Otentikasi Pesan dengan Fungsi Hash dan MAC

Otentikasi Pesan dan Pengirim

- Otentikasi berhubungan dengan keaslian pesan dan keaslian pengirim pesan

- Otentikasi pesan: memastikan bahwa pesan masih asli, utuh, tidak mengalami perubahan selama ditransmisikan ke penerima pesan.

- Otentikasi pengirim: memastikan bahwa pesan berasal dari pengirim yang asli atau pengirim yang sesungguhnya, bukan dari pihak lain

Ingat Kembali empat layanan kriptografi

1. Kerahasiaan pesan (*Confidentiality/privacy/secrecy*)

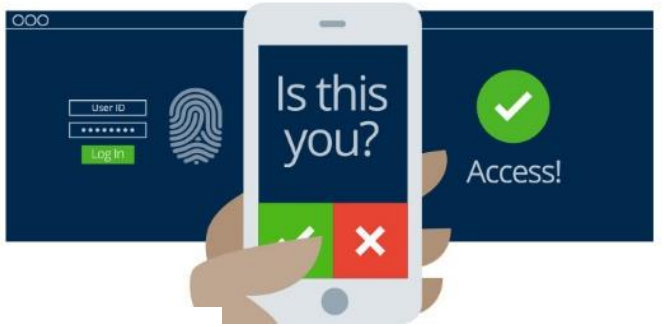
   → dengan mengenkripsi pesan

2. Keaslian pesan (*Data integrity*)

   → menggunakan fungsi hash atau MAC

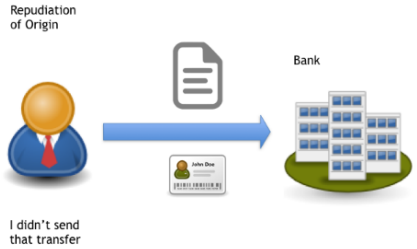
3. Keaslian pengirim dan penerima pesan (*Authentication*)

   → menggunakan MAC atau digital signature

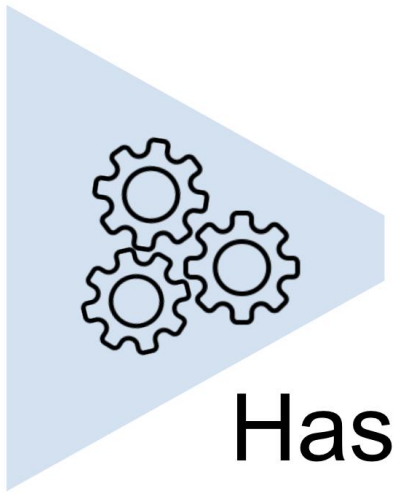


4. Anti penyangkalan (*Non-repudiation*)

   → mengguanakn digital signature



Fungsi Hash



Hash

dfd879...f8d2f4

Fungsi Hash

- Fungsi yang mengkompresi pesan (*M*) berukuran sembarang menjadi *string* (*h*) yang berukuran kecil dan tetap (*fixed*)
- Luaran (*output*) fungsi *hash* tersebut dinamakan **pesan ringkas** (*message-digest)* atau nilai *hash* (*hash value*)
- *Irreversible* (tidak bisa dikembalikan menjadi pesan semula)

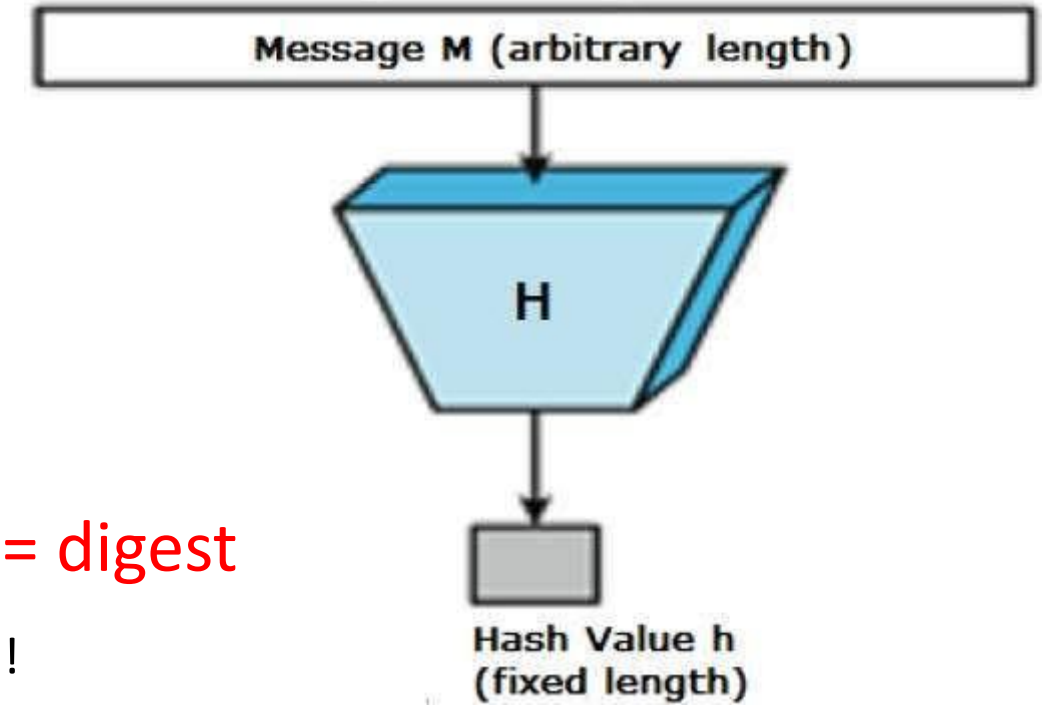
Fungsi Hash:

$$h = H(M)$$

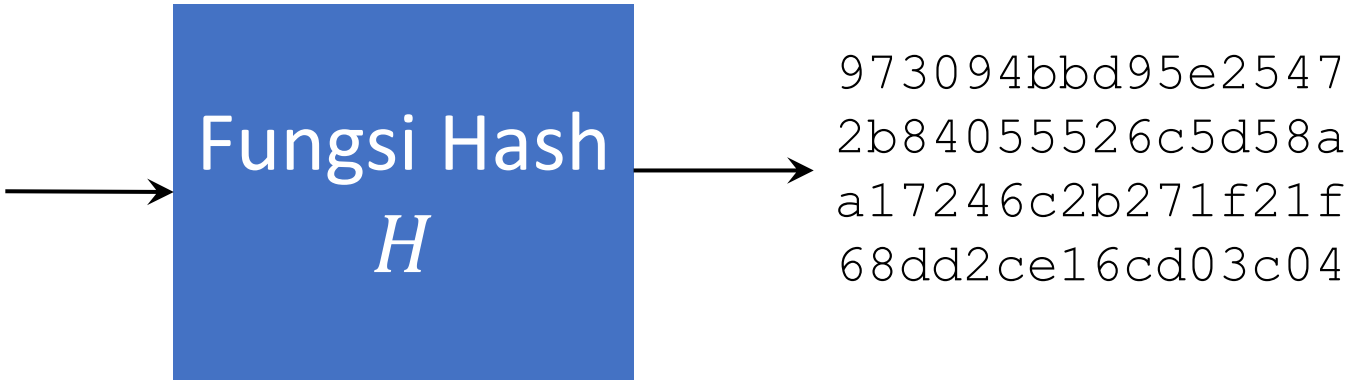
$|h| <<<< |M|$

h = Hash value = message digest = digest

Contoh: *size*(*M*) = 1 MB → *size*(*h*) = 256 bit !!!!

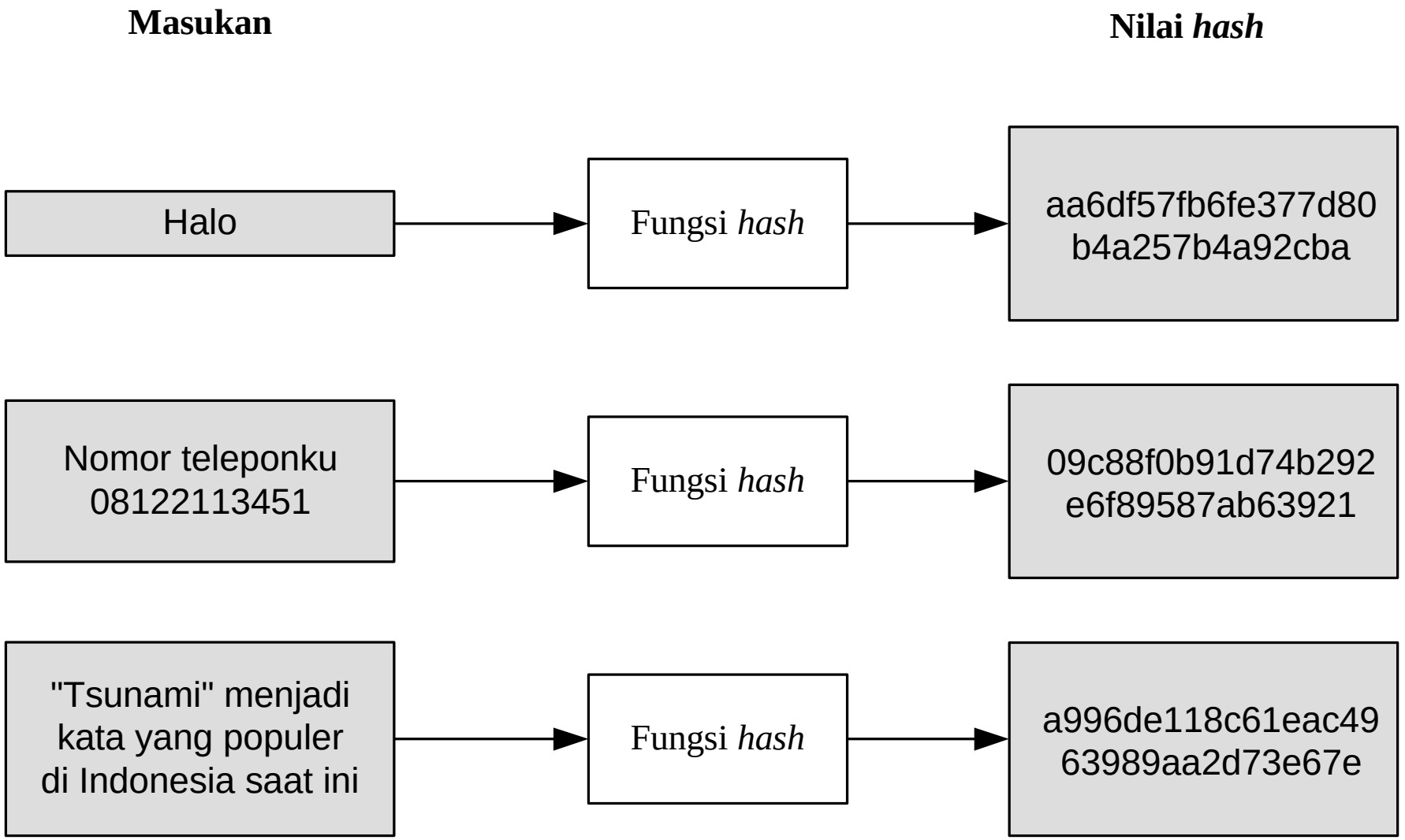


Tolong kirimkan kapal ke pulau Atol. Ada gunung berapi meletus, penduduk pulau harus diungsikan. Kalau perlu dua kapal dikirim berikut makanan buat pengungsi. Nanti malam saya kontak lagi

