da8220d7e60febaa375483d38021da884774c10e994a356515c

Copy

Save

Contoh:

$M = \textit{Halo, Bob!}$

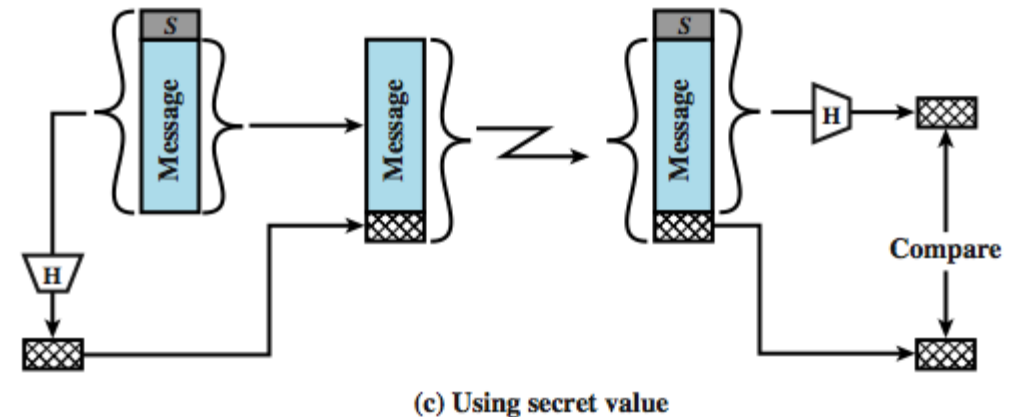
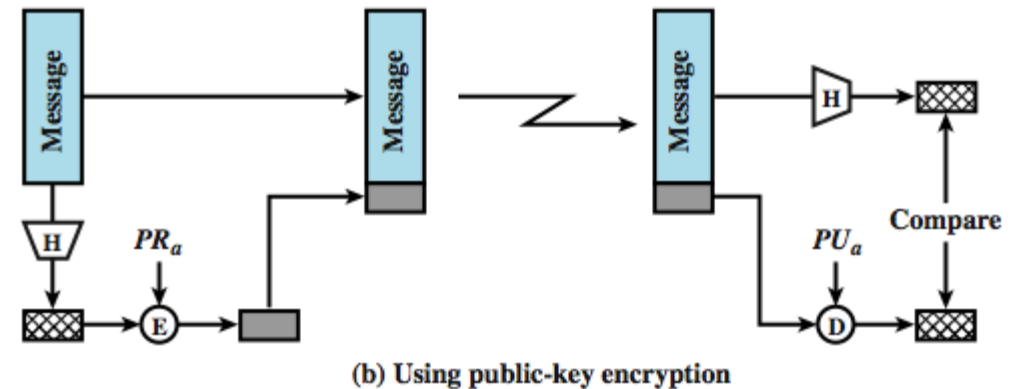
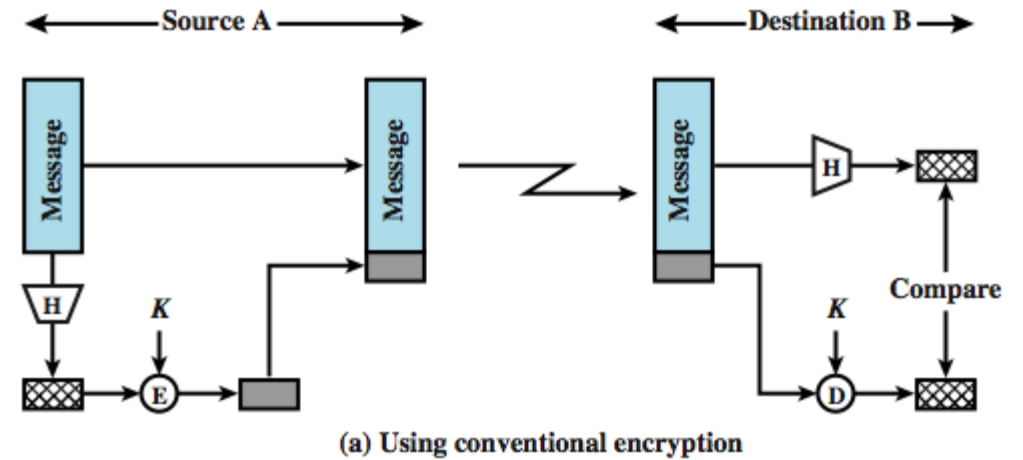
$K = 12345678$

Fungsi Hash: SHA-1

$MAC = 6f8605c7c3a649a40abfb87b44aa21f356e931a0$

Sumber: MAC online <https://www.freeformatter.com/hmac-generator.html>

- Selain menggunakan MAC, terdapat teknik lain untuk mengotentikasi pesan menggunakan kunci
- (a) Menggunakan fungsi *hash* dan enkripsi simetri
- (b) Menggunakan enkripsi dengan kriptografi kunci publik
- (c) Menggunakan nilai rahasia S yang disepakati oleh pengirim dan penerima pesan



SELAMAT BELAJAR