Tinjauan Terhadap Bebersap Kandidat SHA-3

Gozali Harda Kumara (13502066)
Teknik Informatika
Sekolah Tinggi Elektro Informatika
Institut Teknologi Bandung

Abstraksi

Fungsi *hash* kriptografis adalah salah satu komponen terpenting dalam kriptografi. Untuk menetapkan standar yang baru, NIST mengadakan kompetisi yang terbuka untuk umum. Pada makalah ini, akan ditinjau kelengkapan argumen dan pembuktian beberapa peserta dalam kompetisi tersebut. Selain itu, pengujian kecepatan juga dilakukan atas fungsi *hash* yang ditinjau dalam berbagai mode dan panjang pesan.

1 Pendahuluan

Fungsi *hash* kriptografis adalah salah satu komponen utama di kriptografi, dan di tahun 1993, NIST menstandarisasi SHA-0 sebagai standar fungsi *hash*. Dua tahun kemudian, SHA-0 diperbaiki dan digantikan SHA-1 yang bahkan masih dianggap aman ketika NIST memperkenalkan keluarga SHA-2 (SHA-256 dan SHA-512). Pada tahun terakhir kriptanalisis terhadap beberapa fungsi *hash* telah menemukan kelemahan yang serius. Serangan kolisi telah ditemukan pada MDA-5 dan SHA-1. Sampai sekarang, kelemahan SHA-256 dan SHA-512 masih belum ditemukan, namun, karena kemiripan prinsip perancangan fungsi tersebut dengan SHA-1 dan MD-5, membuat NIST khawatir akan ketahanan mereka. Untuk mengatasi situasi ini, NIST mengumumkan kompetisi SHA-3 yang akan menggantikan standar SHA-2.

Sebanyak 64 peserta mengusulkan fungsi *hash* mereka dalam kompetisi ini, dan ketika makalah ini ditulis, program ini telah memasuki ronde pertama, dimana hanya ada 51 fungsi hash yang tersisa. Makalah ini akan meninjau sepuluh dari seluruh fungsi hash yang telah berhasil sampai pada ronde pertama, sepuluh fungsi *hash* tersebut adalah: BLAKE, CHI, ECHO, Grostl, Keccak, Lesamnta, Luffa, MD6, Shabal, dan Skein. Pemilihan fungsi-hash tersebut sepenuhnya adalah pilihan penulis karena batasan waktu dan ruang penulisan yang ada.

Makalah ini disusun sebagai berikut, pada Bagian 2 akan dibahas properti-properti yang diperlukan oleh sebuah fungsi *hash* kriptografis. Bagian 3 akan meninjau pembuktian yang dilakukan oleh peserta atas fungsi *hash* yang dilakukannya. Perlu ditekankan, bahwa peninjauan tidaklah dilakukan atas kebenaran suatu pembuktian, melainkan hanya apakah pengusul fungsi hash menawarkan pembuktian atas fungsi *hash*-nya.

Kemudian pada Bagian 4 akan dilakukan pengujian implementasi atas fungsi-fungsi *hash* tersebut dan akan dilakukan pengukuran kecepatan. Dan pada akhirnya, Bagian 5 akan memberikan kesimpulan atas peninjauan yang dilakukan.

2 Properti Fungsi Hash

Untuk sebuah fungsi *hash* kriptografis *h* yang melakukan hash terhadap pesan *M* dan menghasilkan *hash D=h(M)*, diperlukan properti-properti:

Ketahanan Terhadap Kolisi: Penyerang seharusnya tidak dapat menemukan dua pesan *M* dan *M'* yang berbeda sehingga *h(M)=h(M')*.

Ketahanan Terhadap *Preimage* Pertama: Penyerang yang mengetahui *D* seharusnya tidak dapat menemukan *preimage* M, sehingga h(M)=D.

Ketahanan Terhadap *Preimage* Kedua: Penyerang yang mengetahui sebuah pesan *M* seharusnya tidak dapat menemukan pesan *M'* yang berbeda sehingga *h(M)=h(M')*.

