

B L A K E

SHA-3 Candidate Hash Function

www.noersilo.web.id

BLAKE adalah kandidat dari SHA-3. Blake memenuhi kriteria yang ditetapkan oleh NIST, menawarkan jaminan keamanan secara teoritis dan empiris, dan bekerja baik pada perangkat PC. Blake dibangun berdasarkan komponen dipelajari sebelumnya, dipilih untuk mereka saling melengkapi. BLAKE memiliki tiga heritage :

- a. Iteration Mode menggunakan HAIFA, sebuah versi perbaikan dari paradigma Merkle Damgard ditunjukkan oleh Biham dan Dunkelman. Blake menyediakan ketahanan terhadap pesan panjang *second preimage* dan secara eksplisit menangani hashing dengan salt.
- b. Internal Structure menggunakan local wide-pipe, sudah digunakan dengan fungsi hash Lake. Hal itu membuat local collision mustahil dalam fungsi hash Blake.
- c. Compression Algorithm menggunakan versi modifikasi dari Bernstein's stream cipher ChaCha, keamanannya telah secara intensif dianalisis dan kinerjanya sangat baik dan parallelizable kuat.

Mode iterasi HAIFA sangat bermanfaat bagi standar hash yang baru, karena menyediakan random hashing dan struktur resisten terhadap *second pre-image attack*. Struktur LAKE local wipe-pipe adalah cara untuk memberikan jaminan keamanan yang kuat terhadap serangan collision attack. Akhirnya, dipilihlah Cha-Cha Stream cipher (setelah persetujuan dari penulis).

Karakteristik BLAKE :

- a. Message Digest 224, 256, 384, dan 512 bit
- b. Ukuran parameter yang sama pada SHA-2
- c. Mode streaming one-pass
- d. Panjang pesan maksimum setidaknya $2^{64} - 1$ bit

Selain itu, BLAKE dipilih karena :

- a. Menangani secara eksplisit hashing dengan salt
- b. Menjadi parallelizable
- c. Memungkinkan pertukaran kinerja
- d. Cocok untuk lightweight environments

Selain itu, Blake memungkinkan trade-off throughput / area untuk menyesuaikan pelaksanaan untuk perangkat keras yang tersedia, belum termasuk tujuan berikut:

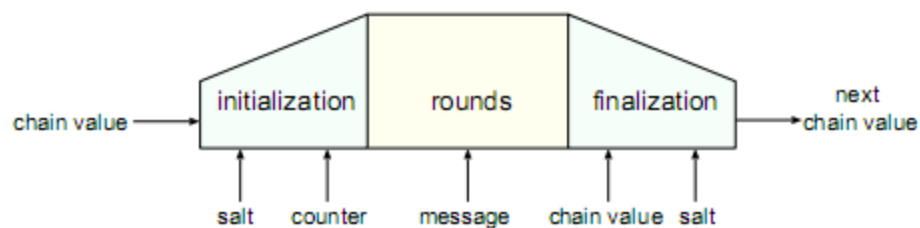
- Memiliki reduksi untuk masalah yang dianggap sulit
- Memiliki sifat homomorphic atau inkremental
- Memiliki desain scalable
- Memiliki spesifikasi untuk panjang variabel hashing

Blake adalah family dari empat fungsi hash: Blake-28, Blake-32, Blake-48, dan Blake-64. Seperti SHA-2, kami memiliki versi 32-bit (Blake-32) dan 64-bit (Blake - 64), dari yang lain yang menggunakan nilai awal yang berbeda, different padding, dan truncated output.

Algorithm	Word	Message	Block	Digest	Salt
BLAKE-28	32	$<2^{64}$	512	224	128
BLAKE-32	32	$<2^{64}$	512	256	128
BLAKE-48	64	$<2^{128}$	1024	384	256
BLAKE-64	64	$<2^{128}$	1024	512	256

Tabel 1: Sifat-sifat fungsi hash Blake (ukuran dalam bit).

BLAKE menggunakan mode iterasi HAIFA, fungsi kompresinya tergantung pada salt dan sepanjang bilangan bit yang dihash (counter), untuk mengkompres setiap pesan menggunakan fungsi yang berbeda. Struktur dari fungsi kompresi Blake diwarisi dari LAKE.



