
Fungsi Hash &
Algoritma SHA-256
Kelompok 5 – Sistem Informasi 2A

Kelompok 5

- Muh. Irfan Aziz (14.1.03.03.0035)
- Rifa'i Dwi Cahyono (14.1.03.03.0143)
- Prista Avinda D. (14.1.03.03.0017)
- Dwi Bagus Kurniawan (14.1.03.03.0019)
- Aditya Gusti Tammam (14.1.03.03.0052)
- Hamim Arifunas (14.1.03.03.0092)

Content

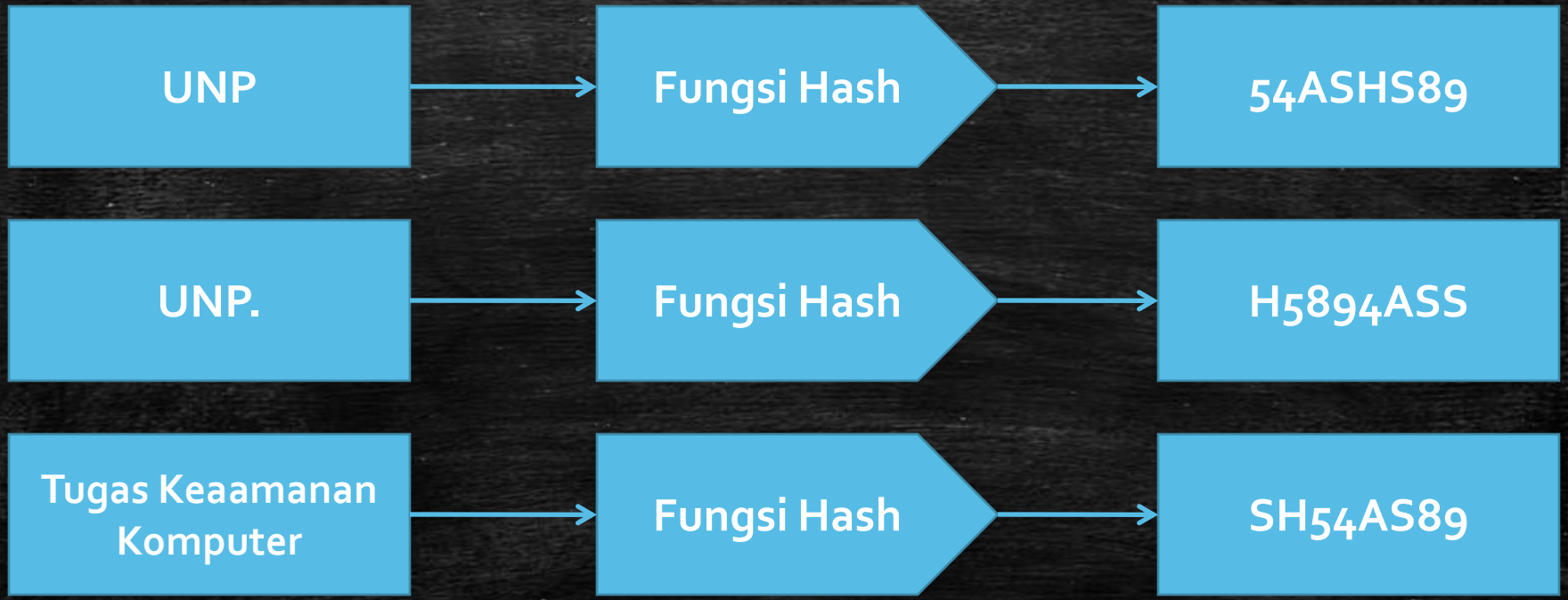
- Pengertian Fungsi Hash
- Sifat-sifat Fungsi Hash
- Manfaat Fungsi Hash
- Message Integrity
- Message Fingerprint
- Macam-macam Fungsi Hash
- Sejarah SHA-256
- Awal Perkembangan SHA-256
- Dasar Prinsip SHA-256
- Cara Kerja SHA-256
- Tahapan SHA-256
- Penerapan SHA-256
- Kelebihan & Kekurangan SHA-256

Apa itu Fungsi Hash ?

- Suatu fungsi yang merubah suatu pesan dengan panjang sembarang menjadi suatu pesan ringkas yang panjangnya selalu tetap meskipun panjang pesan aslinya berbeda-beda.

$$h = H(M)$$

Contoh

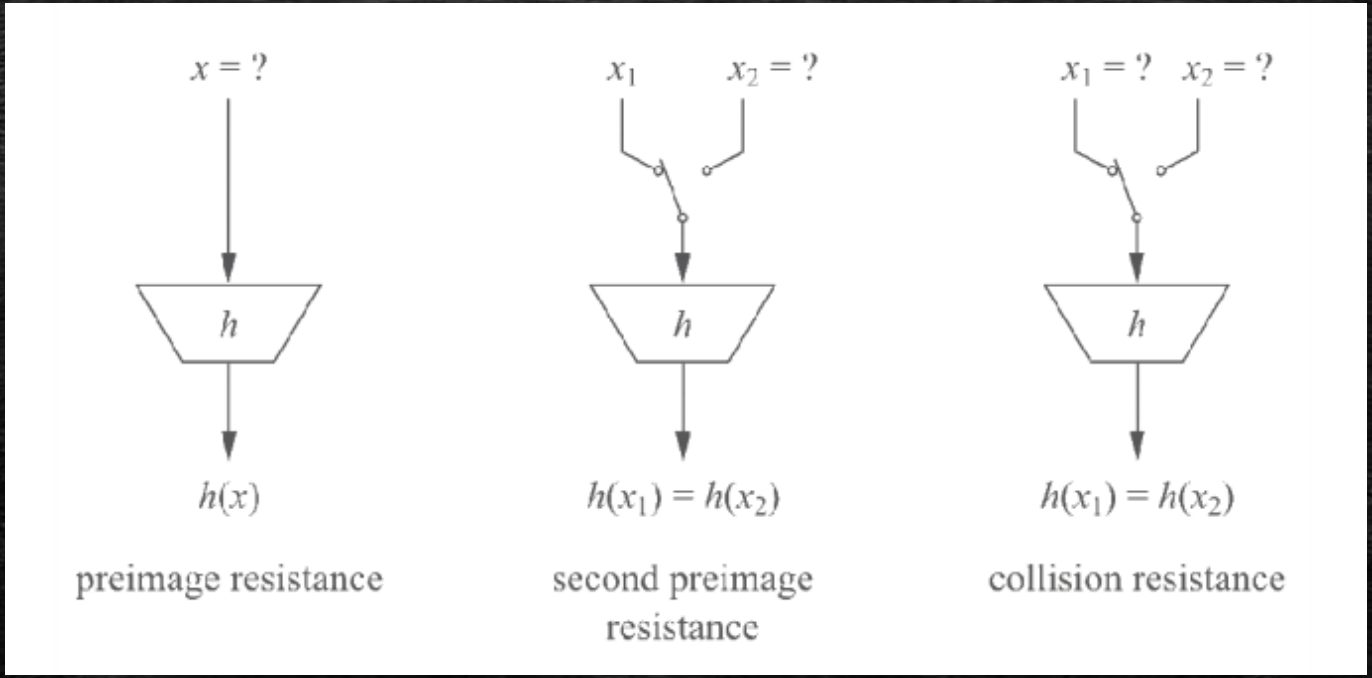


Catatan : Nilai hash diatas hanya contoh dan bukan nilai yang sebenarnya.

Sifat Fungsi Hash

- Fungsi H dapat diterapkan pada blok data berukuran berapa saja.
- H menghasilkan (*h*) dengan panjang tetap.
- *H(x)* mudah dihitung untuk setiap nilai x yang diberikan.

Sifat Fungsi Hash (lanjutan)



Sifat Fungsi Hash (lanjutan)

- **Preimage Resistance** : Untuk setiap *h* yang dihasilkan, tidak mungkin dikembalikan ke nilai *x* sedemikian hingga $H(x) = h$.
- **Second Preimage Resistance** : Untuk setiap nilai x^1, tidak mungkin mencari nilai x^2 sedemikian hingga $H(x^1) = H(x^2)$.
- **Collision Resistance** : Tidak mungkin mencari pasangan x^1 dan x^2 sedemikian hingga $H(x^1) = H(x^2)$.

Manfaat Fungsi Hash

- Autentifikasi password
- Autentifikasi keaslian file
- Tanda tangan digital
- Dsb.

Message Integrity

- Berkaitan dengan keutuhan pesan.
- Memastikan bahwa suatu pesan tidak rusak atau berubah.

Message Fingerprint

- Sidik jari digital dari suatu pesan yang terdiri dari kode, dihitung atas dasar isi pesan, yang dapat dimanfaatkan untuk memeriksa dan menjamin keaslian file.

Macam-macam Fungsi Hash

- MD4
- MD5
- SHA-0
- SHA-1
- **SHA-256**
- SHA-512

Macam-macam Fungsi Hash (bonus)

Name	Author(s)	Year	Block Size	Digest Size
AR	ISO [151]	1992		
Boognish	Daemen[58]	1992	32	up to 160
Cellhash	Daemen, Govaerts, Vandewalle [59]	1991	32	up to 256
FFT-Hash I	Schnorr [318]	1991	128	128
GOST R 34.11-94	Government Committee of Russia for Standards [257]	1990	256	256
FFT-Hash II	Schnorr [319]	1992	128	128
HAVAL	Zheng, Pieprzyk, Seberry [402]	1994	1024	128, 160, 192, 224, 256
MAA	ISO [150]	1988	32	32
MD2	Rivest [162]	1989	512	128
MD4	Rivest [288]	1990	512	128
MD5	Rivest [289]	1992	512	128
N-Hash	Miyaguchi, Ohta, Iwata [237]	1990	128	128
PANAMA	Daemen, Clapp [56]	1998	256	unlimited
Parallel FFT-Hash	Schnorr, Vaudenay [320]	1993	128	128

Macam-macam Fungsi Hash (bonus)

RIPEMD	The RIPE Consortium [287]	1990	512	128
RIPEMD-128	Dobbertin, Bosselaers, Preneel [70]	1996	512	128
RIPEMD-160	Dobbertin, Bosselaers, Preneel [70]	1996	512	160
SHA-0	NIST/NSA [61]	1991	512	160
SHA-1	NIST/NSA [255]	1993	512	160
SHA-224	NIST/NSA [255]	2004	512	224
SHA-256	NIST/NSA [255]	2000	512	256
SHA-384	NIST/NSA [255]	2000	1024	384
SHA-512	NIST/NSA [255]	2000	1024	512
SMASH	Knudsen [177]	2005	256	256
Snefru	Merkle [235]	1990	512-m	m = 128, 256
StepRightUp	Daemen [55]	1995	256	256
Subhash	Daemen [57]	1992	32	up to 256
Tiger	Anderson, Biham [8]	1996	512	192
Whirlpool	Barreto, Rijmen [286]	2000	512	512

Sejarah SHA-256

- Bulan Agustus 1991, NIST mengumumkan standard untuk tanda-tangan digital yang dinamakan *Digital Signature Standard (DSS)*.
- DSS terdiri dari :
  - *Digital Signature Algorithm (DSA)*
  - *Secure Hash Algorithm (SHA)*

Sejarah SHA-256 (lanjutan)

- SHA adalah fungsi hash satu arah yang dibuat oleh NIST (*The National Institute of Standard and Technology*).
- Oleh NSA, SHA dijadikan standard fungsi hash satu arah.
- Salah satu varian SHA adalah **SHA-256** yang akan dibahas lebih lanjut dalam materi ini.

Awal Perkembangan SHA-256

- ~~SHA-0~~ dikenal pada tahun 1991.
- SHA-1 dikenal pada tahun 1993.
- Varian SHA-2 sejak 2000 hingga kini :
  - SHA-224
  - **SHA-256**
  - SHA-384
  - SHA-512
- SHA-3 ? (***Segera!***)

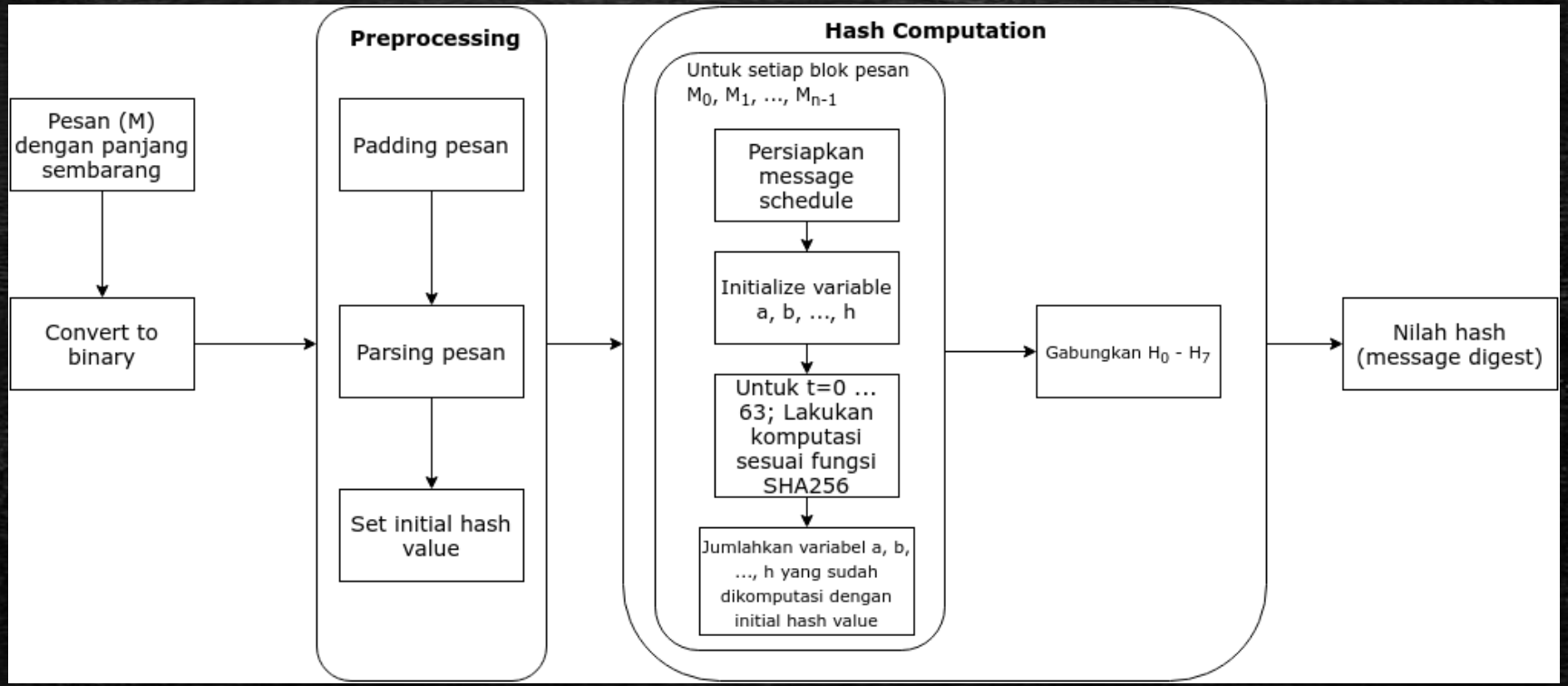
Dasar Prinsip SHA-256

- Pembuatan *message-digest* didasarkan pada pesan dengan panjang maksimum 2^{64} bit.
- Menggunakan message-schedule yang terdiri dari :
  - 64 element word 32-bit.
  - 8 buah variabel 32-bit.
  - Variabel penyimpan nilai hash 8 buah word 32-bit.
- Hasil akhir didapat *message-digest* sepanjang 256 bit.