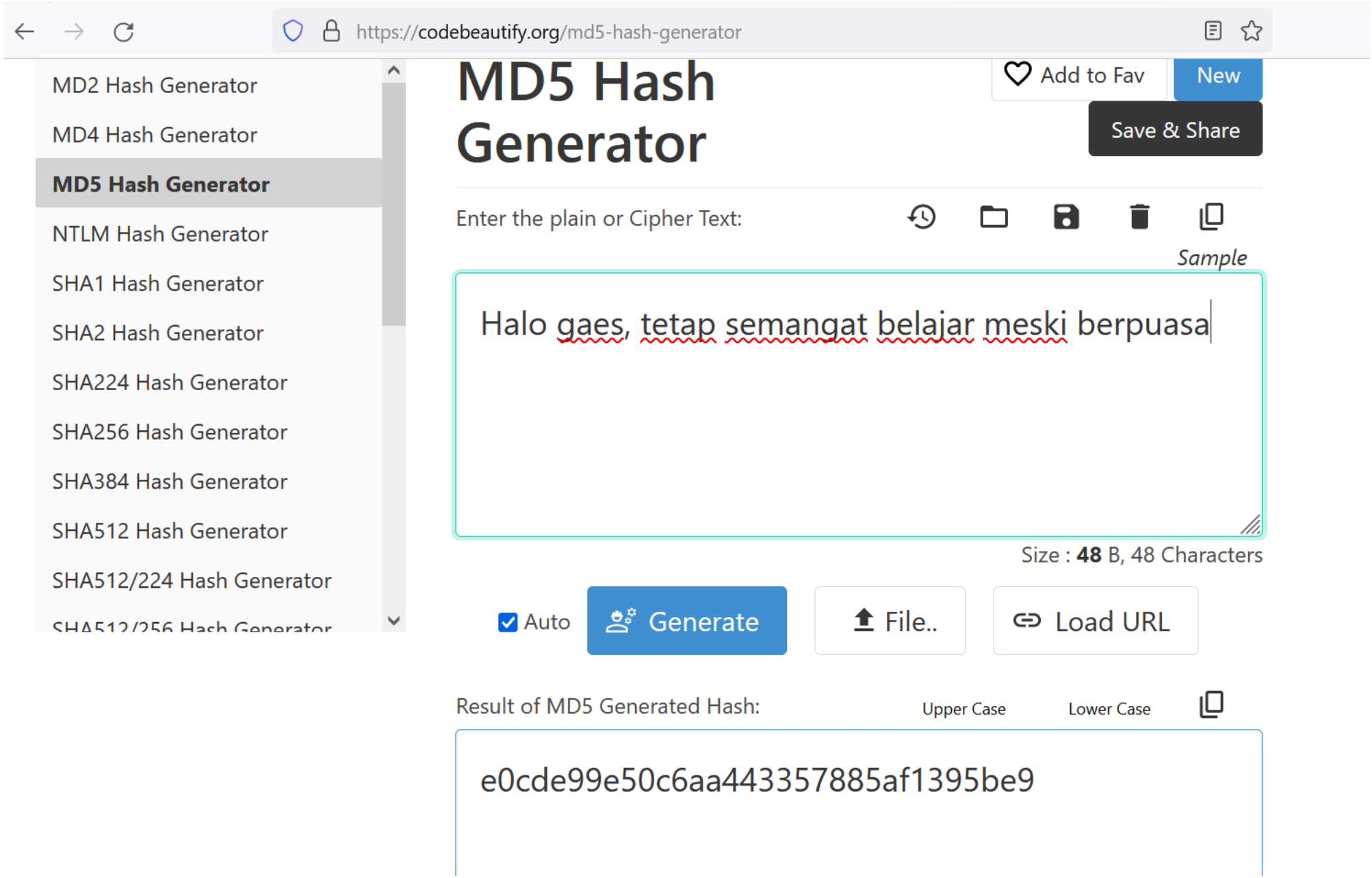


Pesan input

Nilai hash
(256 bit = 64 karakter hexadecimal)


Masukan
Nilai *hash*
Halo
Fungsi *hash*
aa6df57fb6fe377d80
b4a257b4a92cba
Nomor teleponku
08122113451
Fungsi *hash*
09c88f0b91d74b292
e6f89587ab63921
"Tsunami" menjadi
kata yang populer
di Indonesia saat ini
Fungsi *hash*
a996de118c61eac49
63989aa2d73e67e

https://codebeautify.org/md5-hash-generator



Sifat-sifat fungsi *hash H*:

a) collision resistance : sangat sukar menemukan dua pesan *a* dan *b* sedemikian sehingga $H(a) = H(b)$

b) preimage resistance: untuk sembarang nilai hash *y*, sukar menemukan pesan *a* sedemikian sehingga $H(a) = y$

c) second preimage resistance – untuk pesan *a* dan nilai hash $y = H(a)$, sukar menemukan pesan kedua *b* sedemikian sehingga $H(b) = y$

- Ingat: Fungsi *hash* satu arah tidak tepat disebut sebagai sebuah fungsi enkripsi, meskipun nilai *hash* tidak memiliki makna,

- sebab, nilai *hash* tidak dapat ditransformasi balik menjadi pesan semula.

- Alasan lainnya, proses *hashing* tidak menggunakan kunci.

- Cukup banyak fungsi *hash* yang terdapat di dalam kriptografi:

Name	Length
BLAKE-256	256 bits
BLAKE-512	512 bits
BLAKE2s	up to 256 bits
BLAKE2b	up to 512 bits
BLAKE2X	arbitrary
BLAKE3	arbitrary
ECOH	224 to 512 bits
FSB	160 to 512 bits
GOST	256 bits
Grøstl	up to 512 bits
HAS-160	160 bits
HAVAL	128 to 256 bits
JH	224 to 512 bits
LSH[19]	256 to 512 bits
MD2	128 bits
MD4	128 bits
MD5	128 bits
MD6	up to 512 bits
RadioGatún	arbitrary
RIPEMD	128 bits
RIPEMD-128	128 bits
RIPEMD-160	160 bits
RIPEMD-256	256 bits
RIPEMD-320	320 bits
SHA-1	160 bits
SHA-224	224 bits
SHA-256	256 bits
SHA-384	384 bits
SHA-512	512 bits
SHA-3 (subset of Keccak)	arbitrary
Skein	arbitrary
Snefru	128 or 256 bits
Spectral Hash	512 bits
Streebog	256 or 512 bits
SWIFFT	512 bits
Tiger	192 bits
Whirlpool	512 bits

Aplikasi Fungsi *Hash* Satu-Arah

1. Menjaga integritas pesan

   - Fungsi *hash* sangat peka terhadap perubahan 1 bit pada pesan

   - Pesan berubah 1 bit, nilai *hash* berubah sangat signifikan.

   - Bandingkan nilai *hash* baru dengan nilai *hash* lama. Jika sama, pesan masih asli. Jika tidak sama, pesan sudah dimodifikasi

Contoh:

(i) Pesan (berupa *file*) asli

Pada bulan Oktober 2004 ini, suhu udara kota Bandung terasa lebih panas dari hari-hari biasanya. Menurut laporan Dinas Meteorologi Kota Bandung, suhu tertinggi kota Bandung adalah 33 derajat Celcius pada Hari Rabu, 17 Oktober yang lalu. Suhu terseut sudah menyamai suhu kota Jakarta pada hari-hari biasa. Menurut Kepala Dinas Meteorologi, peningkatan suhu tersebut terjadi karena posisi bumi sekarang ini lebih dekat ke matahari daripada hari-hari biasa.

Sebutan Bandung sebagai kota sejuk dan dingin mungkin tidak lama lagi akan tinggal kenangan. Disamping karena faktor alam, jumlah penduduk yang padat, polusi dari pabrik di sekita Bandung, asap knalpot kendaraan, ikut menambah kenaikan suhu udara kota.

Nilai MD5: **2F82D0C845121B953D57E4C3C5E91E63**

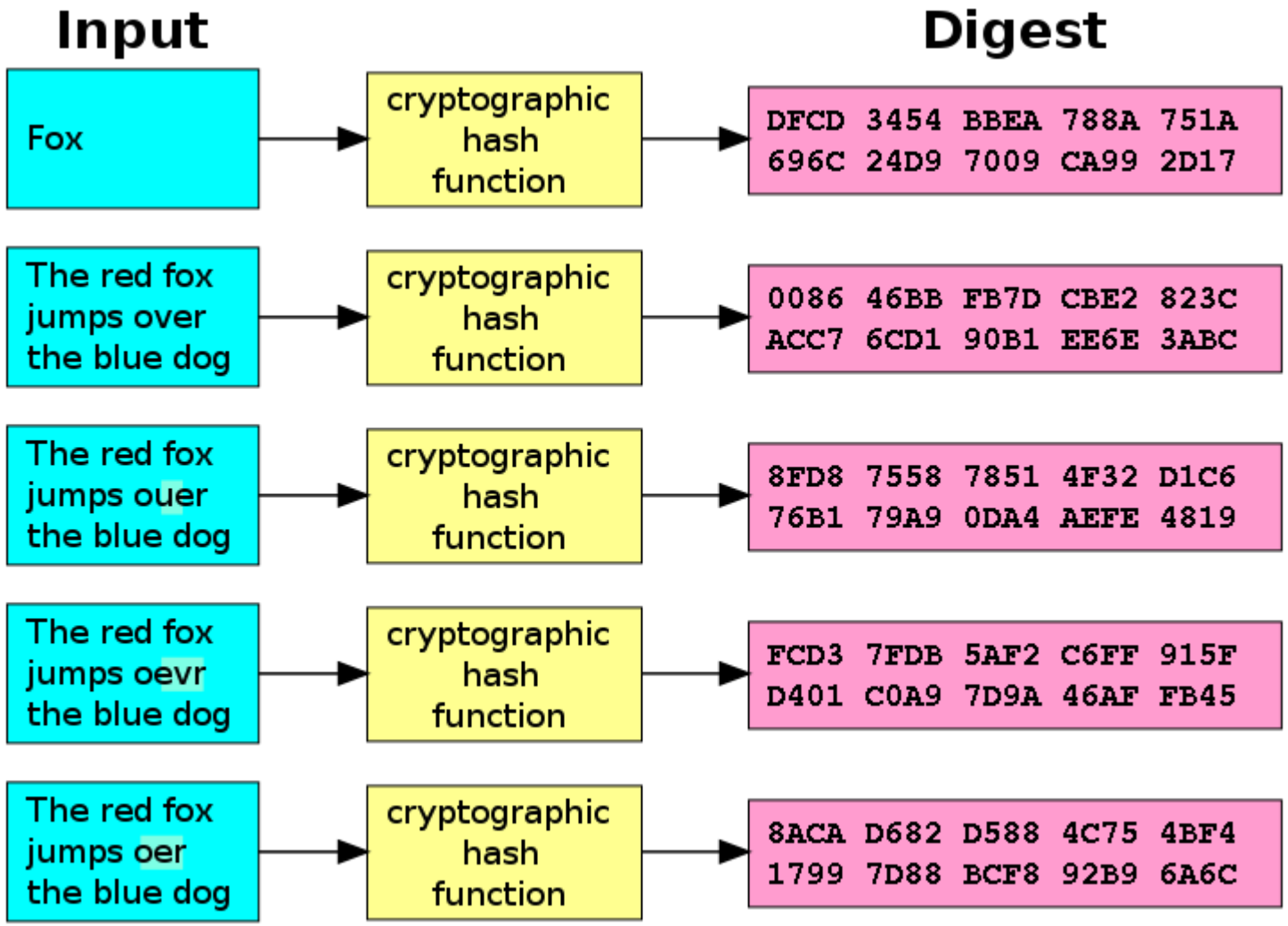
(ii) Misal 33 diubah menjadi 32

```
Pada bulan Oktober 2004 ini, suhu udara kota Bandung terasa
lebih panas dari hari-hari biasanya. Menurut laporan Dinas
Meteorologi Kota Bandung, suhu tertinggi kota Bandung
adalah 32 derajat Celcius pada Hari Rabu, 17 Oktober yang
lalu. Suhu terseut sudah menyamai suhu kota Jakarta pada
hari-hari biasa. Menurut Kepala Dinas Meteorologi,
peningkatan suhu tersebut terjadi karena posisi bumi
sekarang ini lebih dekat ke matahari daripada hari-hari
biasa.

Sebutan Bandung sebagai kota sejuk dan dingin mungkin tidak
lama lagi akan tinggal kenangan. Disamping karena faktor
alam, jumlah penduduk yang padat, polusi dari pabrik di
sekita Bandung, asap knalpot kendaraan, ikut menambah
kenaikan suhu udara kota.
```

