Analisis dan Perbandingan Algoritma Fungsi Hash SHA-2 256 dan Keccak

Abdurrisyad Fikri - 13508017
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18017@students.if.itb.ac.id, abdurrisyadfikri@yahoo.com

***Abstract*—Fungsi Hash adalah fungsi yang menerima masukan berupa string yang panjangnya sembarang dan mengonversi masukan tersebut menjadi string yang mempunyai panjang yang tetap (*fixed*) dan umumnya menjadi menjadi lebih kecil dari panjang semula. Saat ini ada banyak fungsi hash yang umum digunakan salah satunya adalah SHA (Secure Hash Algorithm) yang dikeluarkan oleh NIST (National Institute of Standard and Technology). Saat ini sudah terdapat beberapa varian dari SHA yaitu SHA-0, SHA-1, dan SHA-2. Untuk SHA-3 saat ini masih dilakukan kontes untuk memilih salah satu dari beberapa kandidat terkuat/finalis yang akan dijadikan standar SHA-3. Algoritma-algoritma yang berhasil menjadi finalis untuk kontes SHA-3 ini salah satunya adalah algoritma Keccak. Untuk mengetahui seberapa unggul algoritma Keccak ini dibandingkan pendahulunya, makalah ini berusaha menyajikan hasil analisis dan perbandingan beberapa aspek dari SHA-2 dan Keccak.**

***Index Terms*—SHA, Keccak, NIST**

I. PENDAHULUAN

Fungsi *hash* adalah suatu fungsi yang mengambil masukan, biasanya berupa string, yang panjangnya sembarang dan mengonversinya menjadi keluaran yang ukuran atau panjangnya tetap. Lebih spesifik lagi, fungsi hash adalah suatu prosedur deterministik yang mengambil suatu blok data dan mengembalikan suatu bit string yang ukurannya tetap yang merupakan nilai hash dari fungsi tersebut, sehingga jika ada perubahan pada data masukan, maka nilai keluarannya pun akan berbeda.

Fungsi hash dalam dunia kriptografi memiliki kegunaan yang sangat banyak terutama dalam bidang atau aplikasi keamanan informasi, misalnya untuk tanda tangan digital (*digital signature*), *Message Authentitcation Code* (MAC), pengamanan *password*, dan sebagainya. Sebagai fungsi hash biasa, fungsi-fungsi tersebut juga dapat digunakan sebagai penyimpanan indeks data, pendataan sidik jari, *checksum* dari file-file tertentu, dan sebagainya.

Salah satu fungsi hash yang sudah terstandardisasi dan banyak digunakan adalah SHA (*Secure Hash Algorithm*) yang dikeluarkan oleh NIST (*National Institute of Standard and Technology*).

Saat ni sudah terdapat tiga generasi dari SHA dimana dari generasi tersebut juga terdapat bebeapa varian lagi berdasarkan panjang bit yang dihasilkan. Ketiga varian itu adalah SHA-0 (atau yang biasa disebut SHA saja), SHA-1, dan SHA-2. Untuk SHA-3, saat ini sedang diadakan kompetisi untuk menentukan siapakah pemenang dari beberapa kandidat terkuat yang dapat dikatakan sebagai fungsi hash terbaik saat ini. Sejauh ini terdapat lima algoritma finalis, yaitu BLAKE, Grøstl, Keccak, JH, dan Skein.

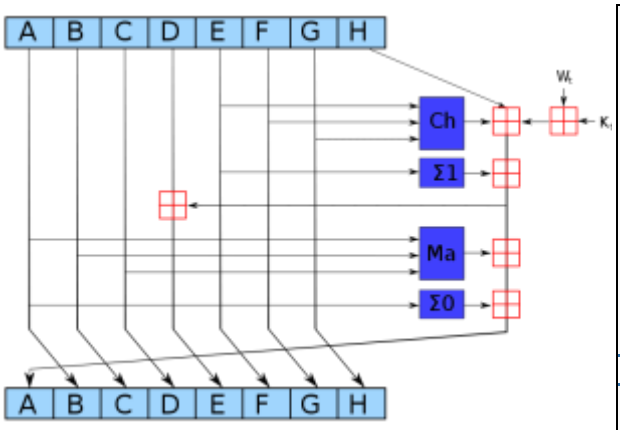
Algoritma hash tidak hanya dilihat dari segi keamanannya saja dari sudut pandang kriptografi, namun juga perlu dilihat/dinilai dari performansinya dan kecepatan memroses datanya. Oleh karena itu, makalah ini mencoba membandingkan keunggulan salah satu dari kelima finalis tersebut, yaitu Keccak, dengan algoritma yang telah biasa digunakan, yaitu SHA-2. Penjelasan lebih lanjut akan diberikan di bab-bab berikutnya.