Gambar 1 : lebar lokal pembangunan pipa Blake's kompresi fungsi.

Kekuatan yang diharapkan.

Untuk semua fungsi hash Blake, tidak ada serangan secara signifikan lebih efisien dibandingkan dengan metode bruteforce standar.

- Menemukan collision, dengan salt yang sama atau yang berbeda.
- Menemukan second-preimages, dengan sembarang salt

Keuntungan

a. Desain

- Algoritma yang sederhana
- Interface untuk hashing dengan salt

b. Kinerja

- Cepat pada software dan hardware
- Paralelisme dan throughput / area trade-off untuk implementasi pada hardware
- Kecepatan sederhana / keyakinan trade-off dengan jumlah putaran tunable

c. Keamanan

- Berdasarkan pada komponen analisis intensif (ChaCha)
- Tahan terhadap 2^{nd} preimage attack
- Tahan terhadap side-channel attack
- Tahan terhadap length extension

Pembatasan

- a. Panjang pesan yang terbatas pada masing-masing 2^{64} dan 2^{128} untuk Blake-32 dan Blake-64
- b. Resisten terhadap Joux's multicollisions mirip dengan SHA-2
- c. Fixed-point yang ditemukan dalam waktu selama kurang dari fungsi ideal (tapi tidak efisien)

NOTASI

Symbol	Meaning
\leftarrow	variable assignment
$+$	addition modulo 2^{32} or (modulo 2^{64})
\oplus	Boolean exclusive OR (XOR)
$\ggg k$	rotation of k bits towards less significant bits
$\lll k$	rotation of k bits towards more significant bits
$\langle \ell \rangle_k$	encoding of the integer ℓ over k bits

Table 1.2: Operations symbols used in this document.

SPESIFIKASI

1. BLAKE-32

Hash function BLAKE-32 beroperasi pada 32 bit word dan mengembalikan nilai 32 byte. Bagian ini menunjukkan BLAKE-32, dimulai dari parameter konstan hingga fungsi kompresi, lalu mode iterasi.

1.1 Konstanta

Blake-32 mulai meng-hash dari initial value yang sama seperti SHA-256:

$IV_0 = 6A09E667$	$IV_1 = BB67AE85$
$IV_2 = 3C6EF372$	$IV_3 = A54FF53A$
$IV_4 = 510E527F$	$IV_5 = 9B05688C$
$IV_6 = 1F83D9AB$	$IV_7 = 5BE0CD19$

Blake-32 menggunakan 16 konstanta

$c_0 = 243F6A88$	$c_1 = 85A308D3$
$c_2 = 13198A2E$	$c_3 = 03707344$
$c_4 = A4093822$	$c_5 = 299F31D0$
$c_6 = 082EFA98$	$c_7 = EC4E6C89$
$c_8 = 452821E6$	$c_9 = 38D01377$
$c_{10} = BE5466CF$	$c_{11} = 34E90C6C$
$c_{12} = COAC29B7$	$c_{13} = C97C50DD$
$c_{14} = 3F84D5B5$	$c_{15} = B5470917$

1.2 Fungsi Kompresi

Fungsi kompresi dari Blake-32 menggunakan empat nilai input:

- Rangkaian nilai $h = h_0, \dots, h_7$
- Blok pesan $m = m_0, \dots, m_{15}$
- Salt $s = s_0, \dots, s_3$
- Counter $t = t_0, t_1$

σ_0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
σ_1	14	10	4	8	9	15	13	6	1	12	0	2	11	7	5	3
σ_2	11	8	12	0	5	2	15	13	10	14	3	6	7	1	9	4
σ_3	7	9	3	1	13	12	11	14	2	6	5	10	4	0	15	8
σ_4	9	0	5	7	2	4	10	15	14	1	11	12	6	8	3	13
σ_5	2	12	6	10	0	11	8	3	4	13	7	5	15	14	1	9
σ_6	12	5	1	15	14	13	4	10	0	7	6	3	9	2	8	11
σ_7	13	11	7	14	12	1	3	9	5	0	15	4	8	6	2	10
σ_8	6	15	14	9	11	3	0	8	12	2	13	7	1	4	10	5
σ_9	10	2	8	4	7	6	1	5	15	11	9	14	3	12	13	0

Table 2.1: Permutations of $\{0, \dots, 15\}$ used by the BLAKE functions.