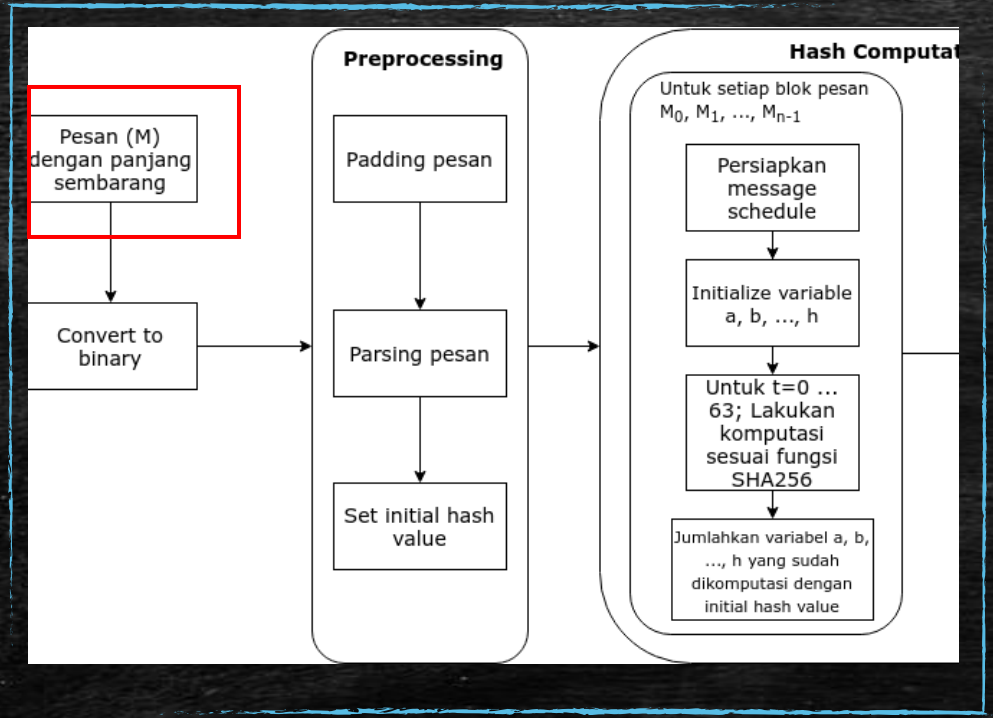
Cara Kerja SHA-256

- SHA-256 mengubah pesan masukan ke dalam message digest 256 bit. Berdasarkan Secure Hash Signature Standard, pesan masukan yang panjangnya lebih pendek dari 2^{64} bit, harus dioperasikan oleh 512 bit dalam kelompok dan menjadi sebuah message diggest 256-bit.

Tahapan SHA-256

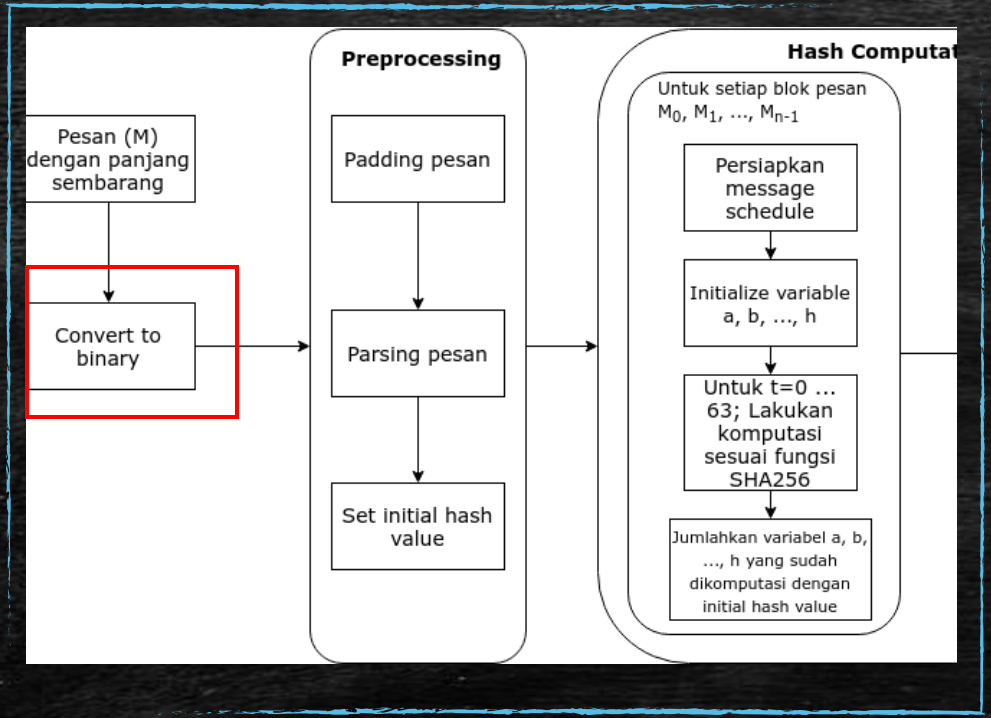


Tahapan SHA-256 (lanjutan)



Misal kita ingin membuat message digest dari sebuah pesan M yang berisi "abc". Maka M = "abc"

Tahapan SHA-256 (lanjutan)

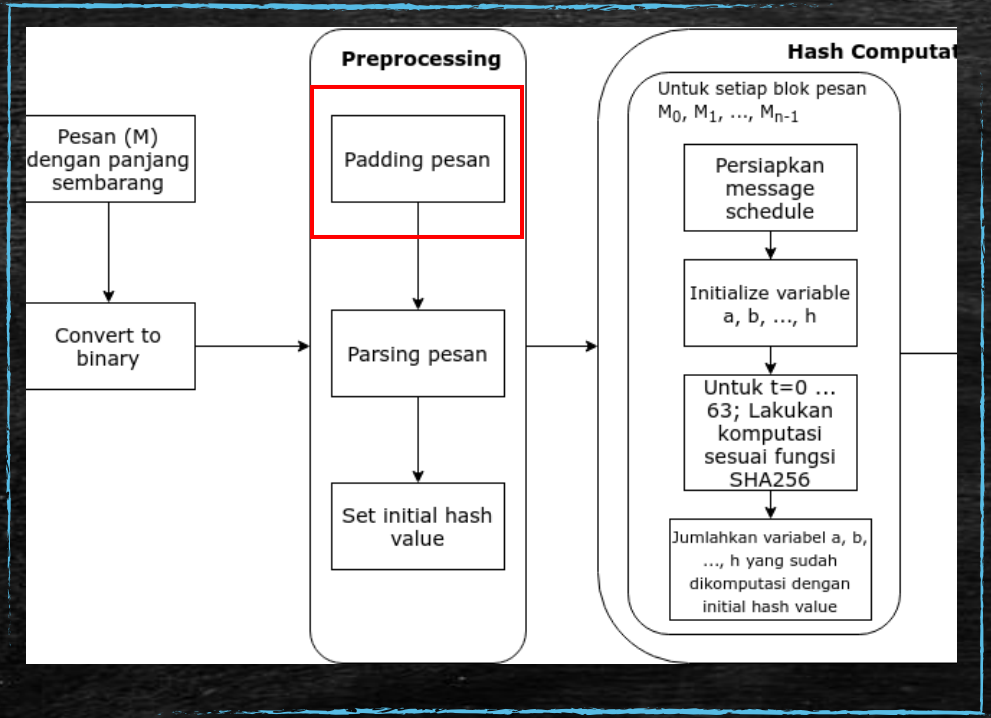


Ubah pesan menjadi binary. Lihat slide selanjutnya !

Tahapan SHA-256 : Ubah pesan ke bentuk binary

- M = "abc"
- 'a' = 01100001
- 'b' = 01100010
- 'c' = 01100011
- Maka :
  - M = 01100001 01100010 01100011

Tahapan SHA-256 (lanjutan)



Padding pesan dengan menambahkan bit-bit pengganjal sehingga total panjangnya 512 bit. Lihat slide selanjutnya !

Note : Meskipun apabila pesan sudah memiliki panjang 512-bit, pesan tersebut tetap dipadding

Gusti Tammam

Tahapan SHA-256 : Padding Pesan

- M = 01100001 01100010 01100011

  8 bit 8 bit 8 bit

- Maka panjang pesan, l =24 bit.
- Padding dilakukan dengan cara menambahkan bit '1' dan sisanya adalah bit '0' sejumlah k. Dimana k merupakan solusi dari persamaan berikut:

$$l + 1 + k \equiv 448 \mod 512$$

Tahapan SHA-256 : Padding Pesan (lanjutan)

- Mencari nilai *k* :
- $k = l + 1 \equiv 448 \bmod 512$

  $k = 24 + 1 \equiv 448 \bmod 512$

  $k = 25 \equiv 448 \bmod 512$

  $k = 448 - 25 = 423$
- Ditemukan nilai *k* = 423. Maka banyak bit '0' yang ditambahkan sejumlah 423 bit.

Tahapan SHA-256 : Padding Pesan (lanjutan)

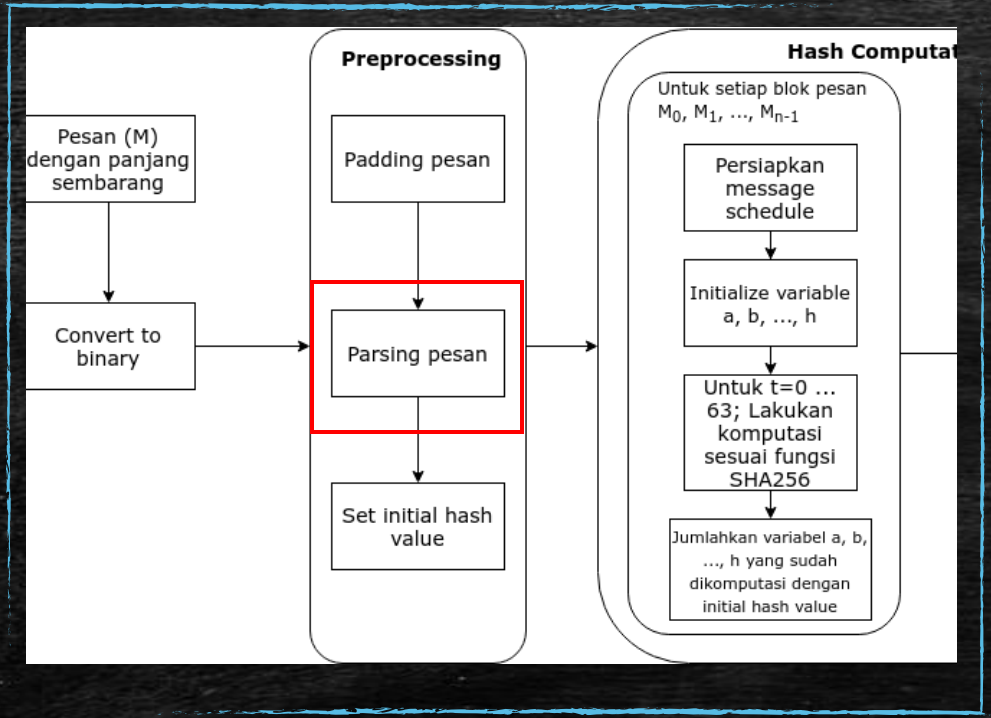
- Setelah itu tambahkan jumlah panjang pesan pada akhir pesan yang di*padding*.
- l = 24 = 00011000

Tahapan SHA-256 : Padding Pesan (lanjutan)

- Hasil pesan yang dipadding [$M^{(0)}$] :

01100001 01100010 01100011 10000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00011000

Tahapan SHA-256 (lanjutan)



Pesan yang dipadding akan menghasilkan blok pesan 512-bit $M^{(0)}$, $M^{(1)}$, ..., $M^{(n-1)}$.

Namun dalam contoh kasus ini karena panjang pesan yang dipadding tidak melebihi 512-bit, maka hanya menghasilkan 1 blok 512-bit yaitu $M^{(0)}$.

Tahap selanjutnya adalah dengan membagi setiap blok 512-bit menjadi 16 buah word 32-bit, $M_0^{(i)}$, $M_1^{(i)}$, ..., $M_{15}^{(i)}$.