Nilai MD5: **2D1436293FAEAF405C27A151C0491267**

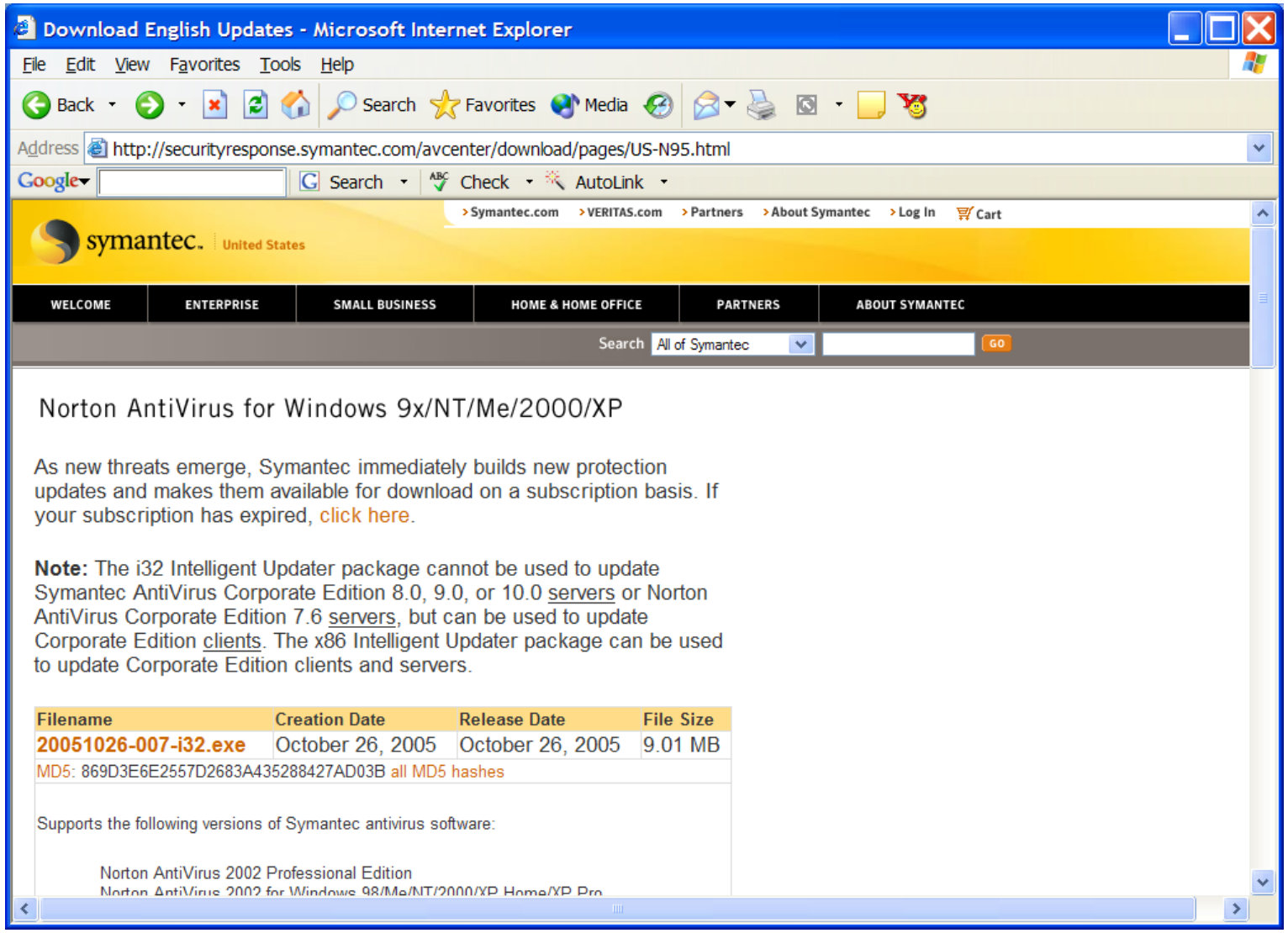
Sebelum diubah : $MD5_1$ = **2F82D0C845121B953D57E4C3C5E91E63**
Sesudah diubah : $MD5_2$ = **2D1436293FAEAF405C27A151C0491267**
Verifikasi: $MD5_1 \neq MD5_2$ (arsip sudah diubah)


Input
Digest
Fox
cryptographic hash function
DFCD 3454 BBEA 788A 751A 696C 24D9 7009 CA99 2D17
The red fox jumps over the blue dog
cryptographic hash function
0086 46BB FB7D CBE2 823C ACC7 6CD1 90B1 EE6E 3ABC
The red fox jumps ouer the blue dog
cryptographic hash function
8FD8 7558 7851 4F32 D1C6 76B1 79A9 0DA4 AEFE 4819
The red fox jumps oevr the blue dog
cryptographic hash function
FCD3 7FDB 5AF2 C6FF 915F D401 C0A9 7D9A 46AF FB45
The red fox jumps oer the blue dog
cryptographic hash function
8ACA D682 D588 4C75 4BF4 1799 7D88 BCF8 92B9 6A6C

- Karena kegunaan untuk mendeteksi perubahan pesan, maka fungsi hash dinamakan juga:

  - *cryptographic checksum*
  - *message integrity check (MIC)*
  - *manipulation detection code (MDC)*

- Program yang di-*downlaod* dari internet sering dilengkapi dengan nilai *hash* untuk menjamin integritas *file.*



2. Menghemat waktu pengiriman.

- Misal untuk memverifikasi sebuah salinan dokumen dengan dokumen aslinya yang berukuran sangat besar (missal 3 MB).
- Salinan dokumen berada di tempat yang jauh dari dokumen asli
- Ketimbang mengirim salinan dokumen tersebut secara keseluruhan ke kantor pusat (yang membutuhkan waktu transmisi lama), lebih sangkil mengirimkan *message digest*-nya.
- Jika *message digest* salinan dokumen sama dengan *message digest* dokumen yang asli, berarti salinan dokumen tersebut sama dengan dokumen asli.

3. Menormalkan panjang data yang beraneka ragam.

- Misalkan *password* panjangnya bebas (minimal 8 karakter)
- *Password* disimpan di komputer *host* (*server*) untuk keperluan otentikasi pemakai komputer.
- *Password* disimpan di dalam basisdata.
- Untuk menyeragamkan panjang *field password* di dalam basisdata, *password* disimpan dalam bentuk nilai *hash* (panjang nilai *hash* tetap).

Kolisi

- Kolisi (*collision*) adalah kondisi dua pesan sembarang memiliki nilai *hash* yang sama.

- Adanya kolisi menunjukkan fungsi *hash* tidak aman secara kriptografis

Fungsi Hash MD5

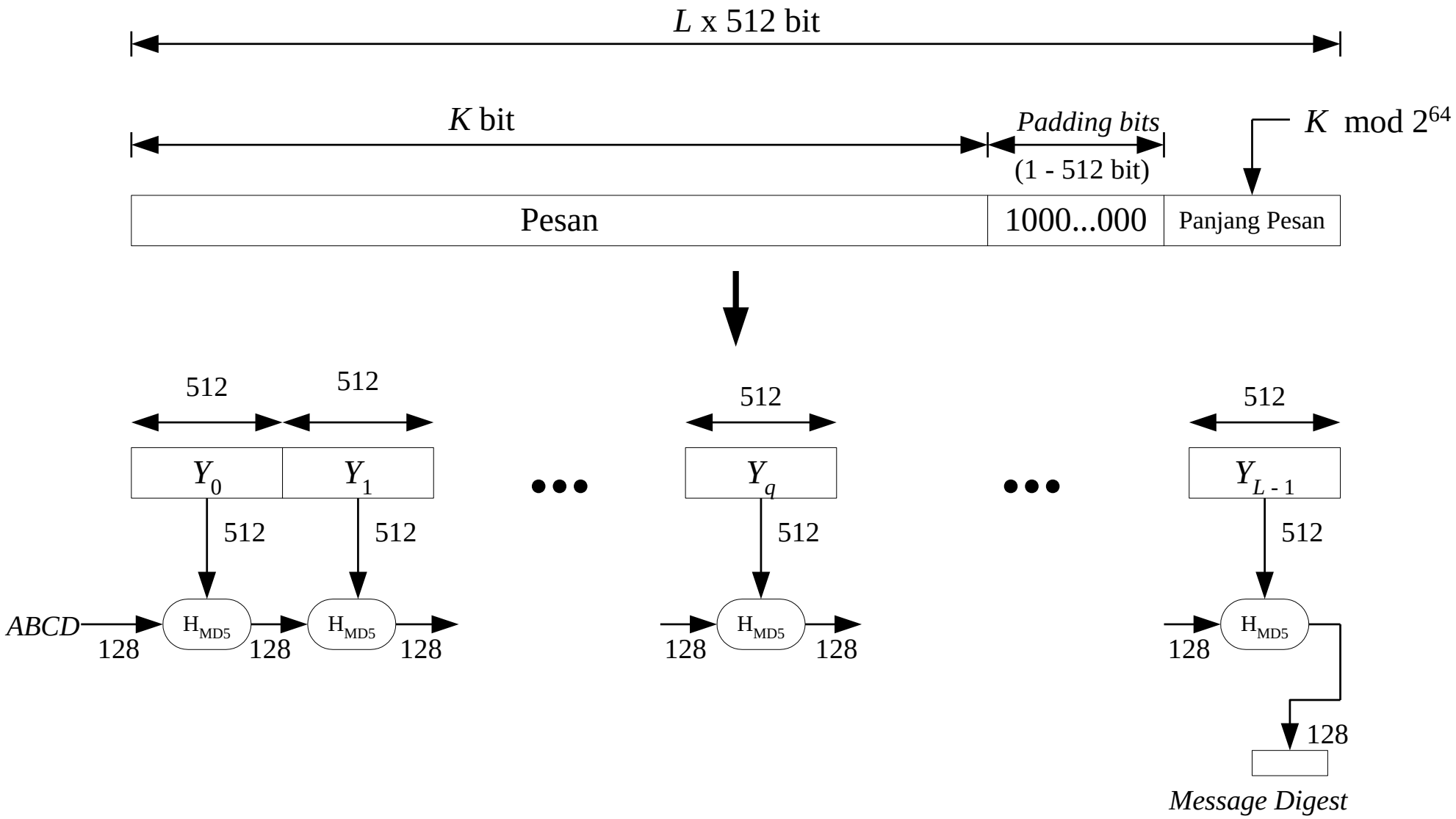
Pendahuluan

- *MD5* adalah fungsi *hash* satu-arah yang dibuat oleh Ronald Rivest.