Empat input ini menunjukkan 30 word total (120 bytes = 960 bits). Output fungsi adalah rangkaian baru $h' = h_0', \dots, h_7'$ dari delapan word (32 bytes = 256 bits). Kompresi h, m, s, t ke h' adalah

$$h' = \text{compress}(h, m, s, t)$$

Inisialisasi

16 word v_0, \dots, v_{15} diinisialisasi sehingga hasil input berbeda menghasilkan initial state berbeda.

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix} \leftarrow \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ s_0 \oplus c_0 & s_1 \oplus c_1 & s_2 \oplus c_2 & s_3 \oplus c_3 \\ t_0 \oplus c_4 & t_0 \oplus c_5 & t_1 \oplus c_6 & t_1 \oplus c_7 \end{pmatrix}$$

Fungsi Round

V diinisialisasi, fungsi kompresi mengiterate seri dari 10 rounds. Sebuah round adalah transformasi dari state v , dimana menghitung

$$\begin{matrix} G_0(v_0, v_4, v_8, v_{12}) & G_1(v_1, v_5, v_9, v_{13}) & G_2(v_2, v_6, v_{10}, v_{14}) & G_3(v_3, v_7, v_{11}, v_{15}) \\ G_4(v_0, v_5, v_{10}, v_{15}) & G_5(v_1, v_6, v_{11}, v_{12}) & G_6(v_2, v_7, v_8, v_{13}) & G_7(v_3, v_4, v_9, v_{14}) \end{matrix}$$

Dimana pada round r , $G_i(a, b, c, d)$ dilet

$$\begin{aligned} a &\leftarrow a + b + (m_{\sigma_r(2i)} \oplus c_{\sigma_r(2i+1)}) \\ d &\leftarrow (d \oplus a) \ggg 16 \\ c &\leftarrow c + d \\ b &\leftarrow (b \oplus c) \ggg 12 \\ a &\leftarrow a + b + (m_{\sigma_r(2i+1)} \oplus c_{\sigma_r(2i)}) \\ d &\leftarrow (d \oplus a) \ggg 8 \\ c &\leftarrow c + d \\ b &\leftarrow (b \oplus c) \ggg 7 \end{aligned}$$

Empat pertama disebut G_0, \dots, G_3 dapat dihitung paralel, karena masing-masing dari mereka memperbaharui kolom yang berbeda dari matriks. Prosedur menghitung G_0, \dots, G_3 disebut column step. Dengan cara yang sama empat terakhir disebut G_4, \dots, G_7 memperbaharui diagonal berbeda dan dapat diparalel dengan baik, disebut diagonal step.

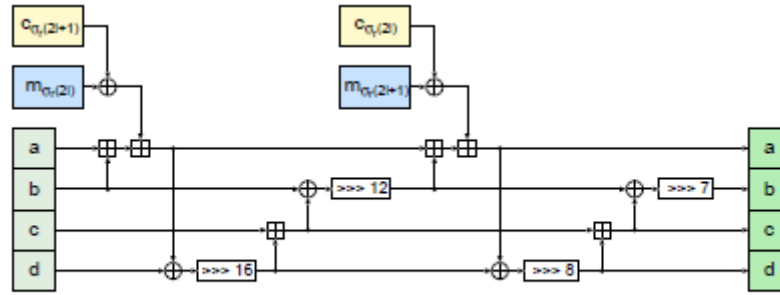


Figure 2.1: The G_i function.