Ke-*pseudorandom*-an: Untuk fungsi *hash* yang diberi kunci, penyerang yang tidak mengetahui kuncinya seharusnya tidak bisa membedakan keluaran fungsi *hash* dari bilangan acak.

3 Tinjauan Pembuktian

Pada bagian ini, akan dibahas satu persatu keformalan dan kelengkapan argumen dan pembuktian atas setiap fungsi *hash* yang ditinjau.

3.1 BLAKE

BLAKE menggunakan mode iterasi HAIFA yang fungsi kompresinya dibangun atas fungsi ChaCha. BLAKE diusulkan oleh Aumasson, Henzen, Meier, dan Phan.

Ketahanan Terhadap Kolisi: Hal ini diklaim dengan mendeskripsikan bahwa BLAKE tahan terhadap *JOUX's technique*, *Schneier's technique*, dan *faster multicollision* (h. 39-40)

Ketahanan Terhadap *Preimage* pertama: Hal ini tidak dijelaskan.

Ketahanan Terhadap *Preimage* kedua: Hal ini dijelaskan dengan mengatakan bahwa karena berdasarkan HAIFA, maka serangan preimage kedua terhadap BLAKE paling tidak memiliki kompleksitas 2^n, kecuali serangan tersebut mengeksploitasi fungsi kompresi (h. 40)

Ke-*pseudorandom*-an: Hal ini diklaim dengan mengatakan bahwa BLAKE menurunkan ke-*pseudorandom*-an ChaCha (h. 38).

Pada setiap klaim tersebut, penulis tidak memberikan pembuktian yang formal.

3.2 CHI

CHI adalah fungsi *hash* yang diusulkan oleh Phil Hawkes dan Cameron McDonald dari Australia.

Ketahanan Terhadap Kolisi: Hal ini tidak dijelaskan.

Ketahanan Terhadap *Preimage* Pertama: Hal ini tidak dijelaskan.

Ketahanan Terhadap *Preimage* Kedua: Hal ini dijelaskan dengan mendeskripsikan ketahanan CHI terhadap serangan ekspansi (h. 90).

Ke-*pseudorandom*-an: Hal ini tidak dijelaskan.

Pengusul CHI menjelaskan secara detail alasan perancangan fungsi hash mereka, namun, pembuktian atas kriteria yang dibutuhkan sebuah fungsi *hash* tidak mencukupi.

3.3 ECHO

Echo dibangun diatas AES dan dengan paradigma Merkle-Damgard. Fungsi ini menggunakan sebanyak mungkin aspek dari AES dan ada akhirnya, perancang fungsi ini mengklaim bahwa fungsi ini menurunkan keamanan dari AES. ECHO juga mengadoptasi fitur dari model HAIFA dan menggunakan strategi pipa ganda.

Ketahanan Terhadap Kolisi dan *Preimage*: Hal ini diklaim dengan mengatakan bahwa dengan konstruksi Merkle-Damgard, ECHO akan memiliki ketahanan terhadap kolisi dan preimage asalkan fungsi kompresi juga memiliki ketahanan-ketahanan tersebut (h.40)

Ketahanan Terhadap Kriptanalisis Diferensial: Para penulis membahas karakteristik diferensial, karakteristik mixed-key, dan truncated differential dari ECHO secara mendalam. Para penulis mengatakan (dengan pembuktian) batas atas dari peluang yang diharapkan atas keberhasilan serangan-serangan ini. Sebagai contoh, kemungkinan karakteristik terbaik dari empat atau lebih operasi di ECHO, dirata-ratakan dengan *salt*, mempunyai batas atas 2^{-750}. Makalah juga membahas ketahanan ECHO atas berbagai serangan yang telah dilakukan terhadap AES, hal ini termasuk struktural, algebraik, *related-key*, dan *known-key*.

Secara umum, para penulis menjelaskan keamanan dari ECHO secara cukup formal.

3.4 Grostl

Grostl adalah fungsi *hash* yang diusulkan oleh tim dari beberapa universitas di Denmark.