- *MD5* merupakan perbaikan dari *MD4* setelah *MD4* ditemukan kolisinya.

- Algoritma *MD5* menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan *message digest* yang panjangnya 128 bit (= 32 karakter heksadesimal).

Gambaran umum MD5



- Contoh *message digest* beberapa pesan yang dihasilkan oleh MD5:

md5(“halo”) = 57f84228617109485 5e51fc3a541c1e2

md5(“halo, apa kabar teman?”) =
d29029a1dcf256466290d773f5dbfc0f

md5(“The quick brown fox jumps over the lazy dog”) =
9e107d9d372bb6826bd81d3542a419d6

Contoh. Contoh message digest dari sebuah file, `bandung.txt`, sebagai berikut:

```
Pada bulan Oktober 2004 ini, suhu udara kota Bandung terasa lebih panas dari
hari-hari biasanya. Menurut laporan Dinas Meteorologi Kota Bandung, suhu
tertinggi kota Bandung adalah 33 derajat Celcius pada Hari Rabu, 17 Oktober
yang lalu. Suhu terseut sudah menyamai suhu kota Jakarta pada hari-hari biasa.
Menurut Kepala Dinas Meteorologi, peningkatan suhu tersebut terjadi karena
posisi bumi sekarang ini lebih dekat ke matahari daripada hari-hari biasa.

Sebutan Bandung sebagai kota sejuk dan dingin mungkin tidak lama lagi akan
tinggal kenangan. Disamping karena faktor alam, jumlah penduduk yang padat,
polusi dari pabrik di sekita Bandung, asap knalpot kendaraan, ikut menambah
kenaikan suhu udara kota.
```

Message digest dari arsip `bandung.txt` yang dihasilkan oleh algoritma *MD5* adalah 128-bit:

```
0010 1111 1000 0010 1100 0000 1100 1000 1000 0100 0101 0001 0010 0001
1011 0001 1011 1001 0101 0011 1101 0101 0111 1101 0100 1100 0101 1001
0001 1110 0110 0011
```

atau, dalam notasi HEX adalah:

```
2F82D0C845121B953D57E4C3C5E91E63
```

Kriptanalisis MD5

Review kembali sifat-sifat fungsi *hash H*:

a) collision resistance : sangat sukar menemukan dua input *a* dan *b* sedemikian sehingga $H(a) = H(b)$

b) preimage resistance: untuk sembarang output *y*, sukar menemukan input *a* sedemikian sehingga $H(a) = y$

c) second preimage resistance – untuk input *a* dan output $y = H(a)$, sukar menemukan input kedua *b* sedemikian sehingga $H(b) = y$

Kriptanalisis MD5

- Secara teori, menemukan kolisi pada fungsi *hash* sangatlah sukar dilakukan.

- Pada awalnya penemu algoritma *MD5* menganggap usaha tersebut hampir tidak mungkin dilakukan karena membutuhkan waktu yang sangat lama.

- Tetapi, pada tahun 1996, Dobbertin melaporkan penemuan kolisi pada algoritma *MD5* meskipun kecacatan ini bukan kelemahan yang fatal.

- Pada tanggal 1 Maret 2005, Arjen Lenstra, Xiaoyun Wang, dan Benne de Weger mendemostraskan pembentukan dua buah pesan berbeda (berupa sertifikat X.509, yang akan dijelaskan di dalam bab *Infrastruktur Kunci Publik)* tetapi mempunyai nilai *hash* yang sama (lihat publikasinya di dalam hhttp://eprint.iacr.org/2005/067).

- Beberapa hari kemudian, Vlastimil Klima (http://eprint.iacr.org/2005/075) memperbaiki algoritma Lenstra dkk yang dapat menghasilkan kolisi *MD5* hanya dalam waktu beberapa jam dengan menggunakan komputer *PC.*

(Sumber: Wikipedia)

Contoh dua pesan yang hanya berbeda 6 bit tetapi memiliki nilai *hash* sama (Sumber: *Eric Rescorla (2004-08-17). "A real MD5 collision"*):

M1 =

d131dd02c5e6eec4693d9a0698aff95c2fcab5**8**712467eab400458
3eb8fb7f8955ad340609f4b30283e4888325**7**1415a085125e8f7cd
c99fd91dbd**f**280373c5bd8823e3156348f5bae6dacd436c919c6dd
53e2**b**487da03fd02396306d248cda0e99f33420f577ee8ce54b670
80**a**80d1ec69821bcb6a8839396f965**2**b6ff72a70

M2 =

d131dd02c5e6eec4693d9a0698aff95c2fcab5**0**712467eab400458
3eb8fb7f8955ad340609f4b30283e4888325**f**1415a085125e8f7cd
c99fd91dbd**7**280373c5bd8823e3156348f5bae6dacd436c919c6dd
53e2**3**487da03fd02396306d248cda0e99f33420f577ee8ce54b670
80**2**80d1ec69821bcb6a8839396f965**a**b6ff72a70

Kedua pesan di atas memiliki nilai *hash* **79054025255fb1a26e4bc422aef54eb4.**

Dua buah gambar biner yang memiliki nilai hash MD5 yang sama

Sumber:
https://natmchugh.blogspot.com/2015/05/how-to-make-two-binaries-with-same-md5.html

Secure Hash Algorithm (SHA)

Secure Hash Algorithm (*SHA*)

- *SHA* adalah fungsi *hash* satu-arah yang dibuat oleh *NIST* dan digunakan bersama *DSS* (*Digital Signature Standard*).

- Oleh *NSA*, *SHA* dinyatakan sebagai standard fungsi *hash* satu-arah.

- *SHA* didasarkan pada *MD4* yang dibuat oleh Ronald L. Rivest dari *MIT*.

- Algoritma *SHA* menerima masukan berukuran maksimum 2^{64} bit (2.147.483.648 *gigabyte*) dan menghasilkan *message digest* yang panjangnya 160 bit.

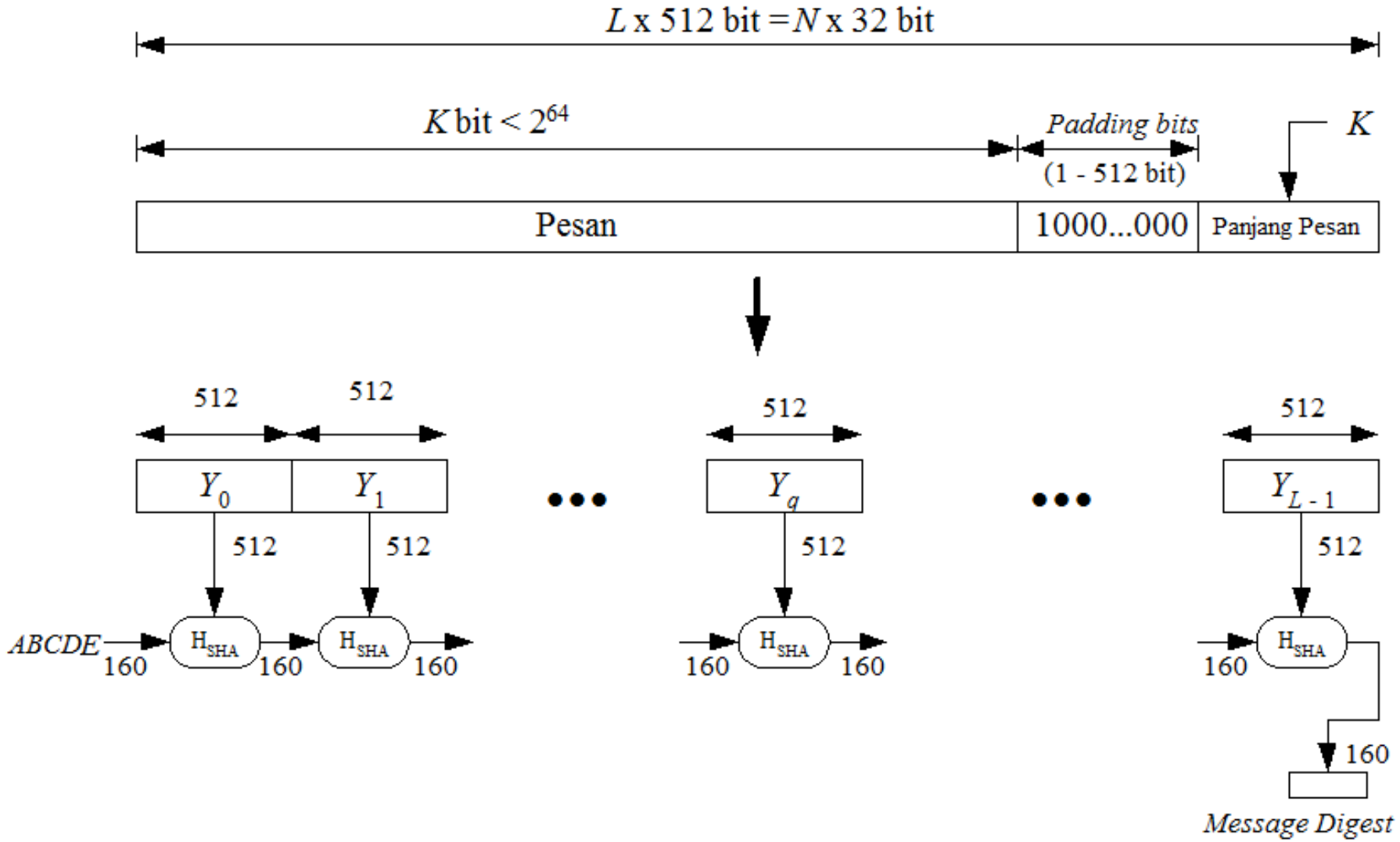
- *Message digest* dari SHA lebih panjang dari *message digest* yang dihasilkan oleh *MD5* (128 bit).