Gusti Tammam

Tahapan SHA-256 : Parsing Pesan

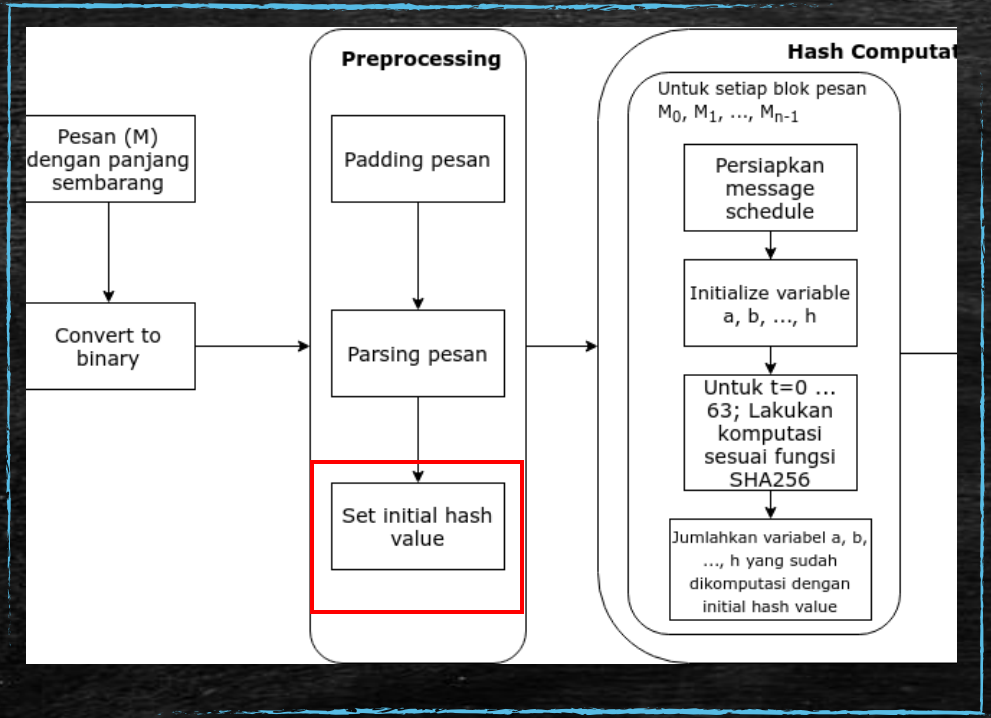
- Hasil pesan yang dipadding [$M^{(0)}$] diparsing :

01100001 01100010 01100011 10000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00011000

Tahapan SHA-256 : Parsing Pesan (lanjutan)

- $M_0^{(0)}$: 0110 0001 0110 0010 0110 0011 1000 0000
- $M_1^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
- $M_2^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
- $M_3^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
- $M_4^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
- $M_5^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
- $M_6^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
- $M_7^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
- $M_8^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
- $M_9^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
- $M_{10}^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
- $M_{11}^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
- $M_{12}^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
- $M_{13}^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
- $M_{14}^{(0)}$: 0000 0000 0000 0000 0000 0000 0000 0000
- $M_{15}^{(0)}$: 0000 0000 0000 0000 0000 0000 0001 1000

Tahapan SHA-256 (lanjutan)



Initial hash value untuk SHA-256 adalah sebagai berikut :

$H_0^{(0)}$ = 6a09e667

$H_1^{(0)}$ = bb67ae85

$H_2^{(0)}$ = 3c6ef372

$H_3^{(0)}$ = a54ff53a

$H_4^{(0)}$ = 510e527f

$H_5^{(0)}$ = 9b05688c

$H_6^{(0)}$ = 1f83d9ab

$H_7^{(0)}$ = 5be0cd19

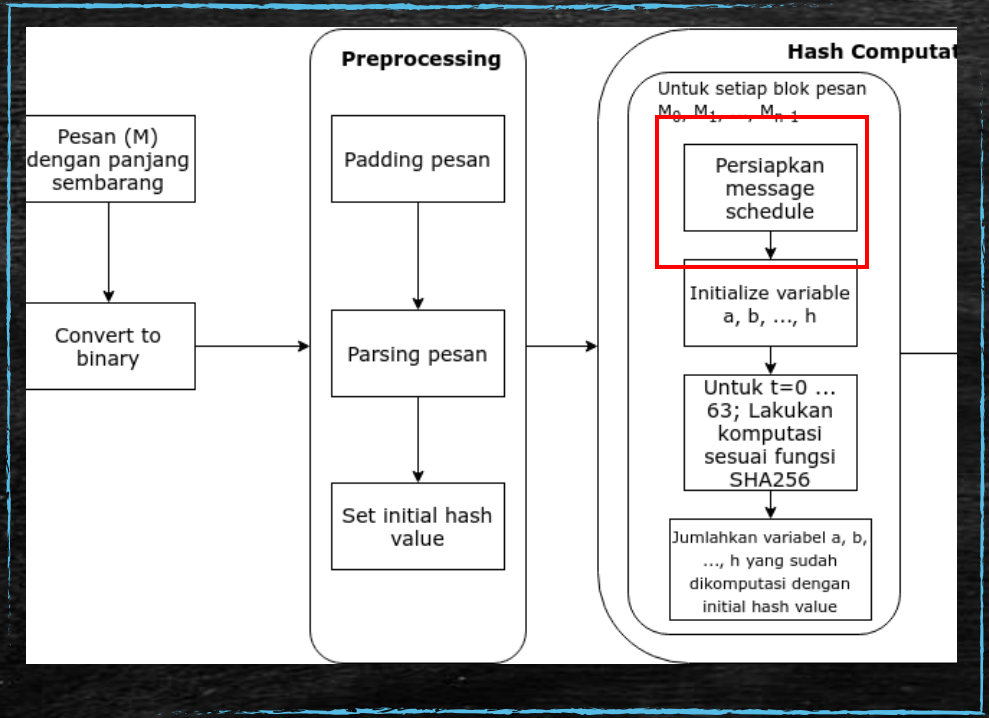
Tahapan SHA-256 (lanjutan)

$$
\begin{aligned}
Ch(x,y,z) &= (x \wedge y) \oplus (\neg x \wedge z) \\
Maj(x,y,z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)
\end{aligned}
$$

$$
\begin{aligned}
\sum_0^{\{256\}}(x) &= ROTR^{2}(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x) \\
\sum_1^{\{256\}}(x) &= ROTR^{6}(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x) \\
\sigma_0^{\{256\}}(x) &= ROTR^{7}(x) \oplus ROTR^{18}(x) \oplus SHR^{3}(x) \\
\sigma_1^{\{256\}}(x) &= ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)
\end{aligned}
$$

SHA-256 Function

Tahapan SHA-256 (lanjutan)



Terdiri dari 64 buah 32-bit word.

Tahapan SHA-256 : Prepare Message Schedule (lanjutan)

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1^{\{256\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{256\}}(W_{t-15}) + W_{t-16} & 16 \le t \le 63 \end{cases}$$

Tahapan SHA-256 : Prepare Message Schedule (lanjutan)

- $M_0^{(0)}$: 61626380
- $M_1^{(0)}$: 00000000
- $M_2^{(0)}$: 00000000
- $M_3^{(0)}$: 00000000
- $M_4^{(0)}$: 00000000
- $M_5^{(0)}$: 00000000
- $M_6^{(0)}$: 00000000
- $M_7^{(0)}$: 00000000
- $M_8^{(0)}$: 00000000
- $M_9^{(0)}$: 00000000
- $M_{10}^{(0)}$: 00000000
- $M_{11}^{(0)}$: 00000000
- $M_{12}^{(0)}$: 00000000
- $M_{13}^{(0)}$: 00000000
- $M_{14}^{(0)}$: 00000000
- $M_{15}^{(0)}$: 00000018

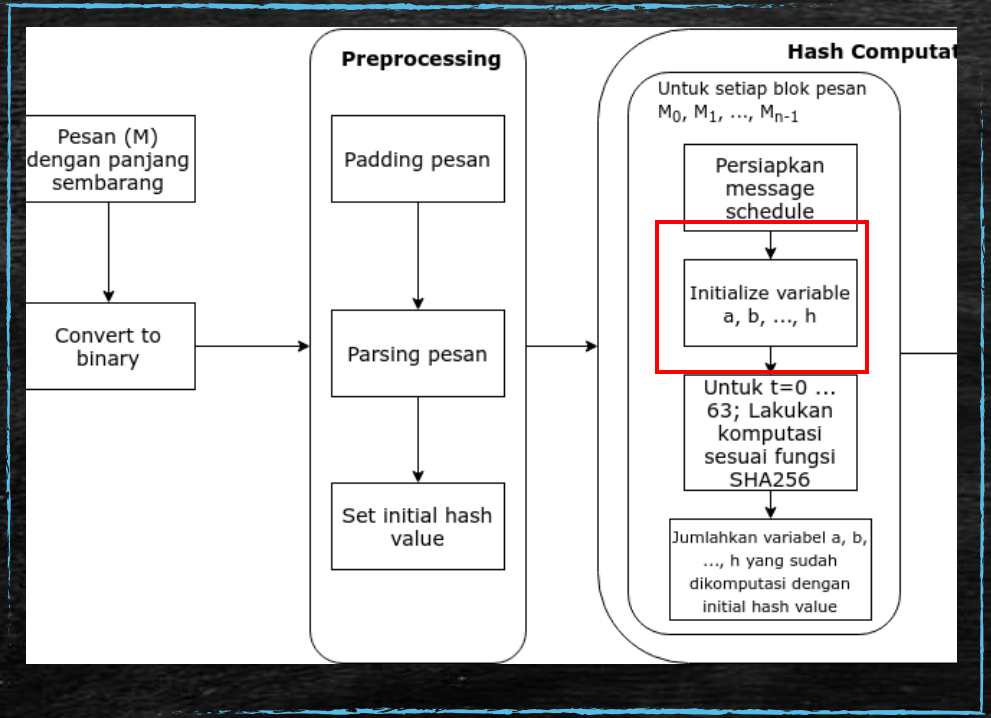
Gusti Tammam

Tahapan SHA-256 : Prepare Message Schedule (lanjutan)

- $W_0^{(0)}$: 61626380
- $W_1^{(0)}$: 00000000
- $W_2^{(0)}$: 00000000
- $W_3^{(0)}$: 00000000
- $W_4^{(0)}$: 00000000
- $W_5^{(0)}$: 00000000
- $W_6^{(0)}$: 00000000
- $W_7^{(0)}$: 00000000
- $W_8^{(0)}$: 00000000
- $W_9^{(0)}$: 00000000
- $W_{10}^{(0)}$: 00000000
- $W_{11}^{(0)}$: 00000000
- $W_{12}^{(0)}$: 00000000
- $W_{13}^{(0)}$: 00000000
- $W_{14}^{(0)}$: 00000000
- $W_{15}^{(0)}$: 00000018

Dan seterusnya hingga $W_{63}^{(i-1)}$, dimana i adalah jumlah blok 512-bit.

Tahapan SHA-256 (lanjutan)



Working variable diambil dari initial hash value

$a = H_0^{(0)} = 6a09e667$

$b = H_1^{(0)} = bb67ae85$

$c = H_2^{(0)} = 3c6ef372$

$d = H_3^{(0)} = a54ff53a$

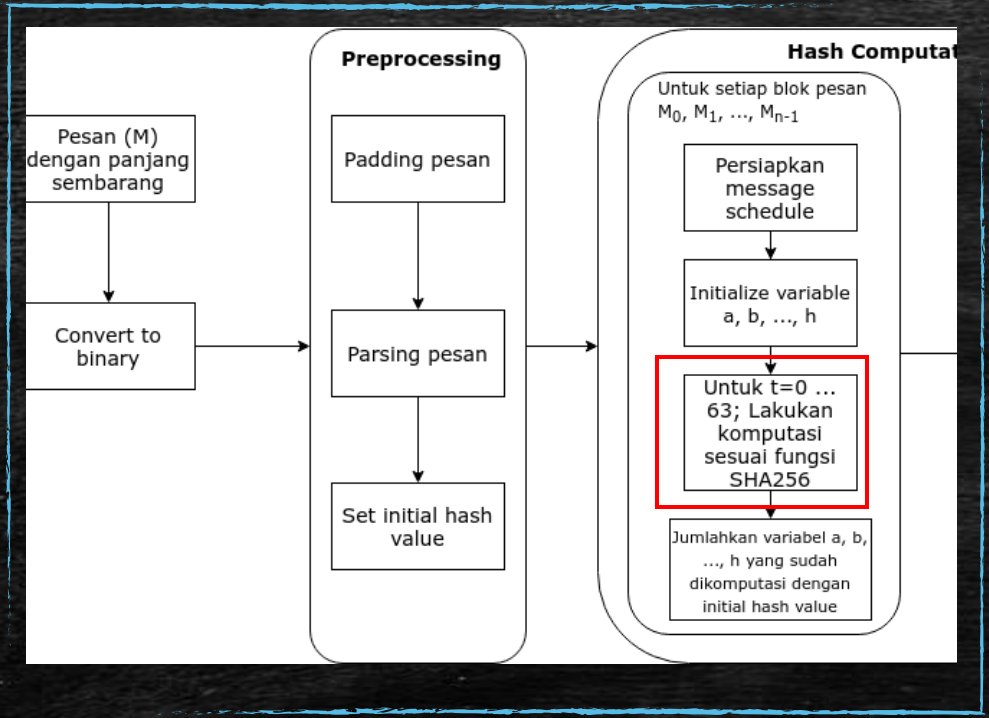
$e = H_4^{(0)} = 510e527f$

$f = H_5^{(0)} = 9b05688c$

$g = H_6^{(0)} = 1f83d9ab$

$h = H_7^{(0)} = 5be0cd19$

Tahapan SHA-256 (lanjutan)



The real computation has begin....