II. PENEJALASAN ALGORITMA SHA-2

Algoritma SHA-2 merupakan pengembangan dari algoritma SHA-1 yang memuat banyak perubahan. Algoritma ini didesain oleh National Security Agency (NSA) of United States dan dipublikasikan pada tahun 2001 oleh NIST sebagai standar bagi pemrosesan informasi federal bagi Amerika Serikat atau yang biasa disebut *Federal Information Processing Standard* (FIPS).

Algoritma SHA-2 ini terdiri dari beberapa algoritma berdasarkan panjang bit yang digunakan/dihasilakn sebagai nilainya yaitu SHA-224, SHA-256, SHA-384,SHA-512.

Algoritma SHA-2 ini menggunakan beberapa operasi dasar dalam teknik kriptografi, yaitu operasi AND, OR, XOR, SHIFT (right), dan ROTATE (right). Berikut skema iterasi yang terdapat pada SHA-2, skema ini mengacu pada SHA-256.



Gambar 1Skema iterasi SHA-2

Pada skema diatas terdapat beberapa fungsi yang digunakan dalam proses iterasi, serperti yang terlihat pada blok biru, yaitu :

$$\mathrm{Ch}(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$$
$$\mathrm{Ma}(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

Berikut disertakan juga *pseudocode* dan langkah-langkah dari penggunaan algoritam SHA-256

```
Note 1: Semua variabel adalah unsigned 32 bit dan wrap modulo 232 ketika dihitung
Note 2: Semua konstan dalam pseudocode adalah big endian
```

```
Inisialisasi variabel
h[0..7] :=
   0x6a09e667, 0xbb67ae85, 0x3c6ef372, 0xa54ff53a, 0x510e527f, 0x9b05688c, 0x1f83d9ab, 0x5be0cd19
```

```
Inisialisas tabel round constants
k[0..63] :=
   0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
   0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
   0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
   0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
   0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
   0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
   0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
   0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2
```

```
Pre-processing:
append bit '1' ke pesan
append k bits '0', dimana k adalah the angka minimum >= 0 sedemikian sehingga pesan yang dihasilkan
    panjang (dalam bit) adalah kongruen dengan 448 (mod 512)
append panjang pesan (sebelum pre-processing), dalam bit, sebagai 64-bit big-endian integer
```

```
Proses pesan dalam suksesif 512-bit chunks:
pecah pesan menjadi 512-bit chunks
for each chunk
    break chunk into sixteen 32-bit big-endian words w[0..15]
```

```
Extend enambelas 32-bit words menjadi enampuluhempat 32-bit words:
    for i from 16 to 63
        s0 := (w[i-15] rightrotate 7) xor (w[i-15] rightrotate 18) xor (w[i-15] rightshift 3)
        s1 := (w[i-2] rightrotate 17) xor (w[i-2] rightrotate 19) xor (w[i-2] rightshift 10)
        w[i] := w[i-16] + s0 + w[i-7] + s1
```

```
Inisialisasi nilai hash untuk chunk ini:
    a := h0
    b := h1
    c := h2
    d := h3
    e := h4
    f := h5
```

```
    g := h6
    h := h7
```

```
Main loop:
    for i from 0 to 63
        s0 := (a rightrotate 2) xor (a
rightrotate 13) xor (a rightrotate 22)
        maj := (a and b) xor (a and c)
xor (b and c)
        t2 := s0 + maj
        s1 := (e rightrotate 6) xor (e
rightrotate 11) xor (e rightrotate 25)
        ch := (e and f) xor ((not e)
and g)
        t1 := h + s1 + ch + k[i] +
w[i]

        h := g
        g := f
        f := e
        e := d + t1
        d := c
        c := b
        b := a
        a := t1 + t2
```

```
Tambahkan hash dari chunk ke hasil:
    h0 := h0 + a
    h1 := h1 + b
    h2 := h2 + c
    h3 := h3 + d
    h4 := h4 + e
    h5 := h5 + f
    h6 := h6 + g
    h7 := h7 + h
```

```
Nilai hash final (big-endian):
digest = hash = h0 append h1 append h2
append h3 append h4 append h5 append
h6 append h7
```