- Sebutan *SHA* mengacu pada keluarga fungsi *hash* satu-arah SHA.
- Enam varian *SHA*:
    - SHA-0
    - SHA-1
    - SHA-224
    - SHA-256
    - SHA-384
    - SHA-512

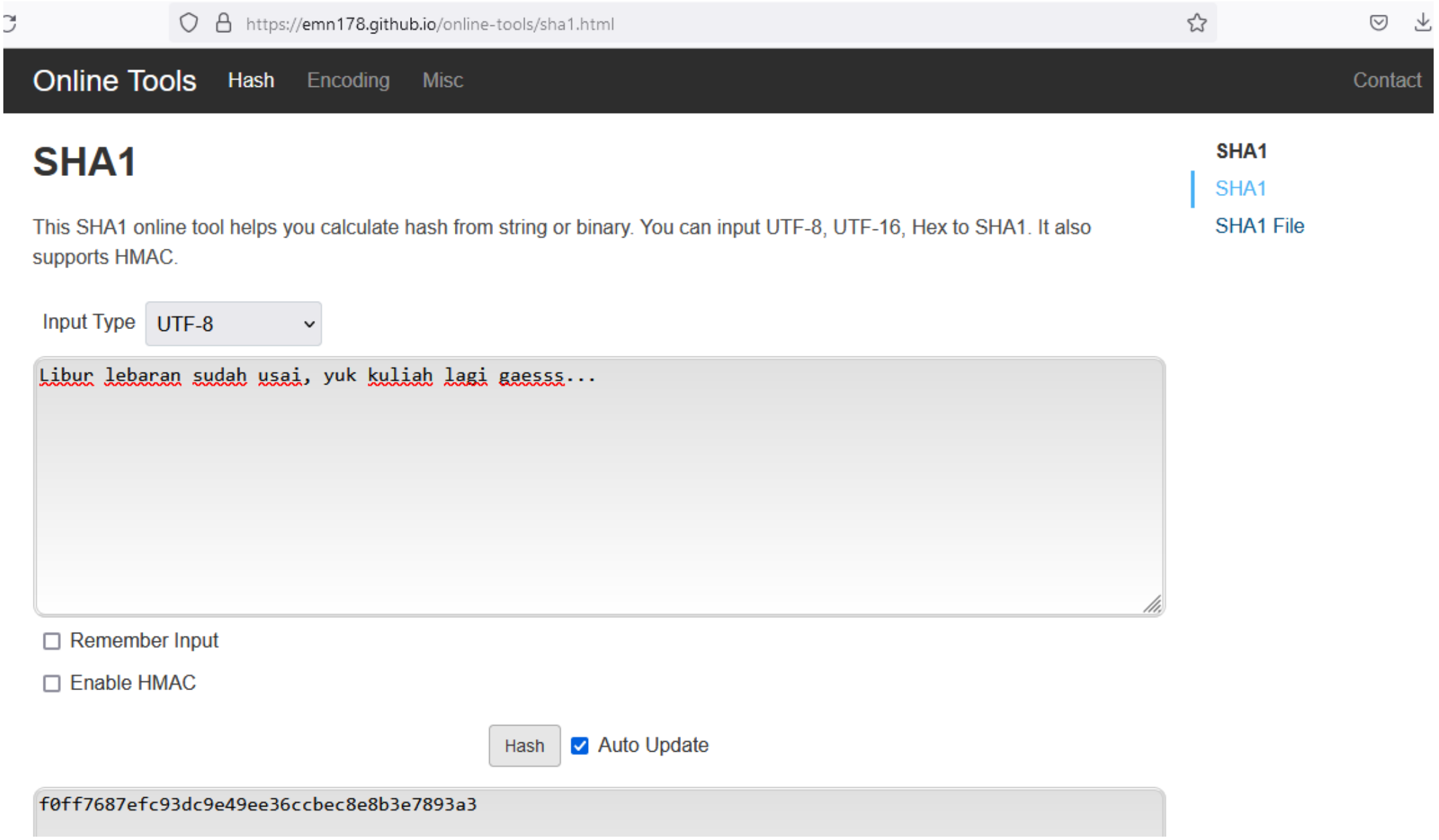
- SHA-0 sering diacu sebagai *SHA* saja
- Yang akan dibahas: SHA-1
- Detil algoritma SHA-1 dapat dibaca di dalam dokumen RFC 3174 (http://www.faqs.org/rfcs/rfc3174.html)

		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Word size (bits)	Rounds	Operations	Collisions found?
SHA-0									Yes
SHA-1		160	160	512	$2^{64} - 1$	32	80	+,and,or, xor,rot	Yes
SHA-2	*SHA-256/224*	256/224	256	512	$2^{64} - 1$	32	64	+,and,or, xor,shr,rot	No
	SHA-512/384	512/384	512	1024	$2^{128} - 1$	64	80		

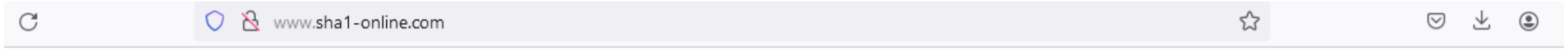
Gambaran Umum *SHA-1*



Demo SHA1 online: https://emn178.github.io/online-tools/sha1.html



Demo SHA1 online lainnya: http://www.sha1-online.com/



Home Page | SHA1 in JAVA | Secure password generator | Linux | Privacy Policy

SHA1 and other hash functions online generator

Balik kampung hari raya ini kah? hash

gost

Result for
gost: 9787c5e68b989d10a418e565192a4fcae7f4f1e5e7aa73126a8e4987aa2e09c4

SHA-1 MD5 on Wikipedia

We love SPAIN and oldpics.org

Contoh *message digest* beberapa pesan yang dihasilkan oleh SHA-1:

sha1("halo")= 33b1eac210971fb02a3b90afce9dbff758be794d

sha1("halo, apa kabar teman? ") =
03d17abd2962dbed1037d1230f012037cd25fe91

sha1("The quick brown fox jumps over the lazy dog") =
2fd4e1c67a2d28fced849ee1bb76e7391b93eb12

Kriptanalisis *SHA*-1

- Pada tahun 2005, Rijmen dan Oswald mempubliksikan serangan pada versi *SHA*-1 yang direduksi (hanya menggunakan 53 putaran dari 80 putaran) dan menemukan kolisi dengan kompleksitas sekitar 2^{80} operasi (lihat di http://eprint.iacr.org/2005/010).

- Pada bulan Februari 2005, Xiayoun Wang, Yiqun Lisa Yin, dan Hongbo Yo mempublikasikan serangan yang dapat menemukan kolisi pada versi penuh SHA-1, yang membutuhkan sekitar 2^{69} operasi (lihat beritanya di:

  http://www.schneier.com/blog/archives/2005/02/sha_1broken.html)

- Rizki Wicaksono, seorang *hacker* alumni Informatika ITB Angkatan 1999 mendemonstrasikan cara membentuk dua file berbeda yang memiliki nilai *hash* SHA-1 sama (silakan baca tulisannya pada pranala:

  http://www.ilmuhacking.com/cryptography/membuat-sha1-collision-file/

Rizki Wicaksono

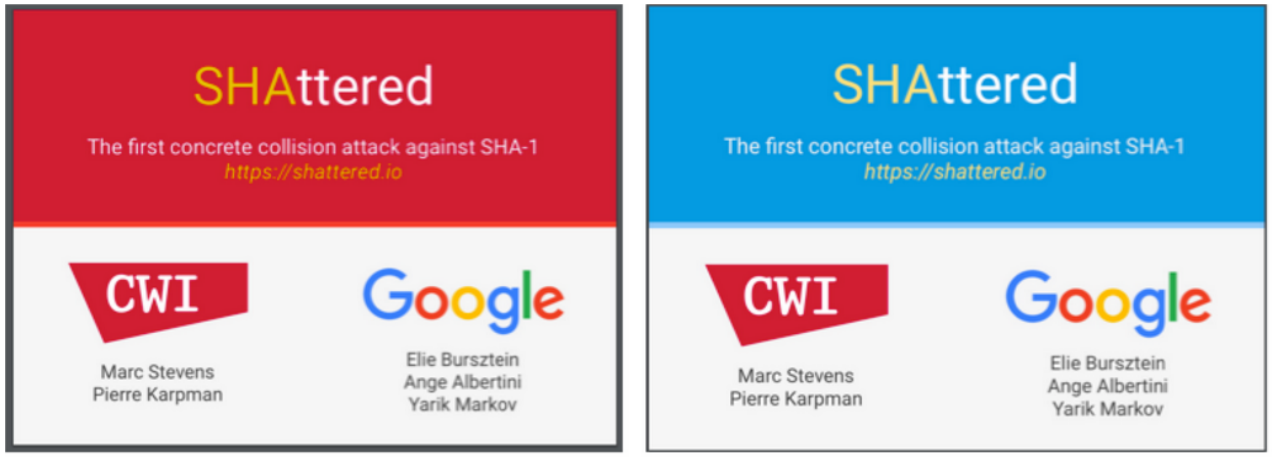
Published

March 21, 2017

in Cryptography

Membuat file dengan SHA1 collision

Sejumlah ilmuwan dengan didukung kekuatan komputasi Google berhasil melakukan serangan SHA1 collision terhadap dokumen PDF. Mereka menciptakan dua file PDF dengan nilai SHA1 hash yang identik walaupun isi filenya berbeda.



shattered-1.pdf dan shattered-2.pdf memiliki SHA1 yang sama

Latihan 1

1. Buka situs https://codebeautify.org/sha256-hash-generator untuk menghitung message digest dengan SHA-1 (atau SHA lainnya).
2. Ketikkan sembarang pesan, lalu hitung nilai *message digest*-nya.
3. Kirim pesan dan *message digest* ke temanmu via Line atau email
4. Temanmu membaca pesan tersebut, *copy*, dan *paste* ke situs tersebut, lalu hitung nilai *message digest*-nya.
5. Temanmu membandingkan message digest yang dia terima dengan message digest yang dia hitung. Jika sama, berarti pesan masih asli (tidak mengalami perubahan)

Latihan 2

1. Buka situs https://codebeautify.org/sha256-hash-generator untuk menghitung message digest dengan MD5.
2. Unggah sebuah file gambar, lalu hitung nilai *message digest*-nya.
3. Edit gambar tersebut dengan *photoshop* atau *paint* atau *software* lainnya, lalu *save*.
4. Hitung nilai hash file gambar hasil editan, lalu bandingkan *message digest* file gambar yang baru dengan message digest gambar yang lama.