Ketahanan Terhadap Kolisi: Hal ini dibuktikan secara formal atas ketahanan kolisi dari fungsi kompresi dan fungsi hash keseluruhan dari Grostl (h. 17-18).

Ketahanan Terhadap *Preimage* Pertama: Hal ini tidak dipaparkan.

Ketahanan Terhadap *Preimage* Kedua: Hal ini dijelaskan dengan mengatakan bahwa karena Grostl menggunakan konstruksi Merkle-Damgard, serangan

preimage kedua yang diketahui terhadap Grostl mempunyai kompleksitas 2^n.

Ke-*pseudorandom*-an: Hal ini tidak dipaparkan.

Ketahanan Terhadap Kriptanalisis Diferensial Dan Linear: Para penulis menawarkan pembuktian terhadap ketahanan dari differential kriptanalisis dengan menunjukkan bahwa serangan diferensial akan lebih kompleks dari *birthday-attack* sederhana (h. 15).

Penulis Grostl memberikan sebagian pembuktian secara formal, namun dalam beberapa bagian (*preimage* kedua), penjelasan hanya berupa deskripsi.

3.5 Keccak

Keccak adalah fungsi *hash* yang diusulkan Bertoni, Daemen, Peeters, dan Van Assche.

Ketahanan Terhadap Kolisi: Hal ini dibuktikan dengan menjelaskan ketahanan fungsi kompresi dari Keccak secara formal (h.28).

Ketahanan Terhadap *Preimage* Pertama dan Kedua: Hal ini dijelaskan dengan mendeskripsikan ketahanan Keccak terhadap berbagai metode (h. 31).

Ke-*pseudorandom*-an: Hal ini tidak dijelaskan secara eksplisit.

Secara umum, pengusul Keccak memberikan argumen, deksripsi, dan pembuktian yang cukup formal untuk menjelaskan fungsi hash mereka.

3.6 Lesamnta

Lesamnta adalah fungsi *hash* yang diusulkan oleh Hirose, Kuwakado, dan Yoshida dari Jepang.

Ketahanan Kolisi: Hal ini dibuktikan dengan asumsi bahwa fungsi kompresi h adalah cipher ideal yang tahan kolisi. Pembuktian dilakukan atas ketahanan kolisi mode MMO yang membutuhkan penyerang untuk paling tidak membutuhkan $2^{n/2}$ untuk menemukan kolisi (h. 60-61).

Ketahanan Terhadap *Preimage* Pertama dan Kedua: Hal ini dibuktikan dengan asumsi bahwa fungsi kompresi *h* adalah cipher ideal, maka keuntungan penyerang sangatlah kecil dan sama dengan $\frac{1}{2^n - q}$.

Ke-*pseudorandom*-an: Pembuktian atas hal ini dilakukan dengan asumsi bahwa *E* dan *L* adalah permutasi *pseudorandom* yang independen. Kemudian, ditunjukan bahwa Lesamnta adalah fungsi pseudorandom.

Secara umum, para pengusul Lesamnta memberikan argumen dan pembuktian mereka secara detail dan formal.

3.7 Luffa

Luffa adalah fungsi *hash* yang diusulkan Dai dan Watanabe dari Jepang. Luffa adalah fungsi *hash* yang didasarkan atas fungsi sponge yang keamanannya didasarkan atas keacakan permutasi.

Ketahanan Terhadap Kolisi: Hal ini dijelaskan dengan deskripsi singkat (h. 21).

Ketahanan Terhadap *Preimage* Pertama dan Kedua: Hal ini dijelaskan dengan deskripsi singkat (h.21).

Ke-pseudorandom-an: Hal ini tidak dibuktikan.

Ketahanan Terhadap Kriptanalisis Diferensial: Hal ini dibuktikan dengan pembuktian yang formal (h. 10)

Pembuktian formal yang dilakukan oleh pengusul Luffa hanya dilakukan atas kriptanalisis diferensial, sedangkan properti lainnya hanya disebutkan argumen singkat.