III. PENJELASAN ALGORITMA KECCAK

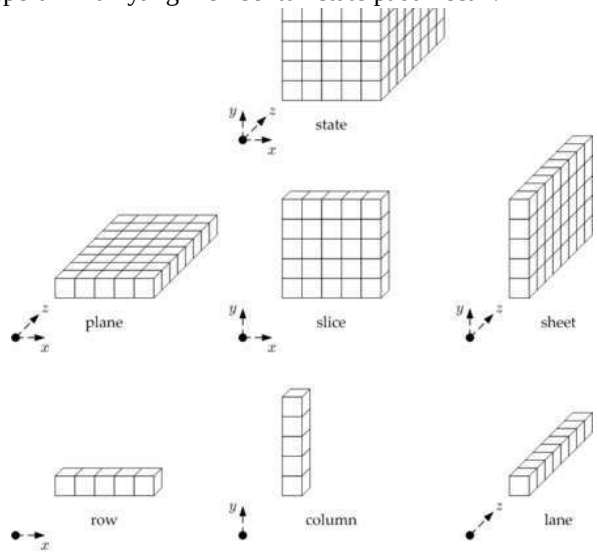
Seperti yang telah disebutkan pada bab sebelumnya, Keccak adalah salah satu algoritma finalis dari kompetisi yang diadakan oleh NIST untuk menjadi algoritma standar baru atau SHA-3.

Keccak didesain oleh Guido Bertoni, Joan Daemen, Michaël Peeters, dan Gilles Van Assche. Berdasarkan klaim dari pendesainnya, Keccak mampu melakukan 12,5 cpb (*cycle per byte*) pada komputer dengan CPU Intel Core 2. Hal ini menunjukkan bahwa algoritma ini sangat cepat bahkan lebih cepat dibandingkan dengan finalis lainnya.

Keccak mengadopsi *sponge structure* (struktur spons). Mengapa disebut spons? Karena pada struktur ini, input pada fungsi hash seperti "diserap" ke dalam state hash dan diberi rate tertentu, lalu ouput hash "disemprot" keluar dengan rate yang sama. Untuk menyerap *r* bit data, data di XORkan ke bit yang memimpin state, lalu permutasi blok diaplikasikan. Untuk "menyemprot", *r* bit pertama dari state dijadikan sebagai output, dan permutasi blok akan diaplikasikan lagi jika dibutuhkan.

Pusat dari semuanya adalah "capacity" *c* dari fungsi keccak ini, dengan *c*=1600-*r* yang tidak disentuh oleh input maupun output data. Untuk menghitung hash keccak, inisialisasi state menjadi 0, tambahkan *pad* pada input, dan pecah input menjadi potongan *r*-bit. Serap input kedalam state, XOR-kan, kemudian aplikasikan ke permutasi blok, seperti itulah algoritma nya.

Struktur pada Keccak adalah array tiga dimensi, jadi dalam melakukan perhitungan, tidak hanya didasarkan pada panjang dan lebar, namun juga *lane*, dan struktur seperti inilah yang membentuk state pada kecak.



Gambar 2 Struktur State pada Keccak

Permutasi blok pada Keccak dibagi menjadi empat tahap, yaitu tahap theta Θ, tahap rho ρ dan pi Π, tahap chi χ, dan tahap iota ι.

Berikut *pseudocode* dari algoritma keccak.