MAC
(*Message Authentication Code*)

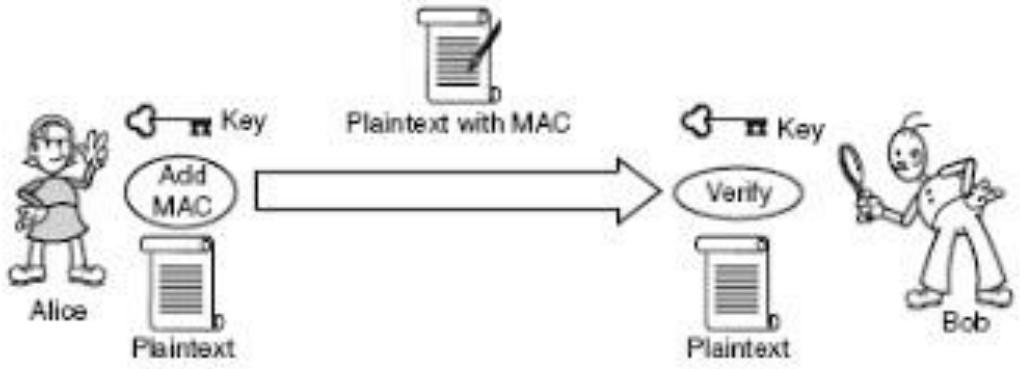


Figure 9.3 Message Authentication Code (MAC)

Definisi

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

$$MAC = C_K(M)$$

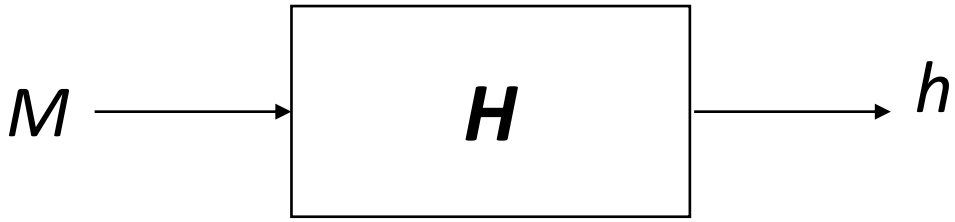
MAC = kode otentikasi pesan

C = algoritma *MAC*

K = kunci rahasia

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

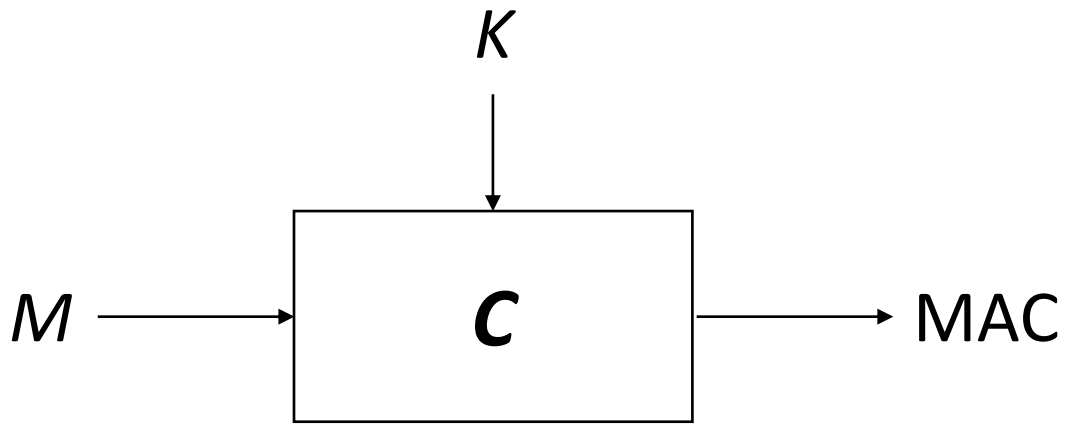
Perbedaan Algoritma MAC dengan Fungsi Hash biasa



$h = H(M)$

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 masih asli.

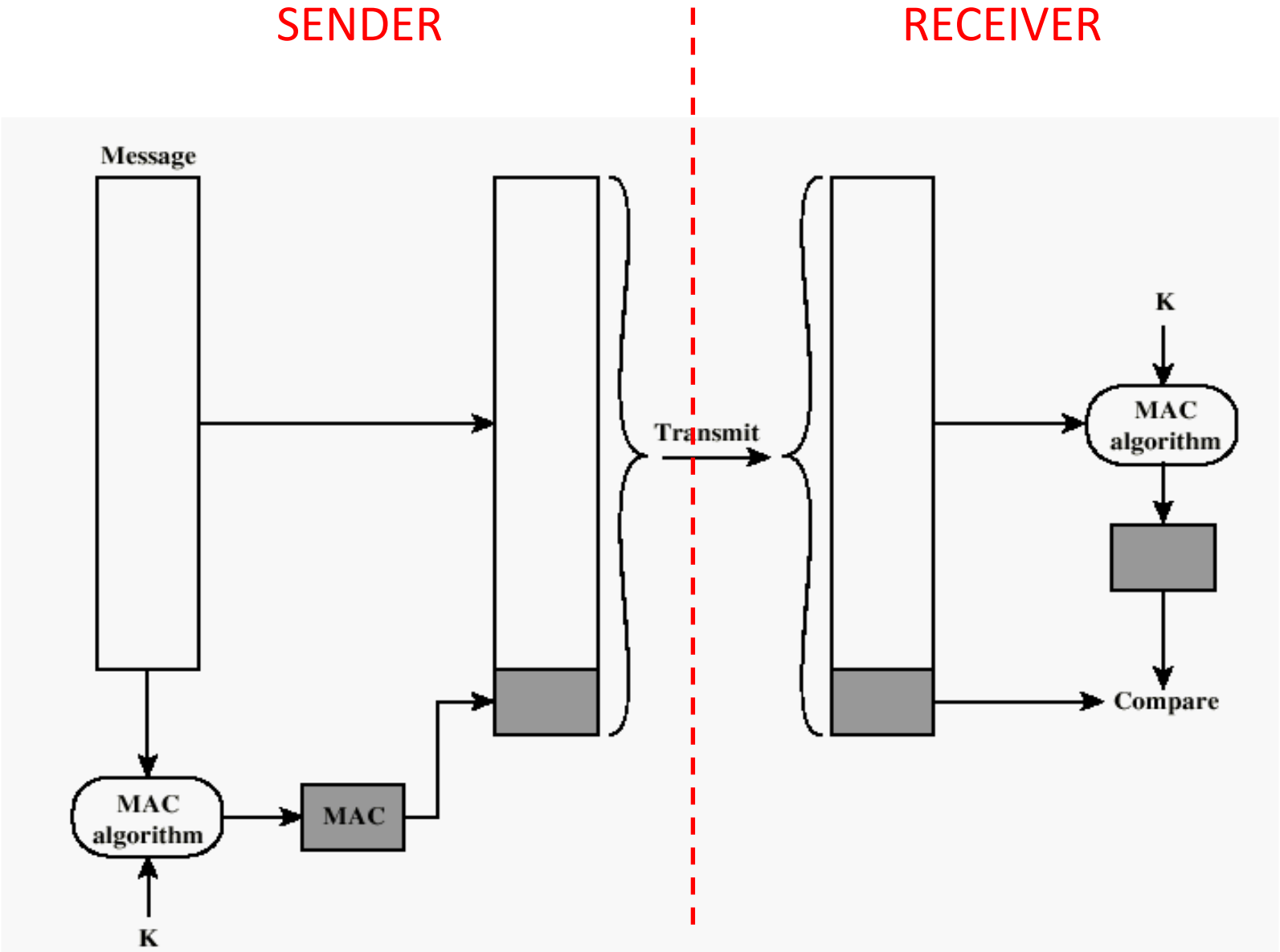
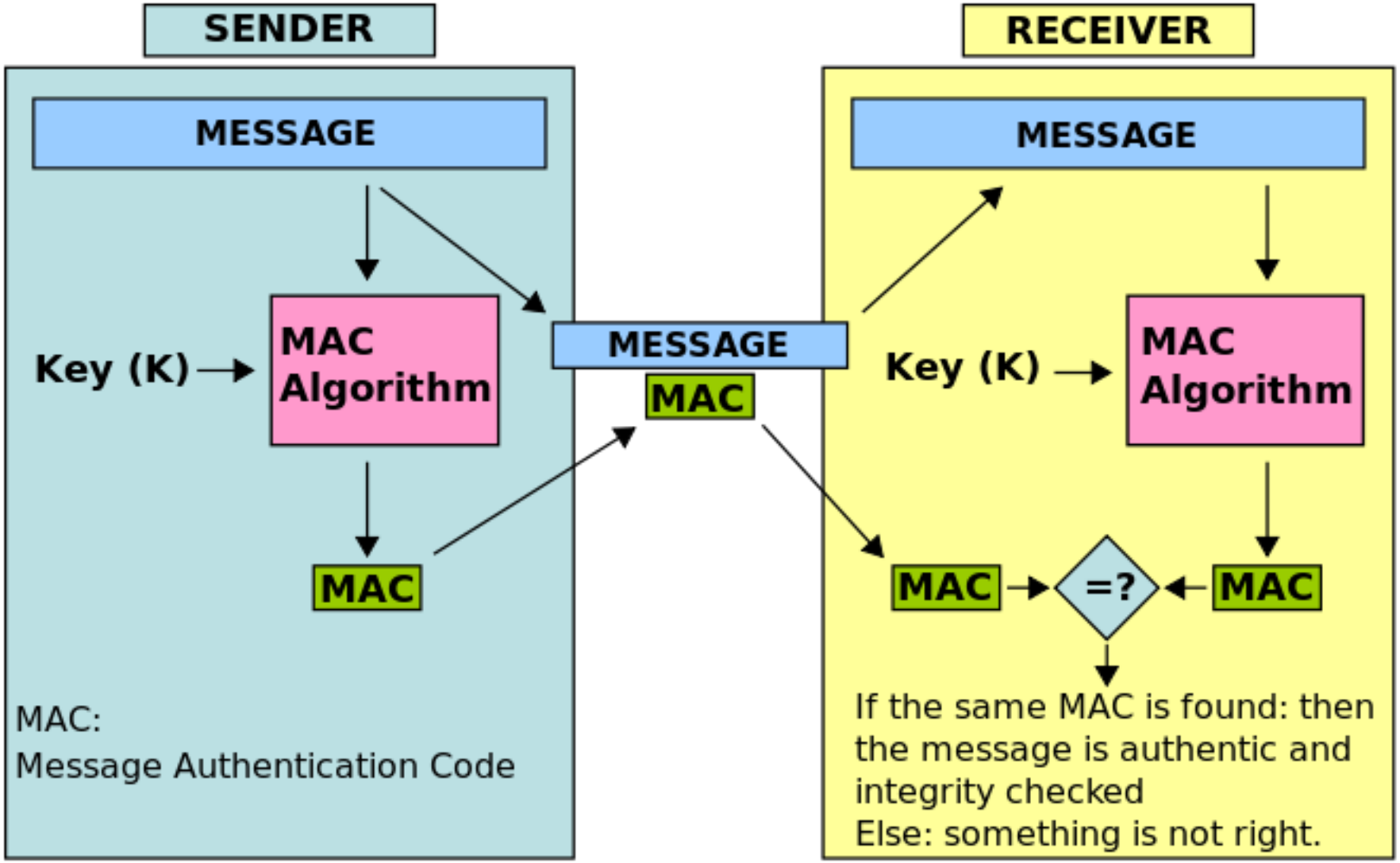


Figure 3.1 Message Authentication Using a Message Authentication Code (MAC)


SENDER
RECEIVER
MESSAGE
MESSAGE
Key (K)
MAC Algorithm
MESSAGE
MAC
Key (K)
MAC Algorithm
MAC
MAC
=?
MAC
MAC:
Message Authentication Code
If the same MAC is found: then the message is authentic and integrity checked
Else: something is not right.