Tahapan SHA-256 : Intermediate Hash Computation

For t=0 to 63:

{

$$T_1 = h + \sum_1^{\{256\}}(e) + Ch(e,f,g) + K_t^{\{256\}} + W_t$$

$$T_2 = \sum_0^{\{256\}}(a) + Maj(a,b,c)$$

$$h = g$$

$$g = f$$

$$f = e$$

$$e = d + T_1$$

$$d = c$$

$$c = b$$

$$b = a$$

$$a = T_1 + T_2$$

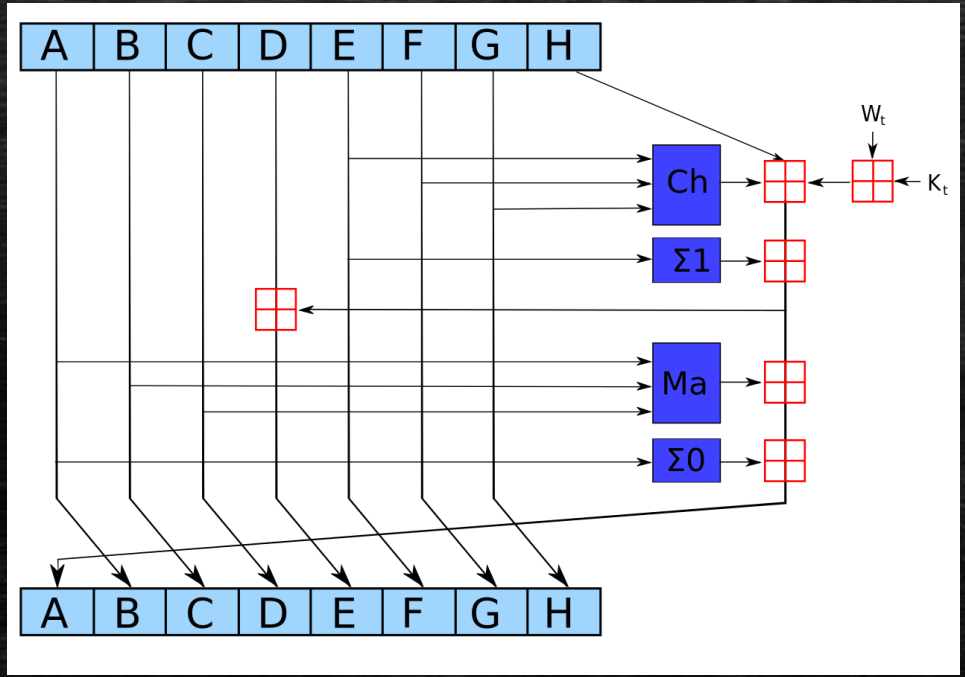
}

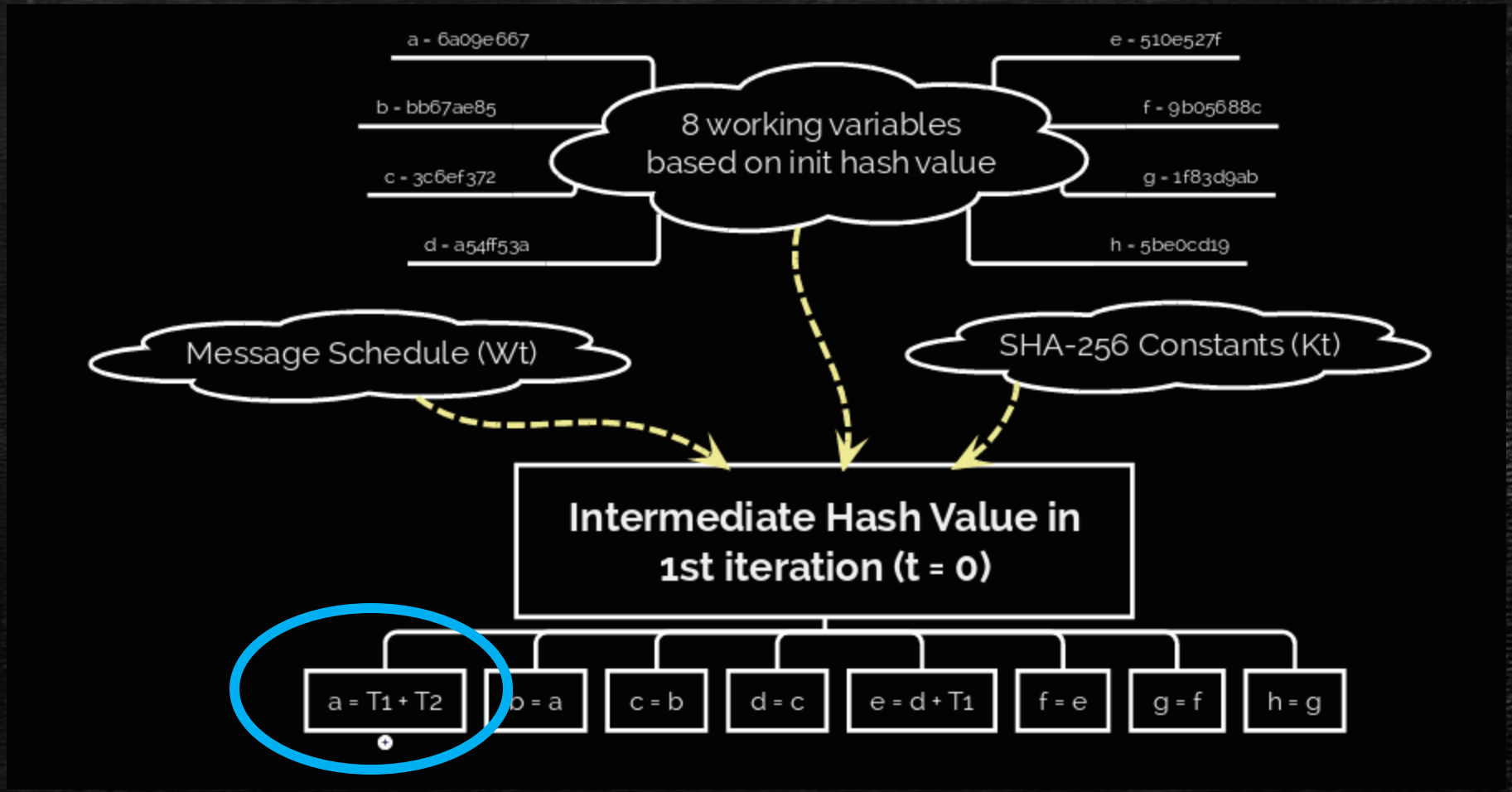
Tahapan SHA-256 : Intermediate Hash Computation

- SHA-256 Constants
- $K_0^{\{256\}}, K_1^{\{256\}}, \ldots, K_{63}^{\{256\}}$

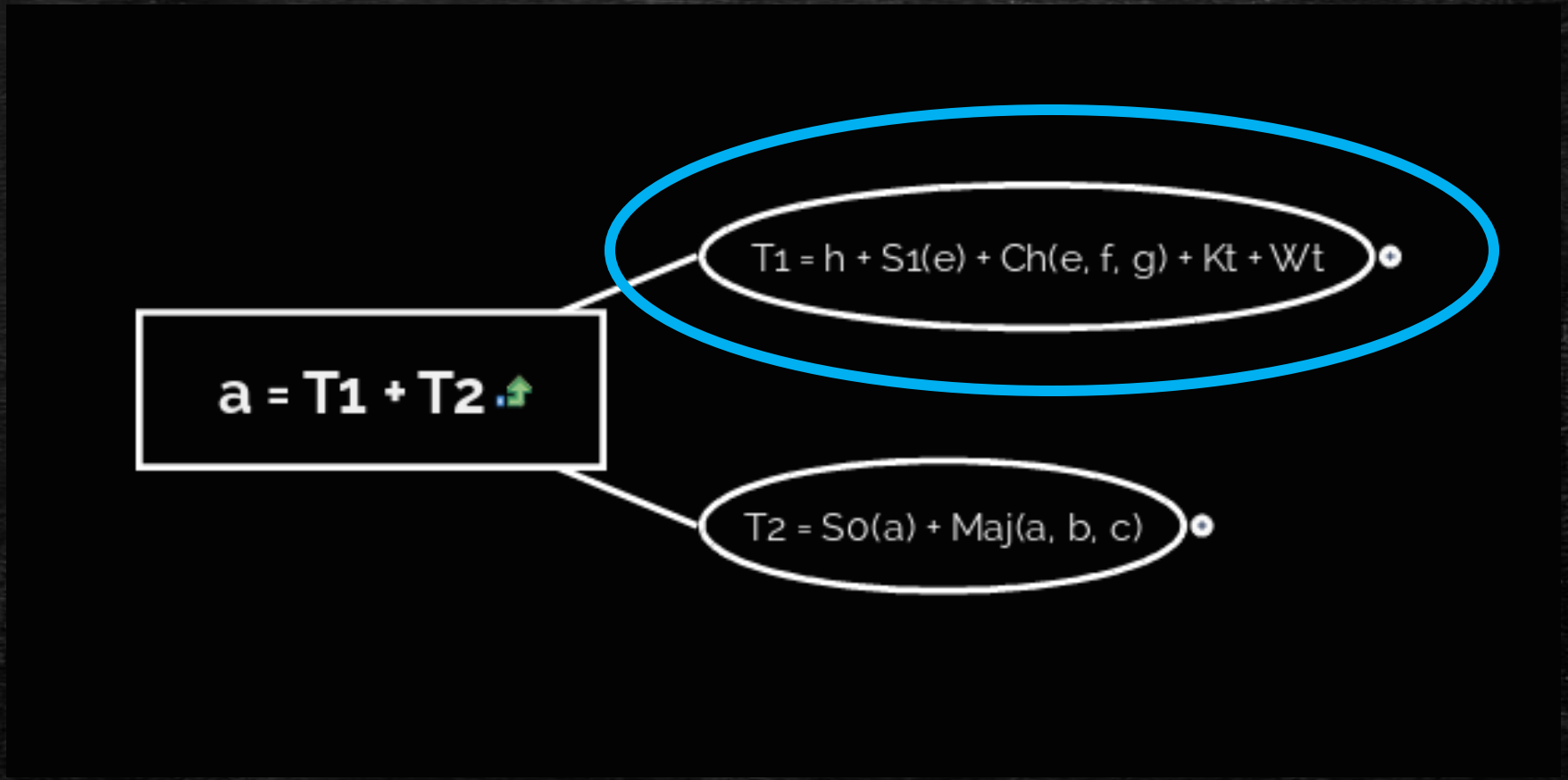
```
428a2f98 71374491 b5c0fbcf e9b5dba5 3956c25b 59f111f1 923f82a4 ab1c5ed5
d807aa98 12835b01 243185be 550c7dc3 72be5d74 80deb1fe 9bdc06a7 c19bf174
e49b69c1 efbe4786 0fc19dc6 240ca1cc 2de92c6f 4a7484aa 5cb0a9dc 76f988da
983e5152 a831c66d b00327c8 bf597fc7 c6e00bf3 d5a79147 06ca6351 14292967
27b70a85 2e1b2138 4d2c6dfc 53380d13 650a7354 766a0abb 81c2c92e 92722c85
a2bfe8a1 a81a664b c24b8b70 c76c51a3 d192e819 d6990624 f40e3585 106aa070
19a4c116 1e376c08 2748774c 34b0bcb5 391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3
748f82ee 78a5636f 84c87814 8cc70208 90befffa a4506ceb bef9a3f7 c67178f2
```

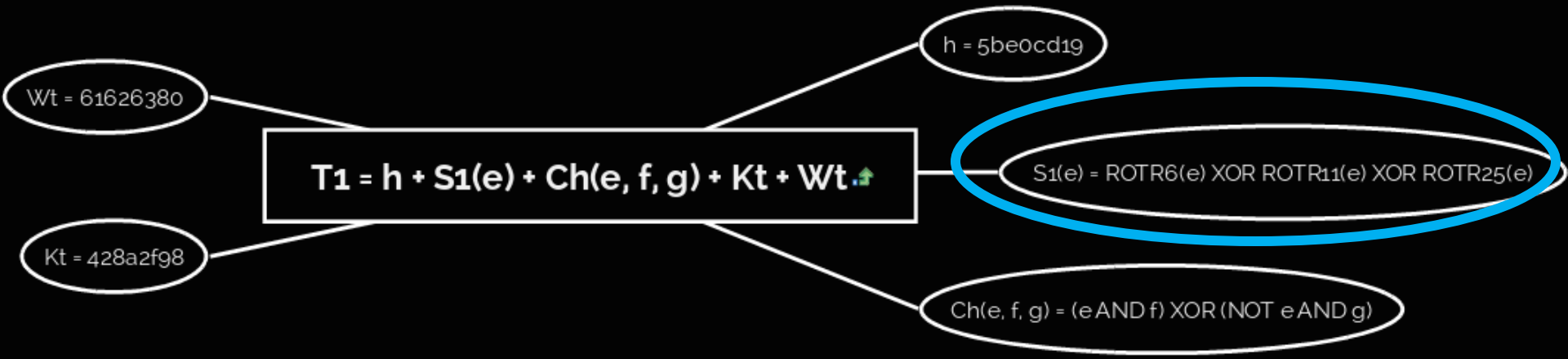
Tahapan SHA-256 : Intermediate Hash Computation





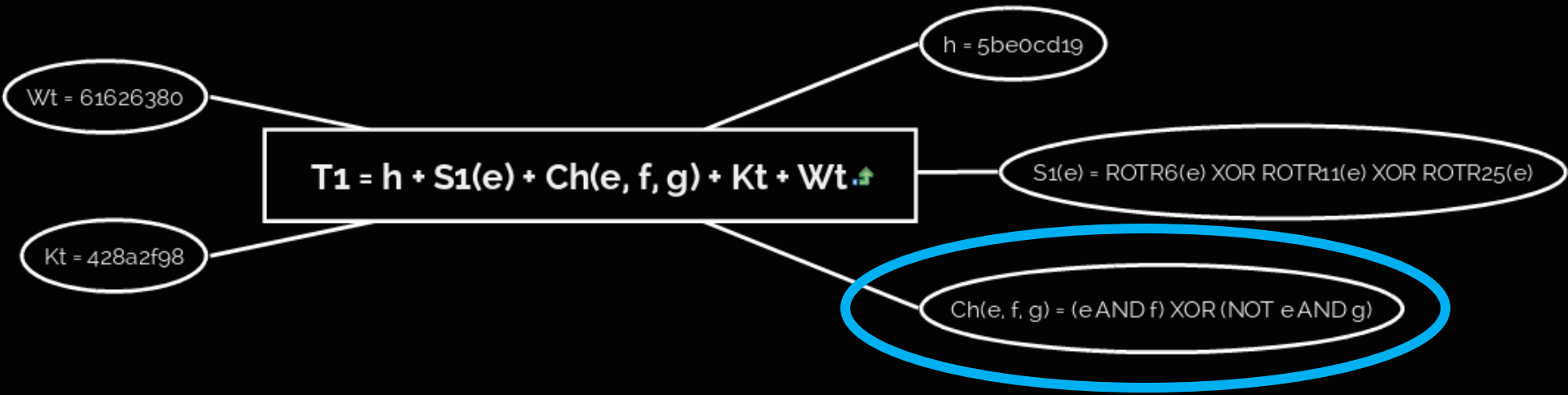
Gusti Tammam


a = T1 + T2
T1 = h + S1(e) + Ch(e, f, g) + Kt + Wt
T2 = S0(a) + Maj(a, b, c)