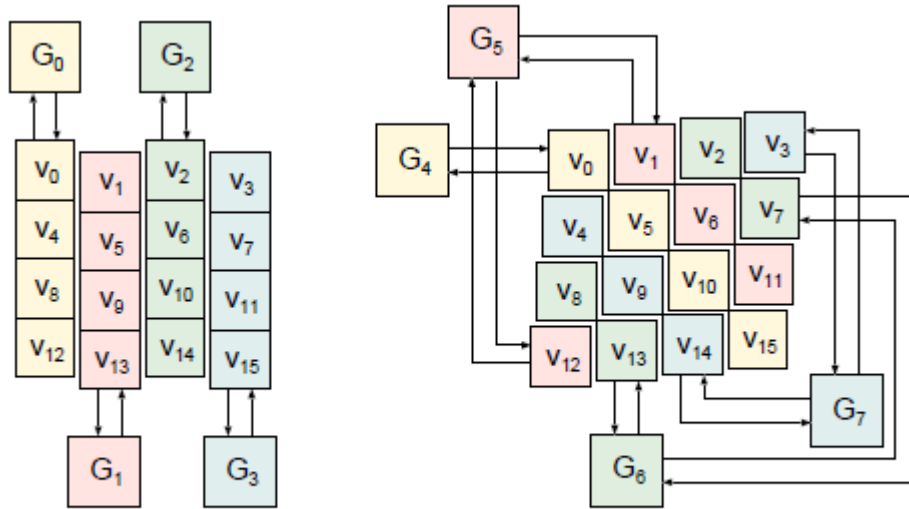


Figure 2.2: Column step and diagonal step.

Finalisasi

Setelah round sequence, nilai rangkaian baru h'_0, \dots, h'_7 diekstraksi dari state v_0, \dots, v_{15} dengan input rangkaian initial value h_0, \dots, h_7 dan salt s_0, \dots, s_3 .

$$\begin{aligned}
 h'_0 &\leftarrow h_0 \oplus s_0 \oplus v_0 \oplus v_8 \\
 h'_1 &\leftarrow h_1 \oplus s_1 \oplus v_1 \oplus v_9 \\
 h'_2 &\leftarrow h_2 \oplus s_2 \oplus v_2 \oplus v_{10} \\
 h'_3 &\leftarrow h_3 \oplus s_3 \oplus v_3 \oplus v_{11} \\
 h'_4 &\leftarrow h_4 \oplus s_0 \oplus v_4 \oplus v_{12} \\
 h'_5 &\leftarrow h_5 \oplus s_1 \oplus v_5 \oplus v_{13} \\
 h'_6 &\leftarrow h_6 \oplus s_2 \oplus v_6 \oplus v_{14} \\
 h'_7 &\leftarrow h_7 \oplus s_3 \oplus v_7 \oplus v_{15}
 \end{aligned}$$

1.3 Hashing Pesan

Prosedur untuk hashing pesan m dari bit length $l < 2^{64}$. Seperti iterated hash function biasanya, pesan pertama kali di pad, lalu diproses blok per blok dengan fungsi kompresi.

Padding

Pertama pesan diperpanjang sehingga panjangnya kongruen dengan 447 modulo 512. Perpanjangan panjang ditunjukkan oleh penambahan bit 1 diikuti bit 0 secukupnya. Paling sedikit satu bit, paling banyak 512 bit. Lalu bit 1 ditambahkan, diikuti 64 bit unsigned big-endian dari l . Padding dapat direpresentasi sebagai

$$m \leftarrow m \parallel 1000 \dots 0001 \langle \ell \rangle_{64}$$

Prosedur ini menjamin panjang bit dari pesan yang dipadd kelipatan 512.

Iterated Hash

Hasil dari iterated hash, pesan yang dipadd dibagi menjadi 16 blok word m^0, \dots, m^{N-1} . l^i adalah banyak bit pesan m^0, \dots, m^i , yaitu mengeluarkan bit ditambah oleh padding. Contoh, pesan yang belum dipadd panjangnya 600 bit, lalu pesan yang dipadd mempunyai dua blok dan $l^0 = 512, l^1 = 600$.

Salt dipilih oleh user, dan diset nilai nol ketika salt tidak dibutuhkan ($s_0 = s_1 = s_2 = s_3 = 0$). Meng-hash pesan yang dipadd m sebagai berikut:

```
 $h^0 \leftarrow IV$   
for  $i = 0, \dots, N - 1$   
     $h^{i+1} \leftarrow \text{compress}(h^i, m^i, s, l^i)$   
return  $h^N$ 
```

Prosedur hashing m dengan Blake-32 alias $\text{Blake-32}(m, s) = h^N$, dimana m adalah pesan yang belum dipadd, dan s adalah salt. Notasi $\text{Blake-32}(m)$ menunjukkan hash dari m ketika salt tidak digunakan ($s=0$).