3.8 MD6

MD6 adalah fungsi hash kriptografis yang diusulkan oleh tim MIT dan dikepalai oleh Ron Rivest. Berikut hasil tinjauan yang dilakukan terhadap pembuktian atas MD6:

Ke-*pseudorandom*-an: Pembuktian terhadap ke-pseudorandom-an dari MD6 dilakukan dengan menggunakan *framework* sistem bilangan acak Maurer(h.50-h.64)

Ketahanan Atas Kolisi dan *Preimage*: Pembuktian terhadap ketahanan atas kolisi dari MD6 dilakukan secara langsung dan dengan kontradiksi dengan membuktikan bahwa jika algoritma pencari kolisi ditemukan pada keseluruhan fungsi hash, maka algoritma pencari kolisi juga akan ditemukan atas fungsi kompresi. Dengan demikian, jika diasumsikan

bahwa tidak ditemukan algoritma pencari kolisi pada fungsi *hash*, maka algoritma pencari kolisi untuk fungsi *hash* keseluruhan tidak akan ditemukan. hal (h. 34-41 kolisi) (h 42 -45 *preimage*) (h. 45-47 *preimage* kedua).

Secara umum, pengusul MD6 memberikan pembuktian yang formal atas fungsi *hash* mereka.

3.9 Shabal

Shabal adalah fungsi *hash* kriptografis yang diusulkan oleh SAPHIR, sebuah proyek riset yang didanai oleh pemerintah perancis. Berikut hasil tinjauan yang dilakukan terhadap pembuktian atas Shabal:

Ke-*pseudorandom*-an: Pembuktian terhadap ke-pseudorandom-an dari shabal dilakukan dengan membuktikan bahwa shabal adalah simulator atas random-oracle (h.51-h.65)

Ketahanan Terhadap Kolisi: Pembuktian terhadap ketahanan atas kolisi dari Shabal dilakukan dengan membuktikan bahwa jika ada pencari kolisi A, maka kompleksitas waktu yang dibutuhkan a sama dengan besar ruang pencarian. (h.65- 74)

Ketahanan Terhadap *Preimage*: Pembuktian terhadap ketahanan atas kolisi dari Shabal dilakukan dengan membuktikan bahwa jika ada pencari preimage *a*, maka kompleksitas waktu yang dibutuhkan *a* sama dengan besar ruang pencarian. (h.74-85)

Ketahanan Terhadap *Preimage* Kedua: Pembuktian terhadap ketahanan atas kolisi dari Shabal dilakukan dengan membuktikan bahwa jika ada pencari *preimage a*, maka kompleksitas waktu yang dibutuhkan *a* sama dengan besar ruang pencarian. (h.85-96)

Ketahanan Terhadap Kriptanalisis Linear dan Diferensial: Pembuktian terhadap ketahanan atas kriptanalisis diferensial dari Shabal dibuktikan dengan membuktikan ketahanan atas *truncated differential trails*, *symmetric dferential trails* atau *diferential trails* tanpa perbedaan di register A (h.123-h.126).

Secara keseluruhan, pengusul Shabal memberikan argumen dan pembuktian yang lengkap.

3.10 Skein

Skein adalah fungsi hash yang diusulkan oleh Ferguson, Schneier, dkk. Skein menggunakan Threefish sebagai blok cipher utamanya.

Ke-*pseudorandom*-an: Hal ini dibuktikan dengan asumsi bahwa Threefish adalah blok cipher ideal, maka Skein adalah *pseudorandom* (h.31).