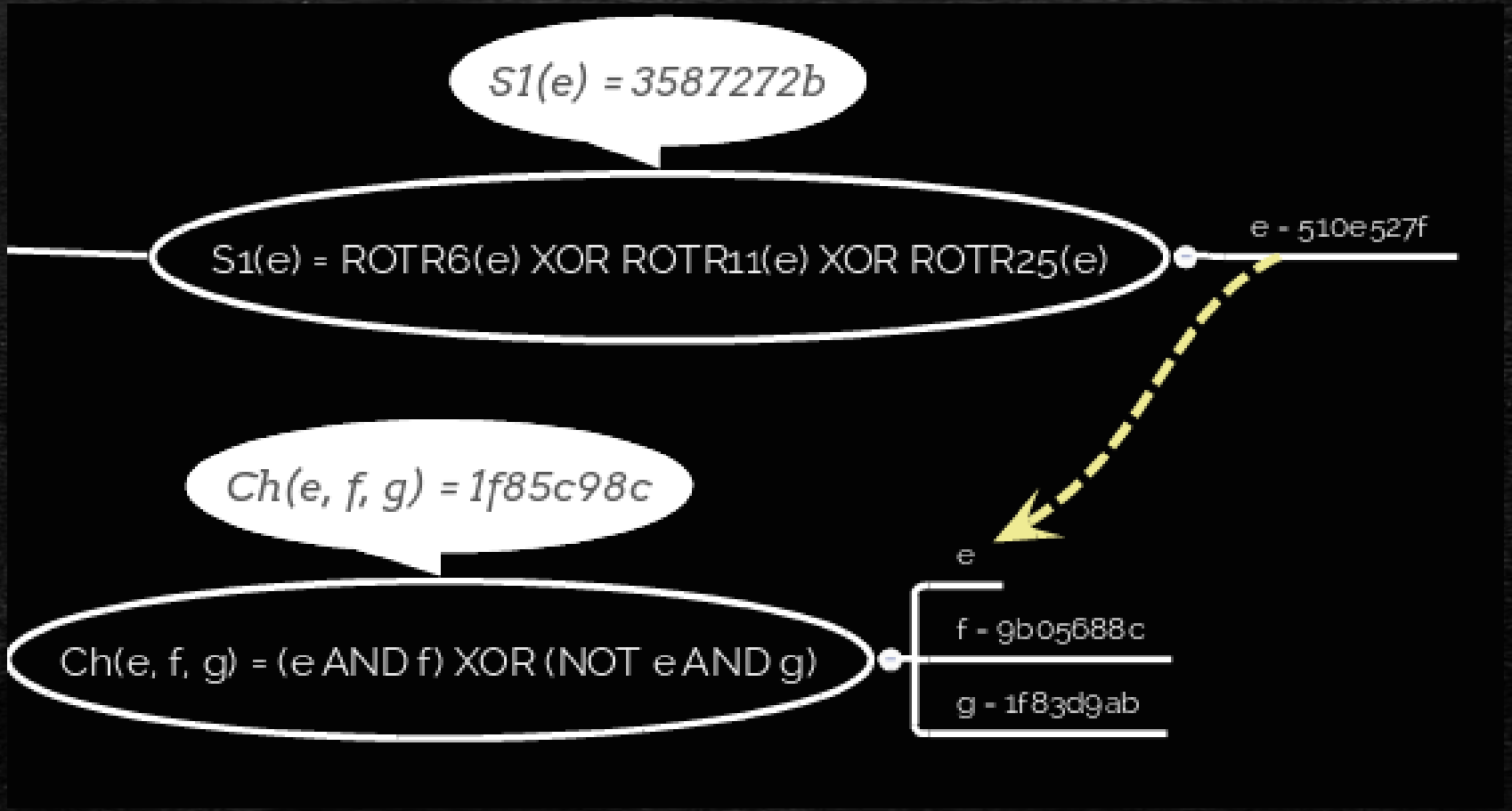

h = 5be0cd19
Wt = 61626380
T1 = h + S1(e) + Ch(e, f, g) + Kt + Wt
S1(e) = ROTR6(e) XOR ROTR11(e) XOR ROTR25(e)
Kt = 428a2f98
Ch(e, f, g) = (e AND f) XOR (NOT e AND g)


S1(e) = ROTR6(e) XOR ROTR11(e) XOR ROTR25(e)
e = 510e527f

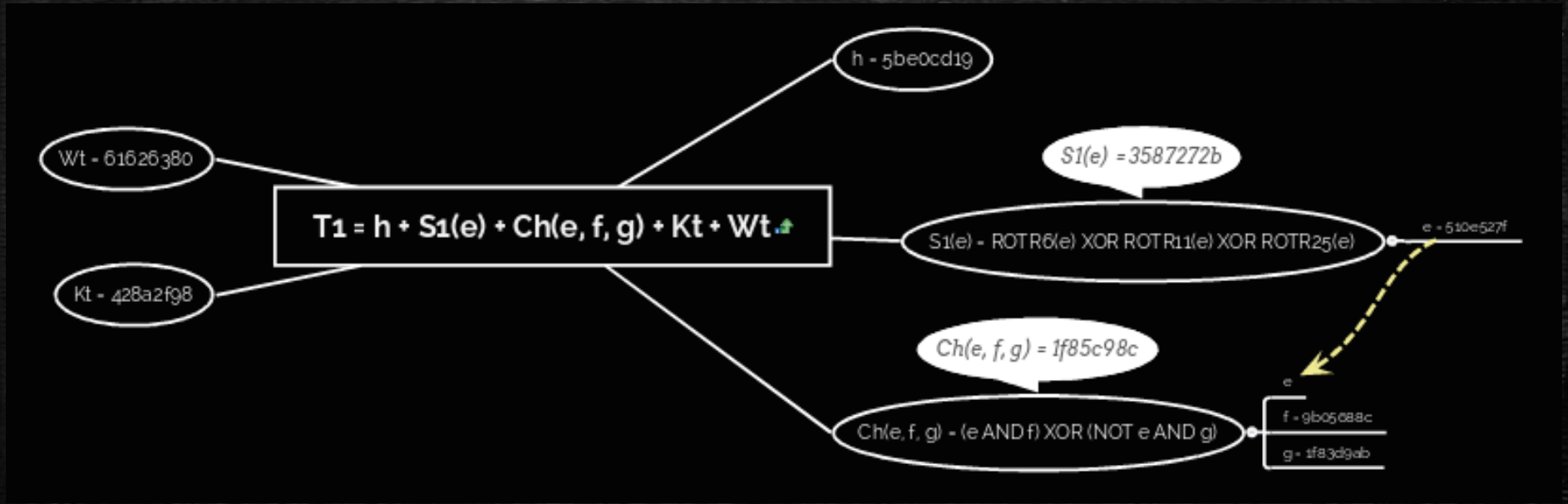

S1(e) = 3587272b
S1(e) = ROTR6(e) XOR ROTR11(e) XOR ROTR25(e)
e - 510e527f


h = 5be0cd19
Wt = 61626380
T1 = h + S1(e) + Ch(e, f, g) + Kt + Wt
S1(e) = ROTR6(e) XOR ROTR11(e) XOR ROTR25(e)
Kt = 428a2f98
Ch(e, f, g) = (e AND f) XOR (NOT e AND g)