Fungsi dibawah ini digunakna untuk permutasi blok

```
KECCAK-f[b](A) {
  forall i in 0…nr-1
    A = Round[b](A, RC[i])
  return A
}
```

```
Round[b](A,RC) {
  θ step
  C[x] = A[x,0] xor A[x,1] xor A[x,2] xor
A[x,3] xor A[x,4],   forall x in 0…4
  D[x] = C[x-1] xor rot(C[x+1],1),
forall x in 0…4
  A[x,y] = A[x,y] xor D[x],
forall (x,y) in (0…4,0…4)

  ρ and π steps
  B[y,2*x+3*y] = rot(A[x,y], r[x,y]),
forall (x,y) in (0…4,0…4)

  χ step
  A[x,y] = B[x,y] xor ((not B[x+1,y]) and
B[x+2,y]), forall (x,y) in (0…4,0…4)

  ι step
  A[0,0] = A[0,0] xor RC

  return A
}
```

Fungsi dibawah ini adalah fungsi sponge

```
Keccak[r,c](M) {
  Initialization and padding
  S[x,y] = 0,
forall (x,y) in (0…4,0…4)
  P = M || 0x01 || 0x00 || … || 0x00
  P = P xor (0x00 || … || 0x00 || 0x80)

  Absorbing phase
  forall block Pi in P
    S[x,y] = S[x,y] xor Pi[x+5*y],
forall (x,y) such that x+5*y < r/w
    S = Keccak-f[r+c](S)

  Squeezing phase
  Z = empty string
  while output is requested
    Z = Z || S[x,y],
forall (x,y) such that x+5*y < r/w
    S = Keccak-f[r+c](S)

  return Z
}
```

RC[0]	0x0000000000000001	RC[12]	0x0000000080008 08B
RC[1]	0x0000000000008082	RC[13]	0x8000000000000 08B
RC[2]	0x8000000000008 08A	RC[14]	0x8000000000008089
RC[3]	0x8000000080008000	RC[15]	0x8000000000008003
RC[4]	0x0000000000008 08B	RC[16]	0x8000000000008002
RC[5]	0x0000000080000001	RC[17]	0x8000000000000080
RC[6]	0x8000000080008081	RC[18]	0x0000000000008 00A
RC[7]	0x8000000000008009	RC[19]	0x8000000080000 00A
RC[8]	0x0000000000000 08A	RC[20]	0x8000000080008081
RC[9]	0x0000000000000088	RC[21]	0x8000000000008080
RC[10]	0x0000000080008009	RC[22]	0x0000000080000001
RC[11]	0x0000000080000 00A	RC[23]	0x8000000080008008

Table 1 : Round Constants

	x = 3	x = 4	x = 0	x = 1	x = 2
y = 2	25	39	3	10	43
y = 1	55	20	36	44	6
y = 0	28	27	0	1	62
y = 4	56	14	18	2	61
y = 3	21	8	41	45	15

Table 2 : Rotations Offsets

Tabel Round Constants dan Tabel Rotation Offsets digunakan untuk melakukan perhitungan di dalam fungsi permutasi.

IV. IMPLEMENTASI

Dalam hal implementasi algoritma, pada makalah ini hanya digunakan program kecil yang hanya mampu untuk mengukur kecepatan kompresi dan melihat apakah perbedaan yang disebabkan oleh data sangat mempengaruhi nilai hash atau tidak.

Untuk implementasi SHA-256, program dibuat dengan menggunakan bahasa java. Program ini berfungsi untuk mengambil input dari file eksternal maupun dari teks inputan langsung, lalu mengeluarkan nilai hash beserta waktu yang dibutuhkan untuk melakukan proses hash tersebut, berikut antarmuka dari program tersebut

Gambar 3 Antarmuka program SHA-256

Untuk program yang digunakan untuk implementasi Keccak, penulis menggunakan program yang telah tersedia di internet, dan karena suatu hal tertentu tidak dapat ditampilkan di makalah ini antar mukanya.

Program ini memiliki fitur menampilkan waktu yang dibutuhkan untuk melakukan hash dan ukuran dari file input.