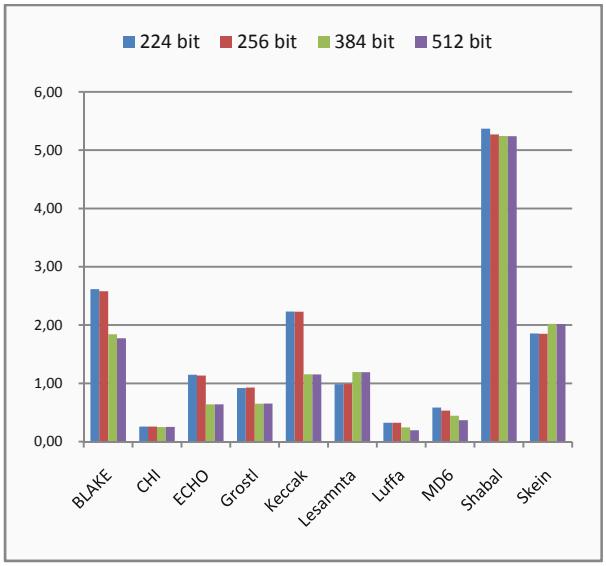
Ketahanan Terhadap Kolisi: Hal ini dibuktikan dengan bahwa jika fungsi kompresi tahan terhadap kolisi, maka demikian juga Skein (h.30).

Ketahanan Terhadap *Preimage* Pertama dan Kedua: Hal ini dibuktikan dengan bahwa jika fungsi kompresi tahan terhadap serangan *preimage*, maka demikian juga Skein (h.28-29).

Secara keseluruhan, pengusul Skein memberikan argumen dan pembuktian dengan formal dan mendetail.

4 Pengujian Kecepatan

Pengujian kecepatan dilakukan dengan membuat sebuah API terhadap seluruh fungsi *hash* yang diuji. API dan fungsi *hash* tersebut kemudian dikompilasi dengan *gcc* pada *platform* OpenSuse 11 dengan prosesor Intel Centrino 1.6GHz.



Gambar 1 Hasil Pengujian Kecepatan (MB/detik)

Nilai rata-rata dari pengeksekusian fungsi *hash* tersebut terhadap file dengan berbagai ukuran kemudian diukur. Waktu eksekusi dihitung dengan pencatatan waktu *user* dengan aplikasi *time* yang disediakan linux. Hasil pengujian untuk fungsi *hash* dengan panjang hasil 224 bit, 256 bit, 384 bit, dan 512 bit ditunjukan pada Gambar 1 dan Tabel 1.

Fungsi Hash	224 bit	256 bit	384 bit	512 bit
BLAKE	2,61600	2,58000	1,84100	1,77300
CHI	0,25700	0,25728	0,25115	0,25168
ECHO	1,14819	1,13435	0,63960	0,63971
Grostl	0,92002	0,92727	0,65021	0,65249
Keccak	2,23261	2,22888	1,15444	1,15232
Lesamnta	0,98553	0,98918	1,19323	1,19252
Luffa	0,32255	0,32320	0,24399	0,19402
MD6	0,58469	0,53342	0,44593	0,36760
Shabal	5,36913	5,27009	5,24246	5,23903
Skein	1,85432	1,84960	2,01956	2,01804

Tabel 1 Hasil Pengujian Kecepatan (MB/detik)

Dari tabel dan gambar tersebut dapat dilihat bahwa untuk semua mode, Shabal merupakan fungsi *hash* tercepat. BLAKE merupakan fungsi *hash* tercepat kedua untuk mode 224 dan 256 bit, sedangkan untuk mode 384 dan 512 bit, yang menempati posisi kedua adalah Skein.

5 Kesimpulan

Berdasarkan kedetailan dan keformalan argumen serta pembuktian yang diberikan pengusul setiap fungsi hash yang dibahas pada makalah ini, ECHO, Keccak, Shabal, MD6, dan Skein adalah fungsi *hash* yang diusulkan dengan dokumentasi atas pembuktian yang lengkap.

Sedangkan berdasarkan pengujian kecepatan yang dilakukan, Shabal merupakan algoritma yang tercepat untuk setiap mode, diikuti oleh BLAKE dan Skein.

6 Referensi

Halaman kompetisi SHA-3 NIST. *http://csrc.nist.gov /groups/ST/hash/sha-3/*. Tanggal Akses: 17 Mei 2009.

The SHA-3 Zoo. *http://ehash.iaik.tugraz.at/wiki /The_SHA-3_Zoo/*. Tanggal Akses: 17 Mei 2009.