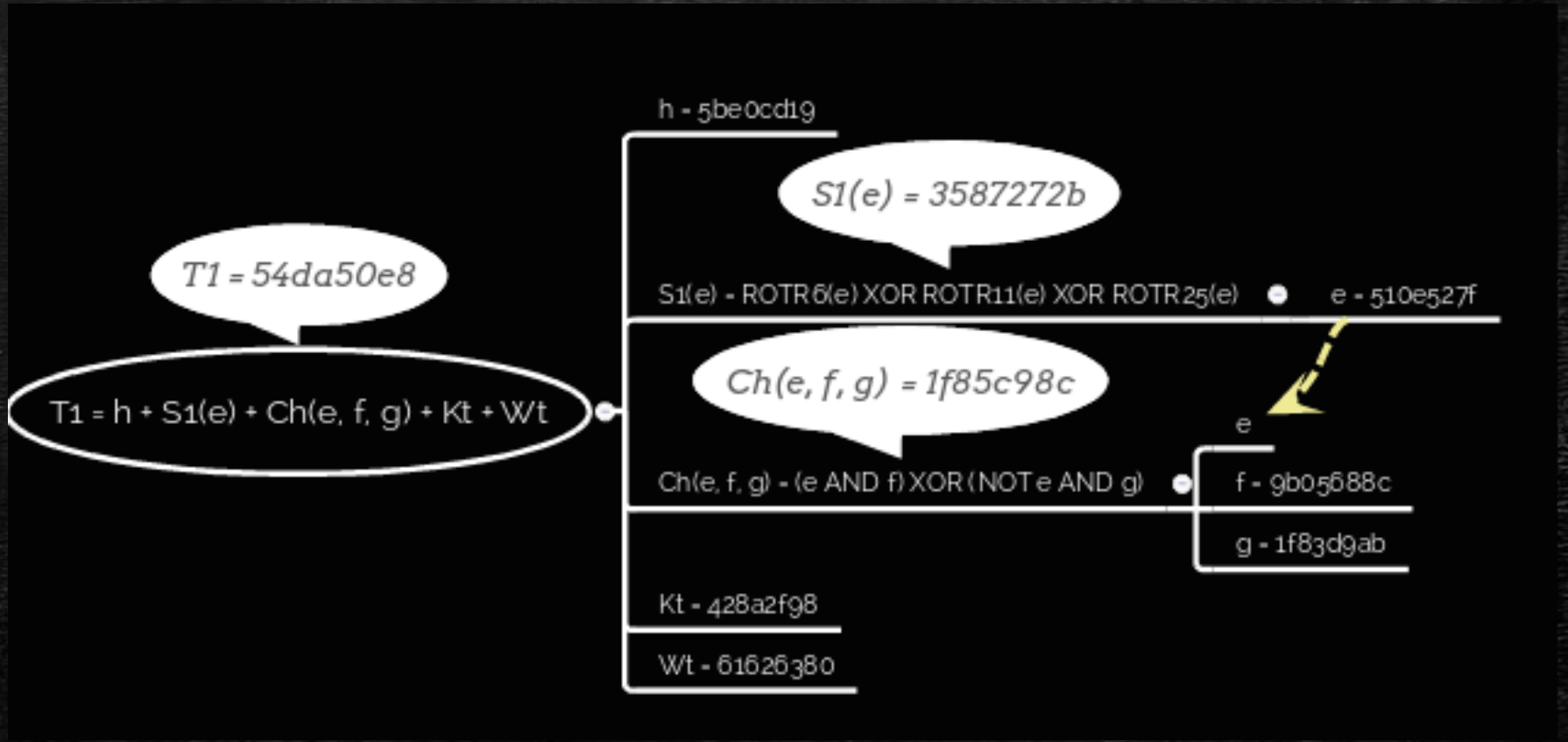
Gusti Tammam



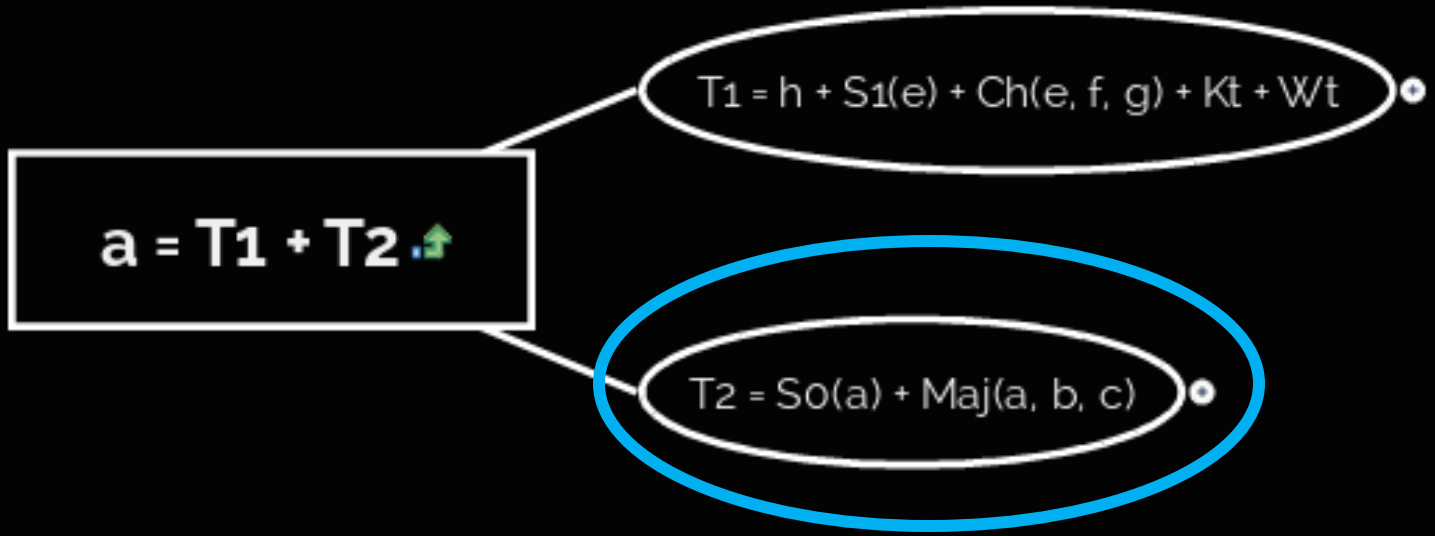
Gusti Tammam



Gusti Tammam

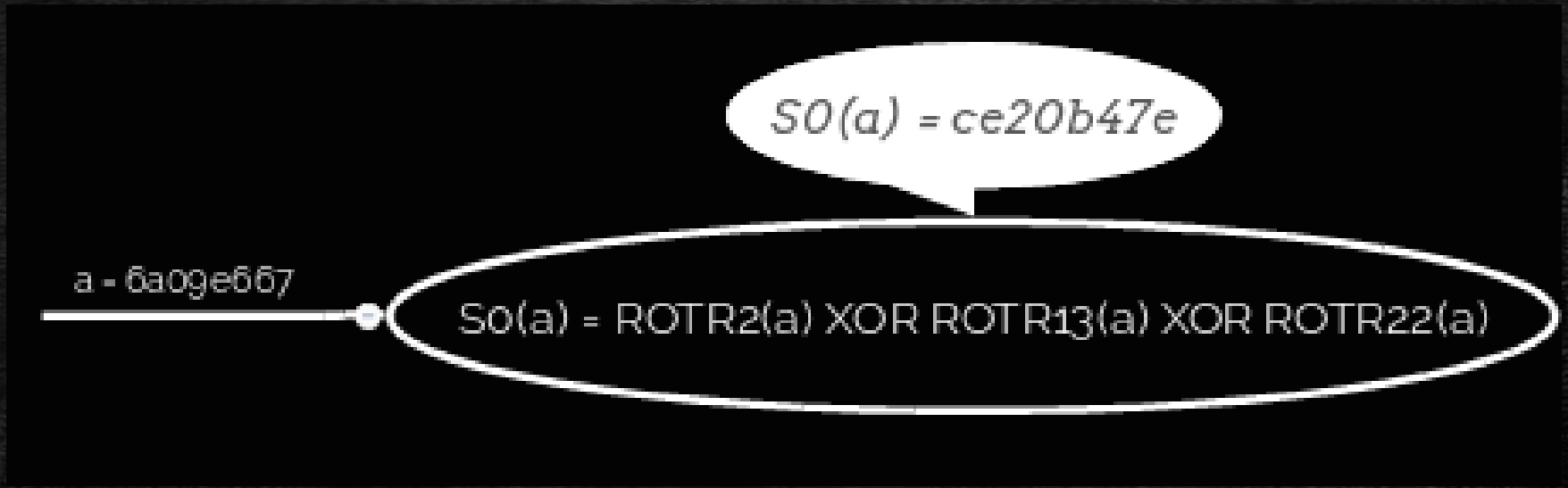


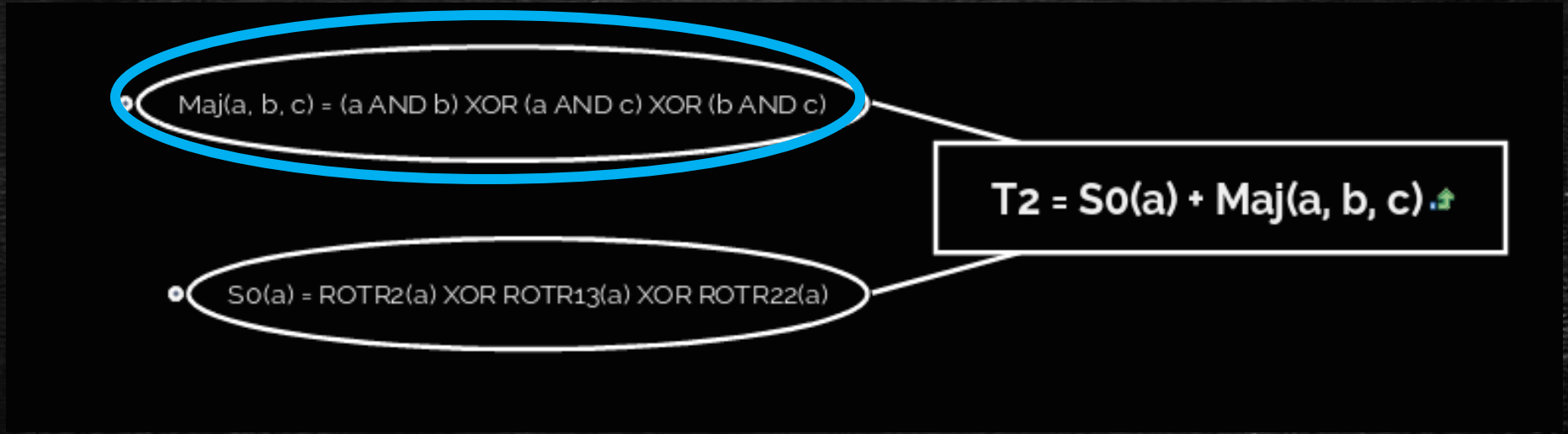
Gusti Tammam

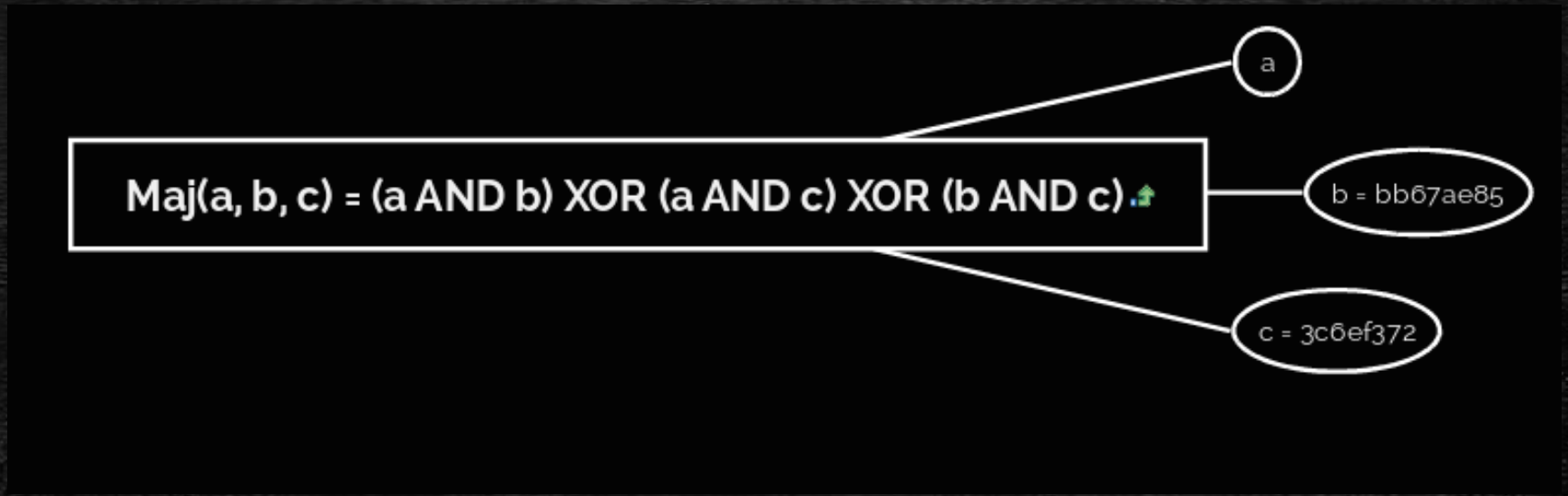

a = T1 + T2
T1 = h + S1(e) + Ch(e, f, g) + Kt + Wt
T2 = S0(a) + Maj(a, b, c)

Gusti Tammam

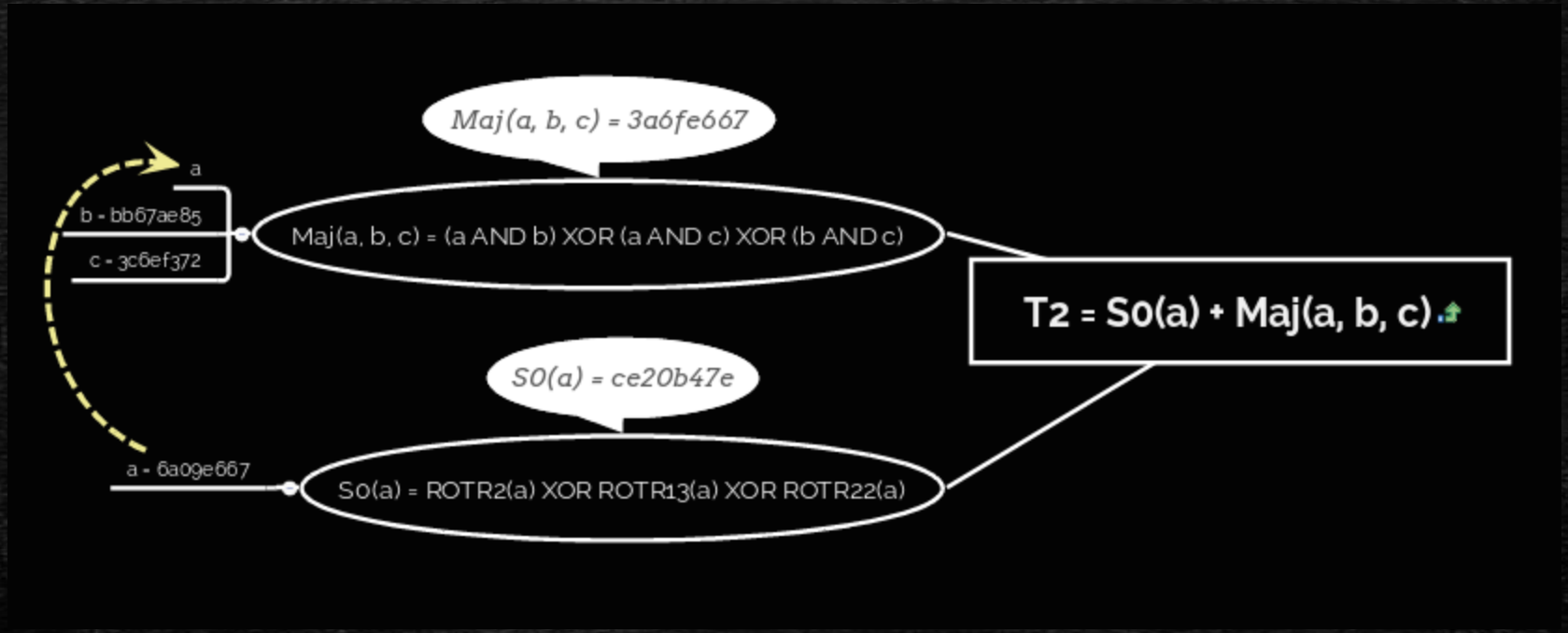

a = 6a09e667
S0(a) = ROTR2(a) XOR ROTR13(a) XOR
ROTR22(a)


S0(a) = ce20b47e
a = 6a09e667
S0(a) = ROTR2(a) XOR ROTR13(a) XOR ROTR22(a)

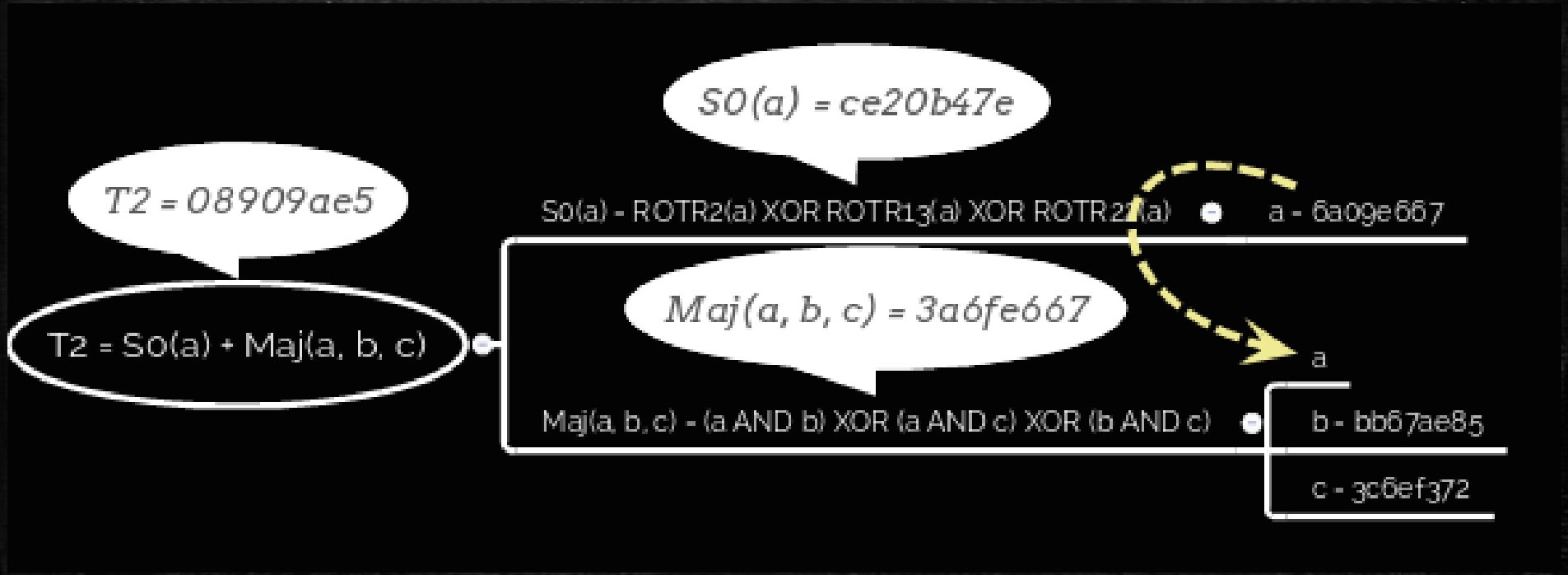

Maj(a, b, c) = (a AND b) XOR (a AND c) XOR (b AND c)
S0(a) = ROTR2(a) XOR ROTR13(a) XOR ROTR22(a)
T2 = S0(a) + Maj(a, b, c)