V. HASIL DAN ANALISIS

A. Hasil

Setelah dilakukan beberapa kali percobaan dengen menggunakan kedua program, didapatkan hasil sebagai berikut :

1. Kedua Algoritma tidak memiliki perbedaan waktu yang signifikan jika digunakan dengan file input berukuran kecil (kurang dari 1Mb)
2. Ketika dilakukan pengujian pada file dengan ukuran kisaran 10 sampai belasan Mb, terdapat perbedaan yang cukup mencolok dari kedua program dalam hal waktu yang dibutuhkan
   Untuk SHA-256 membutuhkan waktu sekitar : 6 detik
   Untuk Keccak hanya membutuhkan waktu sekitar : 2 detik
3. Penambahan ronde (melakuakan hash terus menerus beruturut-turut) menambah waktu yang dibutuhkan program untuk melakukan hash pada file yang sama, walaupun tidak signifikan.
4. Perubahan pada data mengakibatkan perubahan yang signifikan pada nilai hash pada kedua algoritma (SHA-256 dan Keccak)

B. Analisis

Secara keseluruhan, kinerja kedua algoritma ini cukup memuaskan, baik dari segi performansi maupun kecepatan. Dalam hal keamanan secara kriptografi, algortima SHA-256, sampai saat ini belum ada pengumuman resmi tentang adanya serangan yang berhasil secara normal untuk menjebol algoritma tersebut, baik dengan menggunakan *brute force* biasa, maupun dengan berusaha mencari celah dengan menggunakan collision. Kalaupun ada yang menyatakan berhasil, masih terbatas pada konsdisi tertentu, misalnya dengan mengurangi jumlah putaran.

Untuk Algoritma Keccak, karena ini algoritma baru, belum banyak informasi mengenai serangan pada algoritma ini, namun beberapa pihak ada yang mengklaim bahwa Keccak dapat dijebol dengan mengurangi jumlah bit yang diproses (standar dari kecak adalah 64 x 5 x 5). Dengan kata lain bukan dalam kondisi normal.

Dari segi performansi, kedua algoritma cukup memuaskan karena dengan mengubah data sedikit saja dapat mengubah nilai hash secara siginifikan, terutama pada algoritma Keccak, strukutur permutasinya yang cukup rumit mungkin cukup membantu dalam hal ini, begitu pula fungsi sponge nya. SHA-256 juga tidak mengecewakan.

Dalam hal performansi waktu, Algoritma Keccak terlihat lebih unggul dibandingkan dengan SHA-256. Hal ini dapat dilihat dari jumlah waktu yang dibutuhkan oleh SHA-256 untuk melukakan proses hash lebih lama dibandingkan waktu yang dibutuhkan oleh algoritma Keccak, walaupun hanya beberapa detik saja.

Untuk pemakaian kedepannya, sepertinya algoritma Keccak pantas digunakan sebagai pengganti atau pelengkap dari algoritma SHA-256 yang sekarang banyak dipakai. Walaupun serangan-serangan terhadap fungsi hash terus meningkat, sepertinya algoritma ini akan dapat bertahan dalam beberapa tahun ke depan.

Terakhir sebagai informasi pembanding, pada algoritma Keccak, memori yang digunakan cukup besar, karena untuk menyediakan konstanta saja (seperti Tabel Round Constant), dibutuhkan tipe data big integer (walalupun tabel round constant sebenarnya dapat di”potong”), dan juga prosesnya yang menggunakan array tiga dimensi akan lebih banyak menggunakan memori pula. Walaupun mungkin hal tersebut tidak terlalu signifikan, namun dapat dijadikan pertimbangan bagi orang-orang yang akan mengimplementasikan algoritma ini.

VI. KESIMPULAN

1. Algoritma Keccak memiliki performansi kecepatan yang lebih baik dibandingkan SHA-256.
2. Algoritma Keccak dan SHA-256 sampai saat ini dapat dikatakan cukup aman untuk fungsi hash secara kriptografi.
3. Algoritma Keccak dapat menggantikan algoritma SHA-2 yang ada sekarang.
4. Implementasi dari kedua algoritma cukup mudah dan tidak memakan banyak memori, namun pada Keccak sedikit lebih besar.

REFERENSI

[1] Munir, Rinaldi. *Bahan Kuliah IF3058*.2009. Program Studi Teknik Informatika. Institut Teknologi Bandung.

[2] http://en.wikipedia.org/wiki/Keccak

[3] http://plaintext.crypto.lo.gy/article/495/untwisted-a-cryptol-implementation-of-keccak-part-1

[4] http://en.wikipedia.org/wiki/SHA-2

[5] http://keccak.noekeon.org/Keccak-reference-3.0.pdf

[6] http://keccak.noekeon.org/specs_summary.html

[7] http://bryan.ravensight.org/tag/keccak/

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 9 Mei 2011

Abdurrisyad Fikri
13508017