Kegunaan MAC

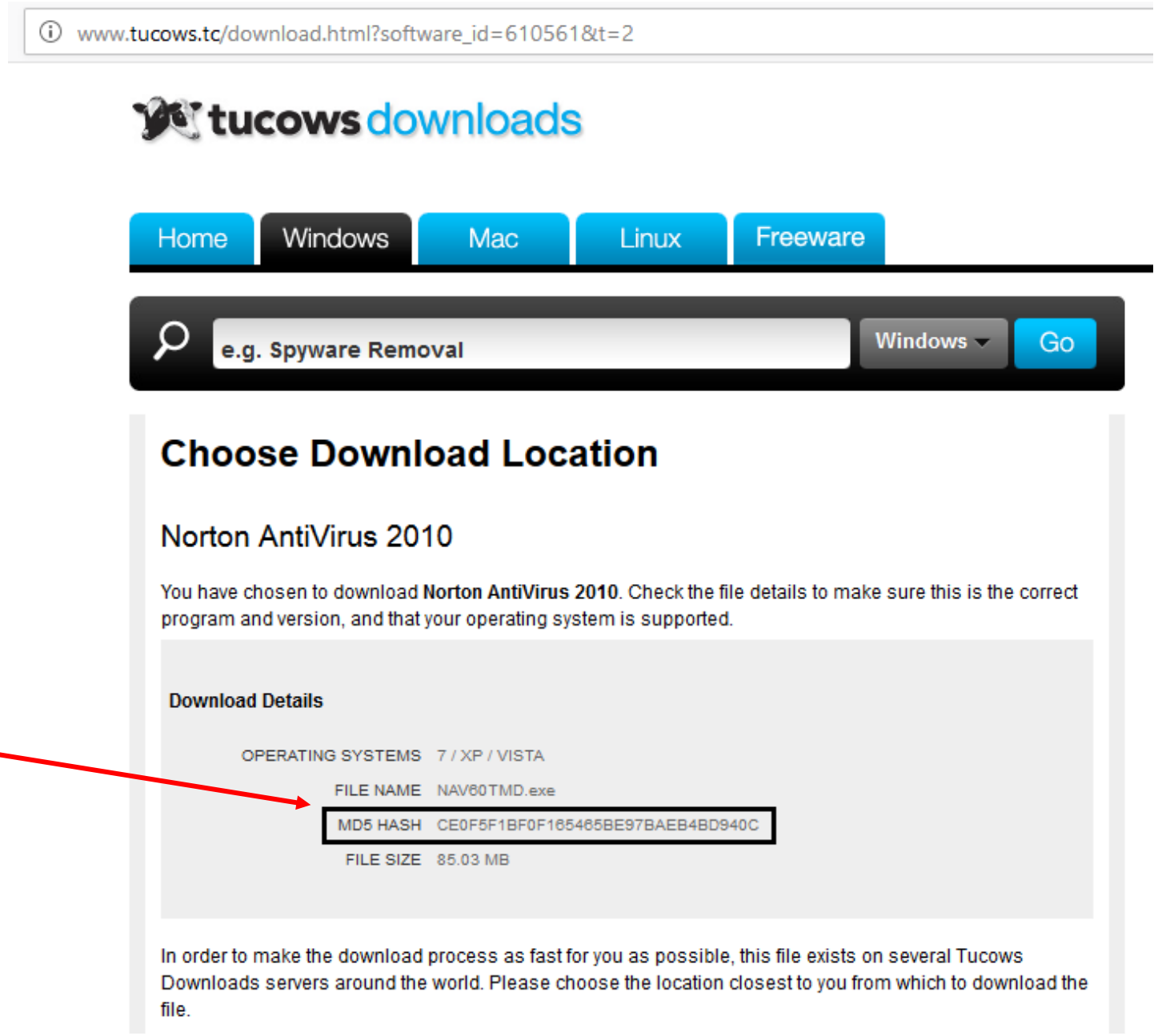
1. Seperti *message digest*, MAC digunakan untuk memeriksa keaslian pesan, dokumen, dsb. (*cryptographic checksum*)

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

- Jika MAC yang dihitung oleh penerima pesan = MAC yang melekat pada pesan, berarti pesan masih asli.
- Jika pemilik pesan menggunakan fungsi *hash* satu-arah biasa (seperti MD5 atau SHA), maka pihak lawan dapat menghitung *message digest* yang baru dari dokumen yang sudah diubah, lalu menggantinya.

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

Hacker bisa mengganti file dengan file lain, mengganti nilai MD5 semula dengan nilai MD5 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 sesunguhnya (asli).

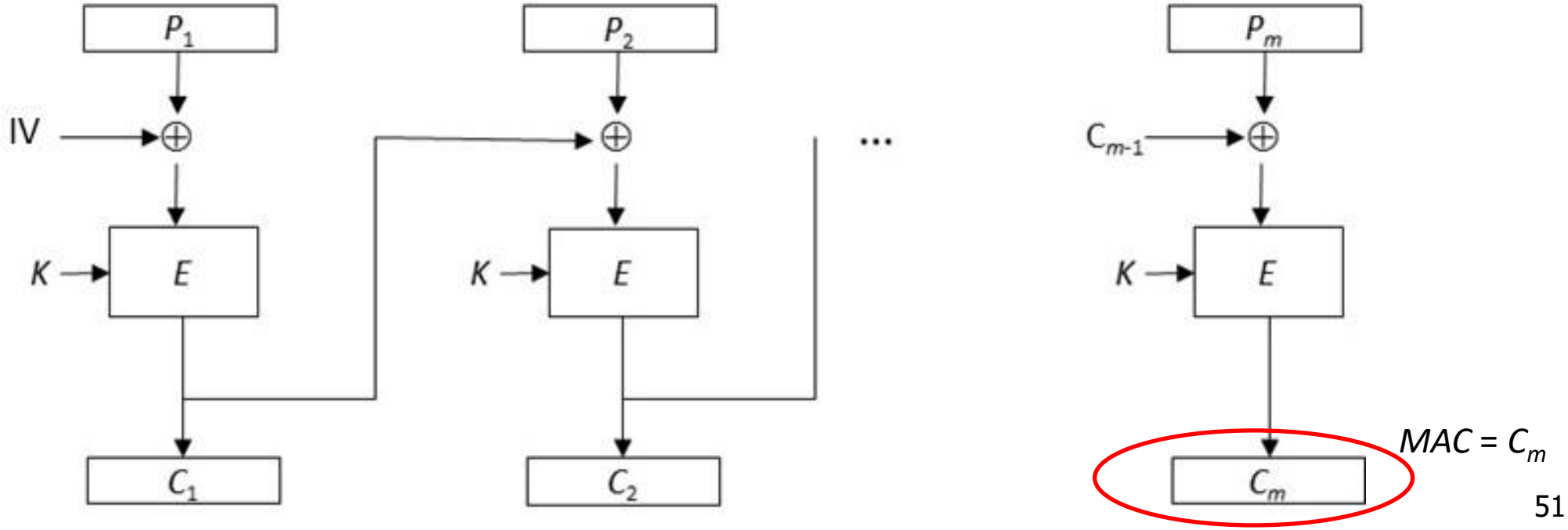
- Hal ini karena hanya pengirim dan penerima pesan yang mengetahui kunci K.

- 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.

Algoritma MAC

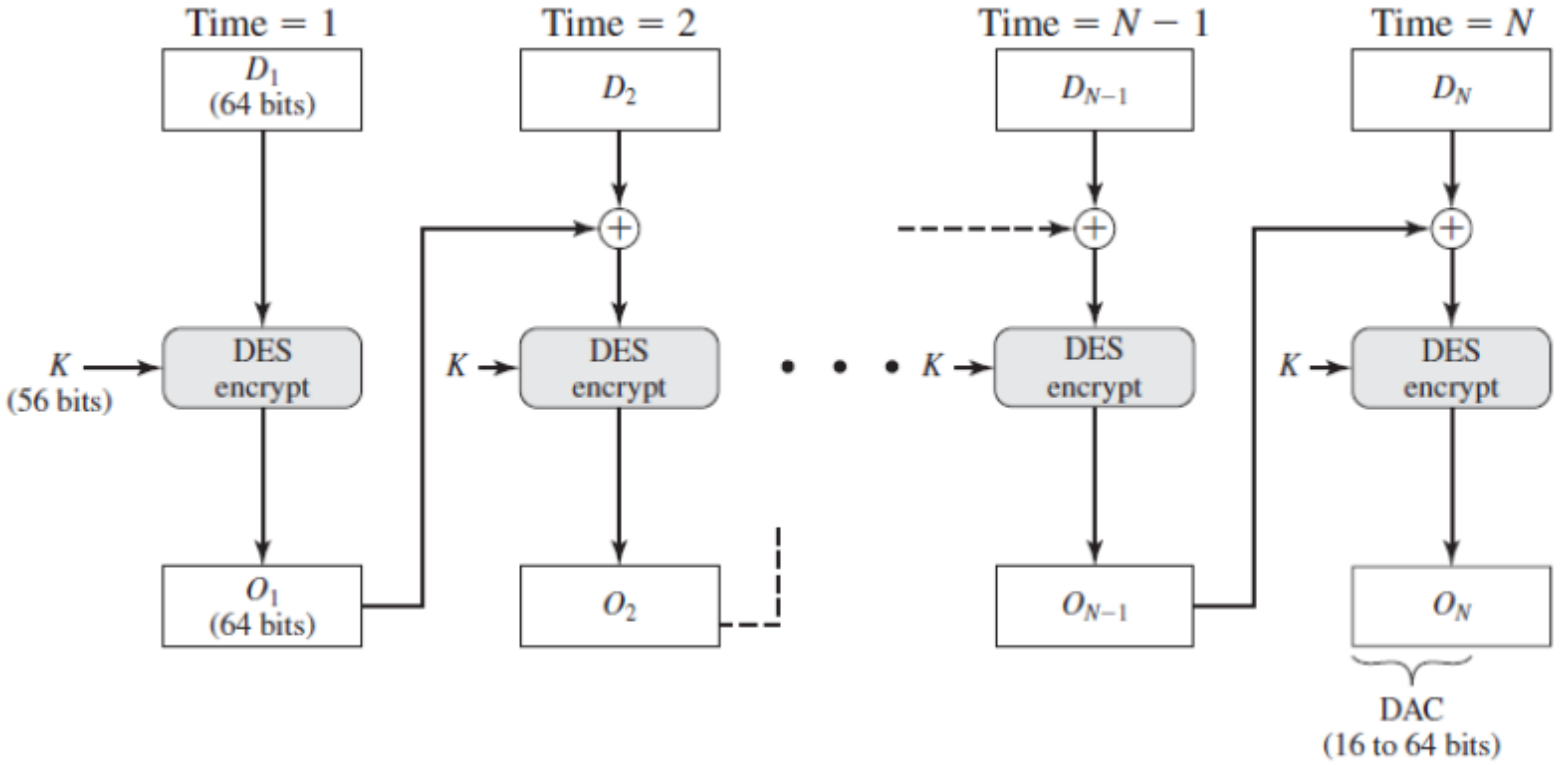
(a) Algoritma MAC berbasis *block cipher*

- *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:



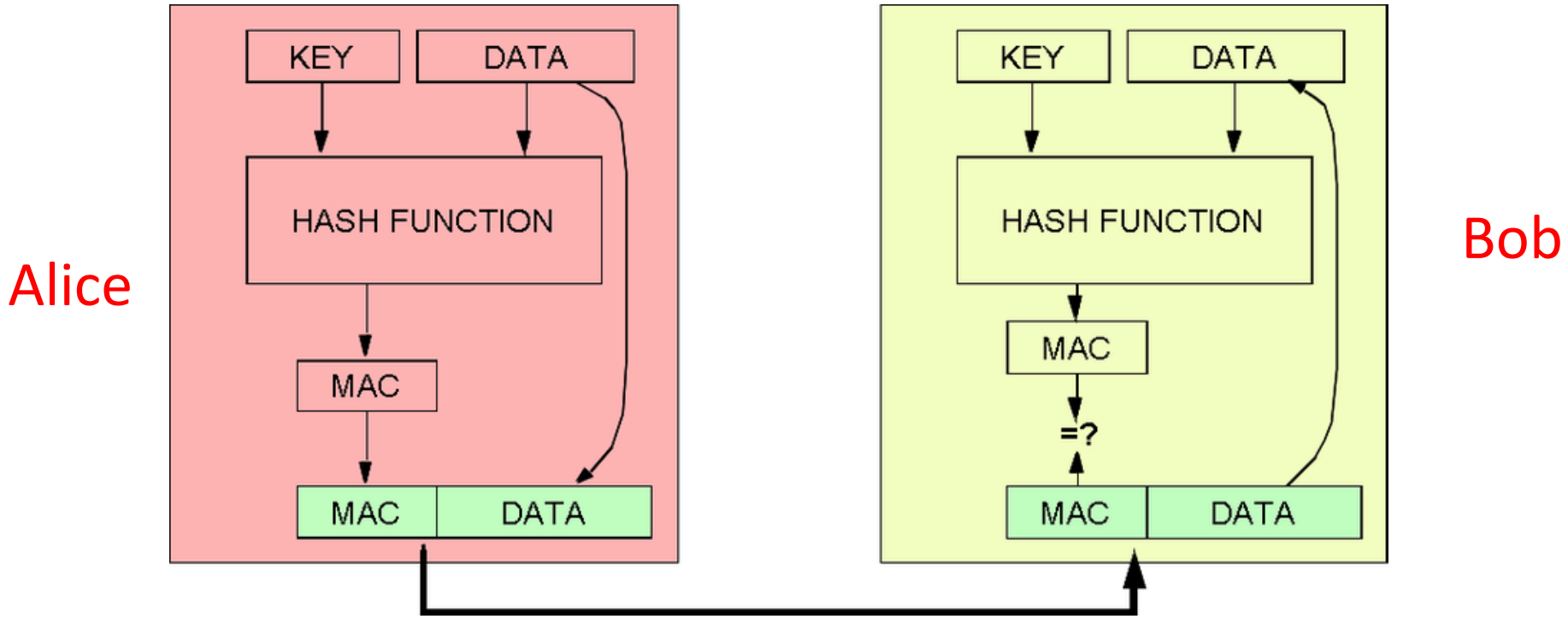
(b) Algoritma MAC berbasis fungsi *hash* satu-arah (HMAC)

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

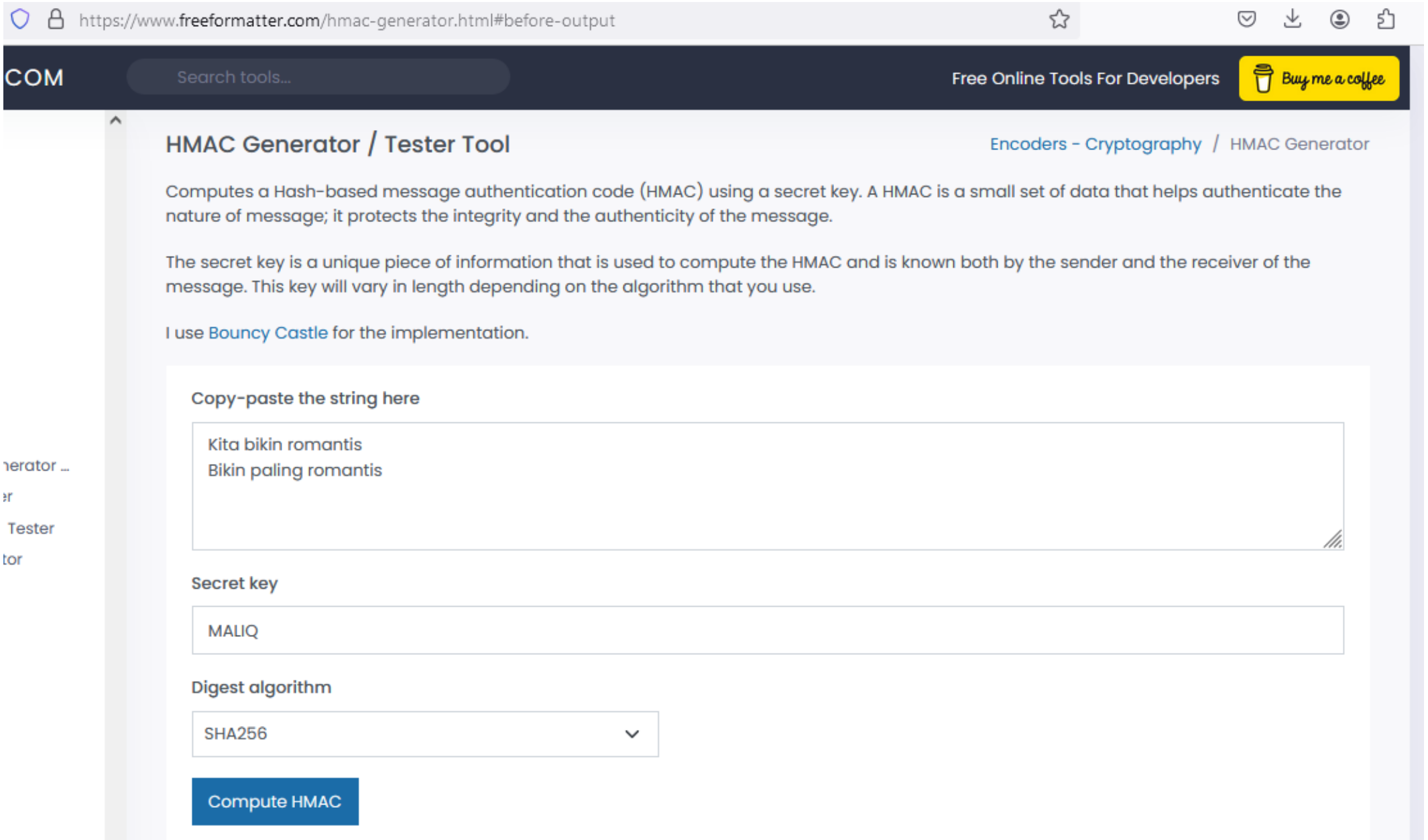
MAC = H(M|K) ket: '|' adalah simbol *concatenation*

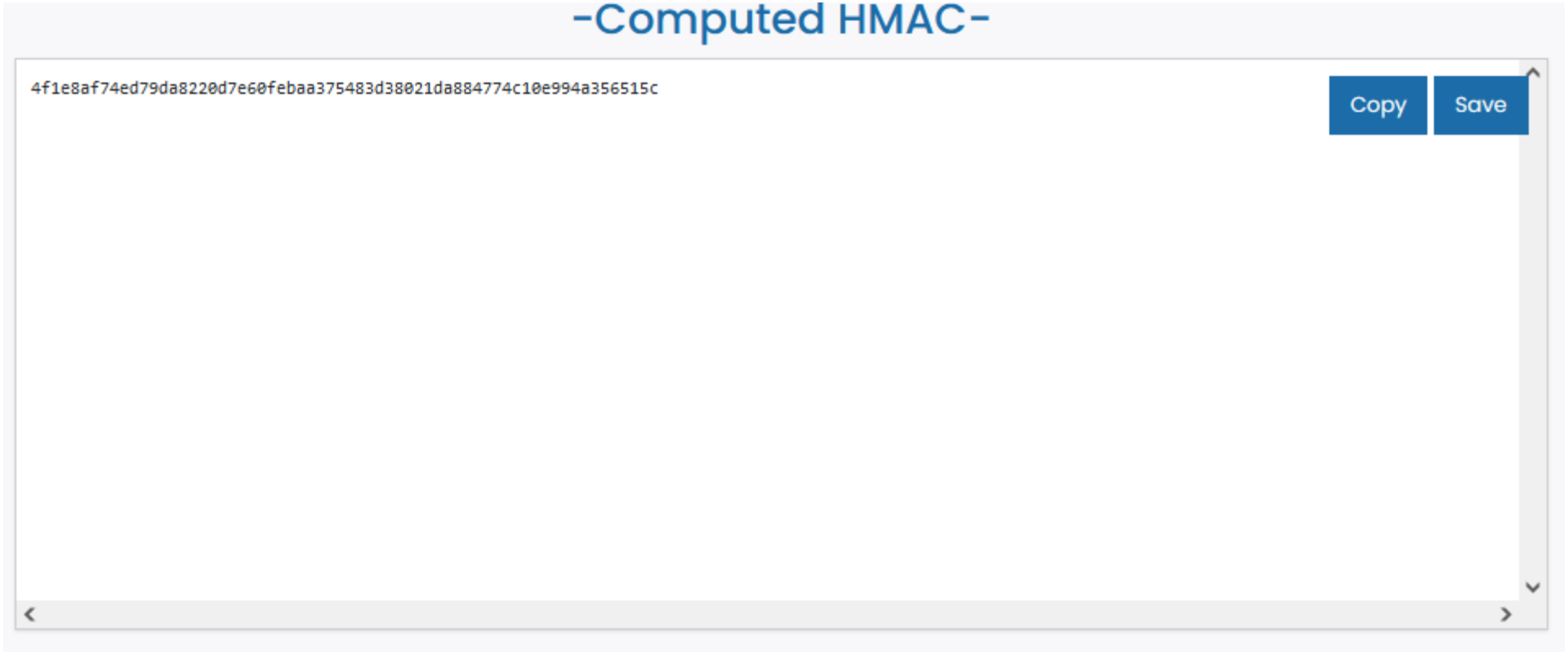
- Panjang MAC tergantung dari fungsi hash yang digunakan. Jika menggunakan fungsi SHA-1, maka MAC yang dihasilkan adalah 160 bit

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



https://www.freeformatter.com/hmac-generator.html




-Computed HMAC-
4f1e8af74ed79da8220d7e60febaa375483d38021da884774c10e994a356515c
Copy
Save

Contoh:

M = *Halo, Bob!*

K = 12345678

Fungsi Hash: SHA-1

MAC = 6f8605c7c3a649a40abfb87b44aa21f356e931a0

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

Latihan 3

1. Buka situs https://www.freeformatter.com/hmac-generator.html untuk menghitung MAC dengan HMAC.
2. Sepakati sebuah kunci K dengan temanmu dengan cara bertemu langsung (jangan pakai *Line* atau *email*)
3. Pilih fungsi hash SHA256 pada situs tersebut, ketikkan sebuah pesan pada situs tersebut, ketikkan kunci K, hitung MAC nya
4. Kirim pesan dan MAC kepada temanmu via *Line* atau email
5. Temanmu mengambil pesanmu dari Line, menghitung MAC dengan kunci K yang sama, lalu bandingkan MAC yang dikirim sebelumnya dengan MAC yang dihitung.

SELAMAT BELAJAR