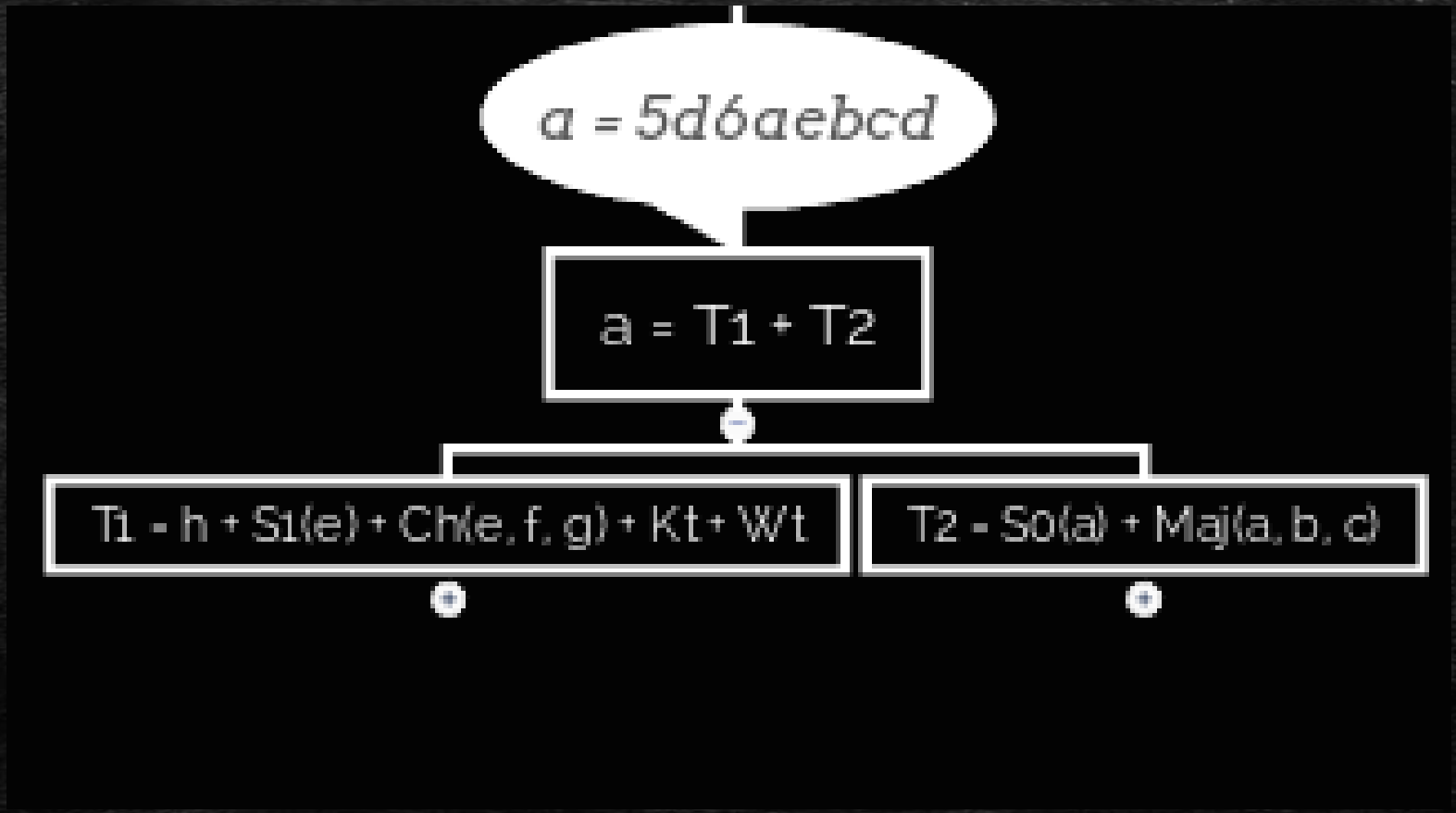
Gusti Tammam

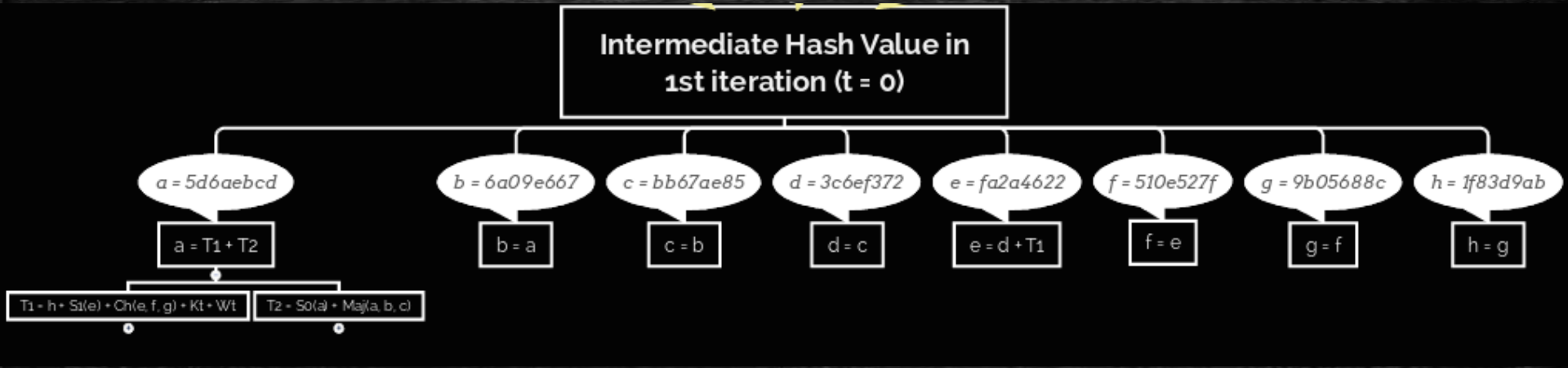


Gusti Tammam

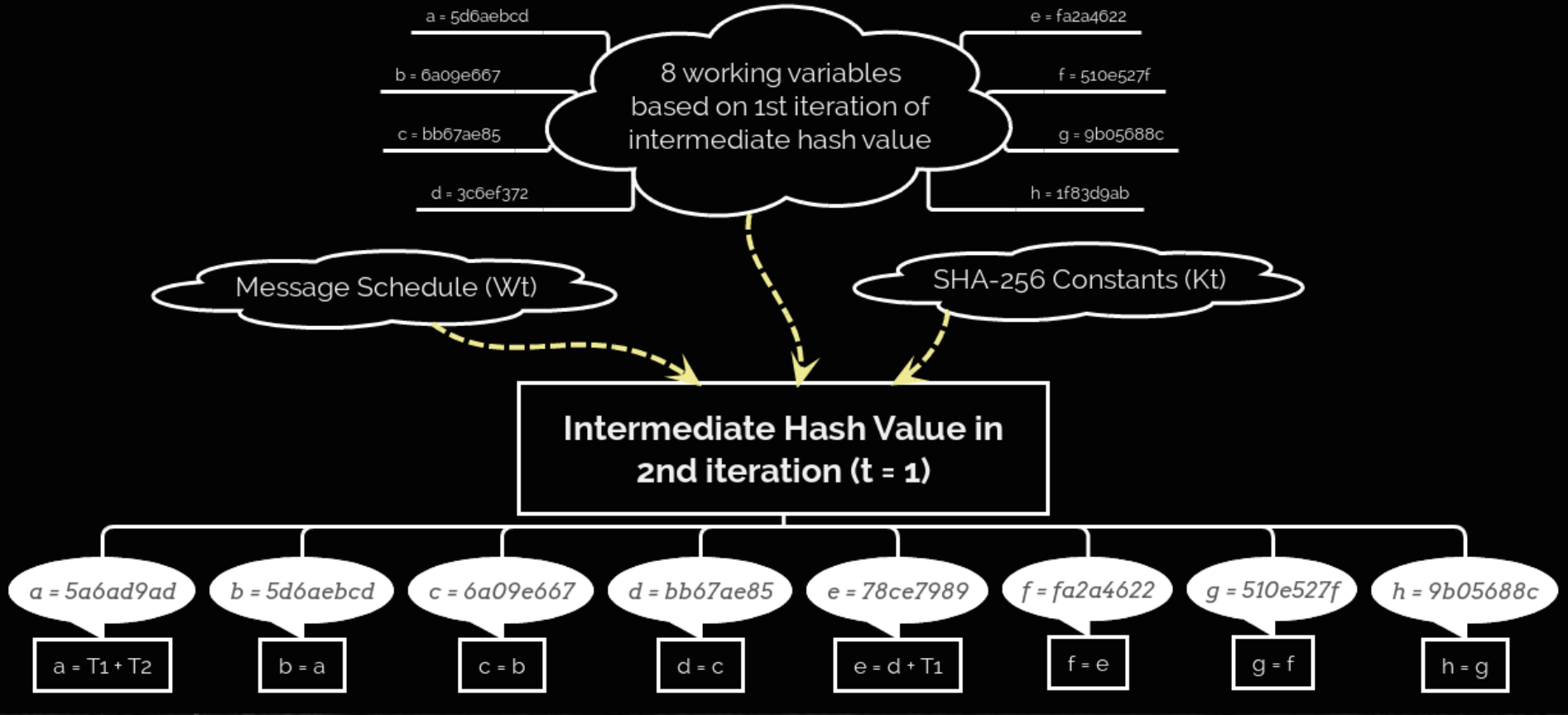


Gusti Tammam


a = 5d6aebcd
a = T1 + T2
T1 = h + S1(e) + Ch(e, f, g) + Kt + Wt
T2 = S0(a) + Maj(a, b, c)



Gusti Tammam

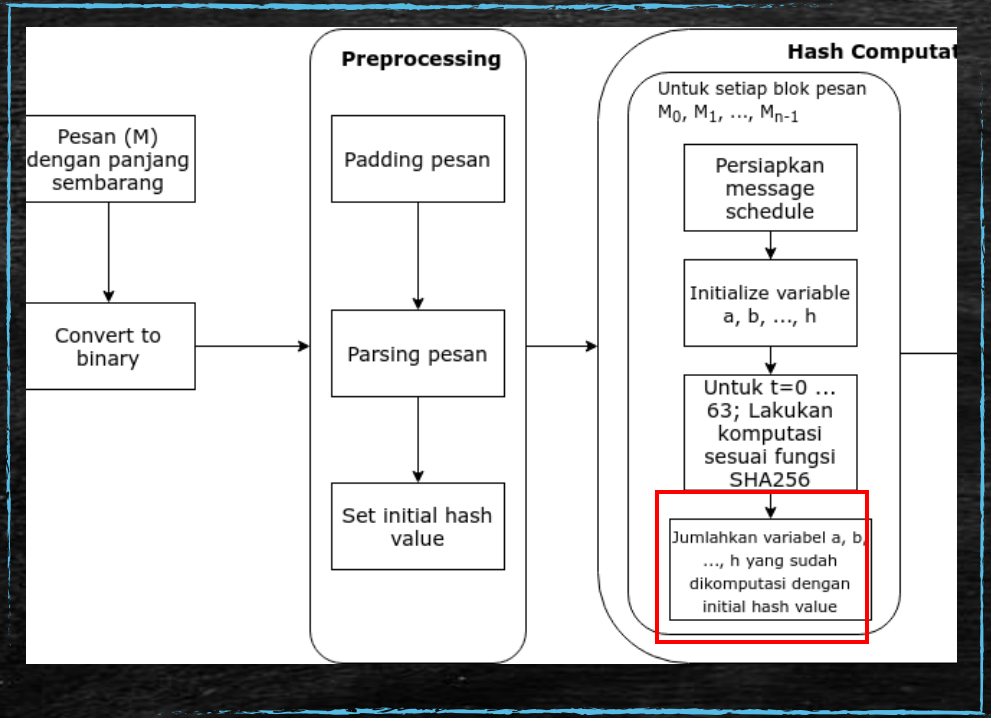


Gusti Tammam

	a	b	c	d	e	f	g	h
init:	6a09e667	bb67ae85	3c6ef372	a54ff53a	510e527f	9b05688c	1f83d9ab	5be0cd19
t = 0	5d6aebcd	6a09e667	bb67ae85	3c6ef372	fa2a4622	510e527f	9b05688c	1f83d9ab
t = 1	5a6ad9ad	5d6aebcd	6a09e667	bb67ae85	78ce7989	fa2a4622	510e527f	9b05688c
t = 2	c8c347a7	5a6ad9ad	5d6aebcd	6a09e667	f92939eb	78ce7989	fa2a4622	510e527f
t = 3	d550f666	c8c347a7	5a6ad9ad	5d6aebcd	24e00850	f92939eb	78ce7989	fa2a4622
t = 4	04409a6a	d550f666	c8c347a7	5a6ad9ad	43ada245	24e00850	f92939eb	78ce7989
t = 5	2b4209f5	04409a6a	d550f666	c8c347a7	714260ad	43ada245	24e00850	f92939eb
t = 6	e5030380	2b4209f5	04409a6a	d550f666	9b27a401	714260ad	43ada245	24e00850
t = 7	85a07b5f	e5030380	2b4209f5	04409a6a	0c657a79	9b27a401	714260ad	43ada245
t = 8	8e04ecb9	85a07b5f	e5030380	2b4209f5	32ca2d8c	0c657a79	9b27a401	714260ad
t = 9	8c87346b	8e04ecb9	85a07b5f	e5030380	1cc92596	32ca2d8c	0c657a79	9b27a401
t = 10	4798a3f4	8c87346b	8e04ecb9	85a07b5f	436b23e8	1cc92596	32ca2d8c	0c657a79
t = 11	f71fc5a9	4798a3f4	8c87346b	8e04ecb9	816fd6e9	436b23e8	1cc92596	32ca2d8c
t = 12	87912990	f71fc5a9	4798a3f4	8c87346b	1e578218	816fd6e9	436b23e8	1cc92596
t = 13	d932eb16	87912990	f71fc5a9	4798a3f4	745a48de	1e578218	816fd6e9	436b23e8
t = 14	c0645fde	d932eb16	87912990	f71fc5a9	0b92f20c	745a48de	1e578218	816fd6e9
t = 15	b0fa238e	c0645fde	d932eb16	87912990	07590dcd	0b92f20c	745a48de	1e578218
t = 16	21da9a9b	b0fa238e	c0645fde	d932eb16	8034229c	07590dcd	0b92f20c	745a48de
t = 17	c2fbd9d1	21da9a9b	b0fa238e	c0645fde	846ee454	8034229c	07590dcd	0b92f20c
t = 18	fe777bbf	c2fbd9d1	21da9a9b	b0fa238e	cc899961	846ee454	8034229c	07590dcd
t = 19	e1f20c33	fe777bbf	c2fbd9d1	21da9a9b	b0638179	cc899961	846ee454	8034229c
t = 20	9dc68b63	e1f20c33	fe777bbf	c2fbd9d1	8ada8930	b0638179	cc899961	846ee454
t = 21	c2606d6d	9dc68b63	e1f20c33	fe777bbf	e1257970	8ada8930	b0638179	cc899961
t = 22	a7a3623f	c2606d6d	9dc68b63	e1f20c33	49f5114a	e1257970	8ada8930	b0638179
t = 23	c5d53d8d	a7a3623f	c2606d6d	9dc68b63	aa47c347	49f5114a	e1257970	8ada8930
t = 24	1c2c2838	c5d53d8d	a7a3623f	c2606d6d	2823ef91	aa47c347	49f5114a	e1257970
t = 25	cde8037d	1c2c2838	c5d53d8d	a7a3623f	14383d8e	2823ef91	aa47c347	49f5114a
t = 26	b62ec4bc	cde8037d	1c2c2838	c5d53d8d	c74c6516	14383d8e	2823ef91	aa47c347
t = 27	77d37528	b62ec4bc	cde8037d	1c2c2838	edffbff8	c74c6516	14383d8e	2823ef91
t = 28	363482c9	77d37528	b62ec4bc	cde8037d	6112a3b7	edffbff8	c74c6516	14383d8e
t = 29	a0060b30	363482c9	77d37528	b62ec4bc	ade79437	6112a3b7	edffbff8	c74c6516
t = 30	ea992a22	a0060b30	363482c9	77d37528	0109ab3a	ade79437	6112a3b7	edffbff8
t = 31	73b33bf5	ea992a22	a0060b30	363482c9	ba591112	0109ab3a	ade79437	6112a3b7

	a	b	c	d	e	f	g	h
t = 32	98e12507	73b33bf5	ea992a22	a0060b30	9cd9f5f6	ba591112	0109ab3a	ade79437
t = 33	fe604df5	98e12507	73b33bf5	ea992a22	59249dd3	9cd9f5f6	ba591112	0109ab3a
t = 34	a9a7738c	fe604df5	98e12507	73b33bf5	085f3833	59249dd3	9cd9f5f6	ba591112
t = 35	65a0cfe4	a9a7738c	fe604df5	98e12507	f4b002d6	085f3833	59249dd3	9cd9f5f6
t = 36	41a65cb1	65a0cfe4	a9a7738c	fe604df5	0772a26b	f4b002d6	085f3833	59249dd3
t = 37	34df1604	41a65cb1	65a0cfe4	a9a7738c	a507a53d	0772a26b	f4b002d6	085f3833
t = 38	6dc57a8a	34df1604	41a65cb1	65a0cfe4	f0781bc8	a507a53d	0772a26b	f4b002d6
t = 39	79ea687a	6dc57a8a	34df1604	41a65cb1	1efbc0a0	f0781bc8	a507a53d	0772a26b
t = 40	d6670766	79ea687a	6dc57a8a	34df1604	26352d63	1efbc0a0	f0781bc8	a507a53d
t = 41	df46652f	d6670766	79ea687a	6dc57a8a	838b2711	26352d63	1efbc0a0	f0781bc8
t = 42	17aa0dfe	df46652f	d6670766	79ea687a	decd4715	838b2711	26352d63	1efbc0a0
t = 43	9d4baf93	17aa0dfe	df46652f	d6670766	fda24c2e	decd4715	838b2711	26352d63
t = 44	26628815	9d4baf93	17aa0dfe	df46652f	a80f11f0	fda24c2e	decd4715	838b2711
t = 45	72ab4b91	26628815	9d4baf93	17aa0dfe	b7755da1	a80f11f0	fda24c2e	decd4715
t = 46	a14c14b0	72ab4b91	26628815	9d4baf93	d57b94a9	b7755da1	a80f11f0	fda24c2e
t = 47	4172328d	a14c14b0	72ab4b91	26628815	fecf0bc6	d57b94a9	b7755da1	a80f11f0
t = 48	05757ceb	4172328d	a14c14b0	72ab4b91	bd714038	fecf0bc6	d57b94a9	b7755da1
t = 49	f11bfaa8	05757ceb	4172328d	a14c14b0	6e5c390c	bd714038	fecf0bc6	d57b94a9
t = 50	7a0508a1	f11bfaa8	05757ceb	4172328d	52f1ccf7	6e5c390c	bd714038	fecf0bc6
t = 51	886e7a22	7a0508a1	f11bfaa8	05757ceb	49231c1e	52f1ccf7	6e5c390c	bd714038
t = 52	101fd28f	886e7a22	7a0508a1	f11bfaa8	529e7d00	49231c1e	52f1ccf7	6e5c390c
t = 53	f5702fdb	101fd28f	886e7a22	7a0508a1	9f4787c3	529e7d00	49231c1e	52f1ccf7
t = 54	3ec45cdb	f5702fdb	101fd28f	886e7a22	e50e1b4f	9f4787c3	529e7d00	49231c1e
t = 55	38cc9913	3ec45cdb	f5702fdb	101fd28f	54cb266b	e50e1b4f	9f4787c3	529e7d00
t = 56	fcd1887b	38cc9913	3ec45cdb	f5702fdb	9b5e906c	54cb266b	e50e1b4f	9f4787c3
t = 57	c062d46f	fcd1887b	38cc9913	3ec45cdb	7e44008e	9b5e906c	54cb266b	e50e1b4f
t = 58	ffb70472	c062d46f	fcd1887b	38cc9913	6d83bfc6	7e44008e	9b5e906c	54cb266b
t = 59	b6ae8fff	ffb70472	c062d46f	fcd1887b	b21bad3d	6d83bfc6	7e44008e	9b5e906c
t = 60	b85e2ce9	b6ae8fff	ffb70472	c062d46f	961f4894	b21bad3d	6d83bfc6	7e44008e
t = 61	04d24d6c	b85e2ce9	b6ae8fff	ffb70472	948d25b6	961f4894	b21bad3d	6d83bfc6
t = 62	d39a2165	04d24d6c	b85e2ce9	b6ae8fff	fb121210	948d25b6	961f4894	b21bad3d
t = 63	506e3058	d39a2165	04d24d6c	b85e2ce9	5ef50f24	fb121210	948d25b6	961f4894

Tahapan SHA-256 (lanjutan)

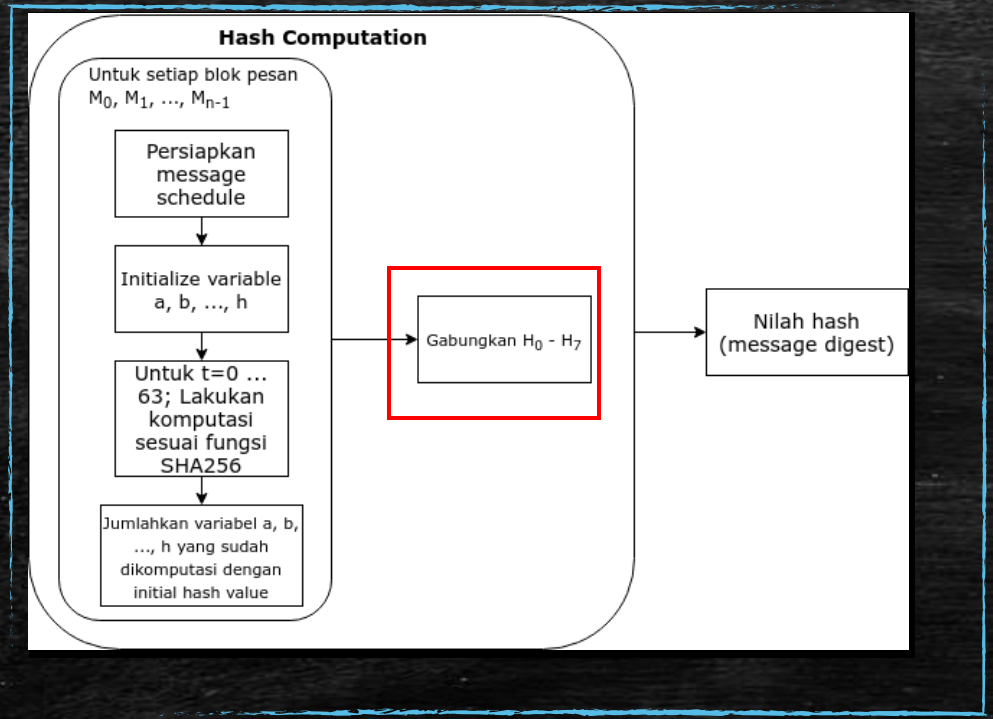


$$H_0^{(i)} = a + H_0^{(i-1)}$$
$$H_1^{(i)} = b + H_1^{(i-1)}$$
$$H_2^{(i)} = c + H_2^{(i-1)}$$
$$H_3^{(i)} = d + H_3^{(i-1)}$$
$$H_4^{(i)} = e + H_4^{(i-1)}$$
$$H_5^{(i)} = f + H_5^{(i-1)}$$
$$H_6^{(i)} = g + H_6^{(i-1)}$$
$$H_7^{(i)} = h + H_7^{(i-1)}$$

Tahapan SHA-256 : Compute Intermediate Hash Value + Init

- $H_0^{(0)}$ = 506e3058 + 6a09e667 = ba7816bf
- $H_1^{(0)}$ = d39a2165 + bb67ae85 = 8f01cfea
- $H_2^{(0)}$ = 04d24d6c + 3c6ef372 = 414140de
- $H_3^{(0)}$ = b85e2ce9 + a54ff53a = 5dae2223
- $H_4^{(0)}$ = 5ef50f24 + 510e527f = b00361a3
- $H_5^{(0)}$ = fb121210 + 9b05688c = 96177a9c
- $H_6^{(0)}$ = 948d25b6 + 1f83d9ab = b410ff61
- $H_7^{(0)}$ = 961f4894 + 5be0cd19 = f20015ad

Tahapan SHA-256 (lanjutan)



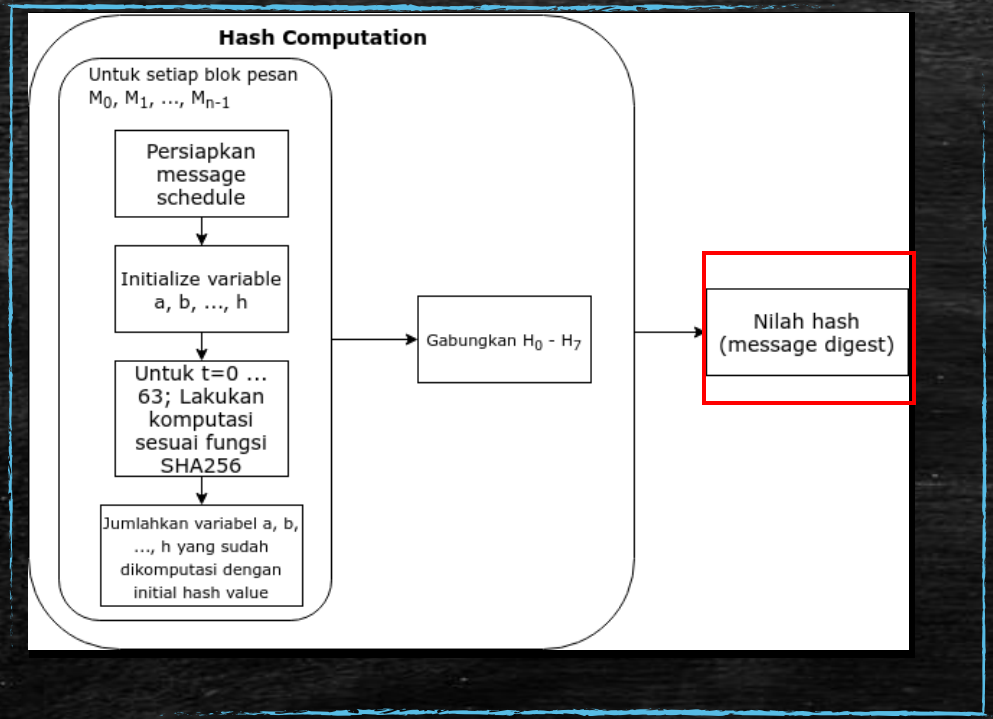
Gabungkan H_0-H_7 !

Tahapan SHA-256 : Gabungkan H_0-H_7

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$$

ba7816bf || 8f01cfea || 414140de || 5dae2223 || b00361a3 || 96177a9c || b410ff61 || f20015ad

Tahapan SHA-256 (lanjutan)



Diakhir tahap didapatkanlah nilai hash atau message digestnya.

Tahapan SHA-256 : Nilai Hash

h **= ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad**

Penerapan SHA-256

- Contoh penerapan SHA-256 pada bahasa pemrograman *Python* silahkan cek *snippet code*-nya di https://github.com/thomdixon/pysha2/blob/master/sha2/sha256.py
- Contoh penerapan SHA-256 dalam aplikasi :
  - Secure Sockets Layer (SSL)
  - Bitcoin

Kelebihan dan Kekurangan

- Untuk mengetahui kelebihan dan kekurangan suatu objek, maka diperlukan objek pembanding lainnya.
- Disini objek pembanding untuk SHA-256 akan dipilih yaitu MD5

SHA-256 Vs MD5

Parameter	SHA-256	MD5
Ketahanan terhadap serangan brute-force	•Message digest dengan panjang 256 bit diyakini mampu menahan serangan brute-force lebih kuat	• Message digest dengan panjang 128 bit cenderung lebih mudah diserang brute force

Score SHA-256 Vs MD5	
1	**0**

SHA-256 Vs MD5

Parameter	SHA-256	MD5
Kerentanan terhadap serangan cryptanalysis	•Tidak rentan	• Rentan

Score SHA-256 Vs MD5	
1	**0**

SHA-256 Vs MD5

Parameter	SHA-256	MD5
Kecepatan komputasi	•Proses komputasi untuk menghasilkan 256 bit message digets lebih lambat	•Proses komputasi untuk menghasilkan 128 bit message digest lebih cepat

Score SHA-256 Vs MD5	
0	1

SHA-256 Vs MD5

Parameter	SHA-256	MD5
Kemudahan implementasi	•Keduanya mudah untuk dijelaskan/dideskripsikan dan diimplemntasikan.	

Score SHA-256 Vs MD5	
1	**1**

SHA-256 Vs MD5

Total Score SHA-256 Vs MD5	
3	2

- SHA-256 **lebih unggul** dari MD5

Kesimpulan

- Fungsi hash memiliki algoritma yang iterative dan searah, yang dapat memproses pesan yang diberikan untuk menghasilkan representasi yang lebih pendek yang disebut message digest.
- Fungsi hash dapat digunakan untuk berbagai kebutuhan yang berkaitan dengan autentifikasi dan tanda tangan digital.
- Fungsi hash haruslah tidak memungkinkan seseorang untuk mengembalikan nilai hash menjadi pesan semula.

Kesimpulan

- Secara umum tahapan SHA-256 terdiri dari *Preprocessing* dan *Hash Computation*.
- Algoritma SHA-256 menghasilkan pesan ringkas dengan panjang 256-bit.
- Algoritma SHA-256 termasuk dalam algoritma satu-arah yang cukup kuat dan aman dibanding dengan algoritma-algoritma sebelumnya.
- Belum ditemukan collision pada SHA-256.

Terimakasih.

Silahkan bertanya !

Referensi

- [1] Angga, Christian. 2011. Analisis Cara Kerja Beragam Fungsi Hash yang Ada. Bandung: Institut Teknologi Bandung.
- [2] Munir, Rinaldi. 2004. Fungsi Hash Satu-Arah dan Algoritma MD5. Bandung: Institut Teknologi Bandung.
- [3] PC Magazine Encyclopedia. Definition of : Message Integrity. Didapat dari : http://www.pcmag.com/encyclopedia/term/46823/message-integrity. Diakses pada : 18 April 2016.
- [4] Ferilli, Stefano. 2011. Automatic Digital Document Processing and Management : Problems, Algorithms, and Techniques. London: Springer.
- [5] Munir, Rinaldi. 2004. Digital Signature Standard (DSS). Bandung: Institut Teknologi Bandung.
- [6] Insani, Agus Yoga. 2008. Proteksi Akses File Executable Menggunakan Sistem Keamanan Teknologi USB Flash Disk. Bandung: Universitas Komputer Indonesia.
- [7] Rodriguez-Henriquez, Francisco. 2006. Cryptographic Algorithms on Reconfigurable Hardware. New York: Springer.
- [8] Sebastian, Amudi. 2007. Implementasi dan Perbandingan Performa Algoritma Hash SHA-1, SHA-256, dan SHA-512. Bandung: Institut Teknologi Bandung.
- [9] Mankar, R.V dan Nipanikar, S. I. 2013. C Implementation of SHA-256 Algorithm. Pune: Pune University.
- [10] Anonim. Descriptions of SHA-256, SHA-384, and SHA-512. Didapat dari : http://www.iwar.org.uk/comsec/resources/cipher/sha256-384-512.pdf. Diakses pada 19 April 2016.
- [11] Dixon, Thomas. PySHA2. Didapat dari : https://github.com/thomdixon/pysha2. Diakses pada : 19 April 2016.
- [12] Lee, Chang-Hsing. Applied Cryptography Chapter 9. Didapat dari : http://people.chu.edu.tw/~chlee/Crypto/Crypto9_1p.pdf. Diakses pada : 19 April 2016.