Analisis dan Pengembangan Merkle-Damgård *Structure*

Boyke Ariesanda
NIM 135 03 036

Program Studi Teknik Informatika STEI, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
e-mail: if13036@students.if.itb.ac.id

Abstrak

Salah satu alternatif dalam aplikasi konsep tandatangan digital (*digital signature*) adalah implementasi yang menggunakan fungsi *hash*. Secara umum, fungsi *hash* adalah jenis fungsi yang mampu menerima *string* dengan panjang sembarang dan mengkonversinya ke dalam *string* dengan panjang atau ukuran tertentu, umumnya berukuran kecil, yang telah ditetapkan (berapapun panjang *string* masukan).

Fungsi atau metode *hashing* yang paling populer dalam aplikasi tandatangan digital ialah Merkle-Damgård *Structure*. Metode ini digunakan pada sebagian besar algoritma tandatangan digital yang ada saat ini, seperti MD5 dan SHA. Makalah ini memaparkan Merkle-Damgård *Structure* secara komprehensif, kelebihan dan kekurangannya, serta kemungkinan pengembangannya untuk dua tujuan yang berbeda, yakni:

1. Menyempurnakan tingkat keamanan algoritma.
2. Melakukan optimasi algoritma, sehingga diperoleh skema yang *compact* namun tetap aman dan dapat diaplikasikan pada berbagai perangkat digital.

Kata Kunci: Tandatangan digital, fungsi *hash*, Merkle-Damgård *Structure*, kriptanalisis.

1. Pendahuluan

Pada Abad Informasi saat ini, peran tandatangan digital (*digital signature*) dirasakan semakin penting. Tandatangan digital mampu menyediakan jaminan keamanan bagi dokumen atau data yang tersedia melalui media elektronik. Hal itu menyebabkan pengembangan tandatangan digital dan komponen-komponennya (*building blocks*) perlu dilakukan secara berkelanjutan.

Konsep tandatangan digital merupakan anggota keluarga besar ilmu kriptografi. Oleh karena itu, beberapa teknik kriptografi telah digunakan untuk mengimplementasikan konsep ini. Teknik-teknik kriptografi tersebut ialah:

1. Tandatangan digital menggunakan teknik enkripsi, baik menggunakan teknik kriptografi kunci simetri maupun kriptografi kunci publik.
2. Tandatangan digital menggunakan fungsi *hash.*
3. Kombinasi keduanya.

Makalah ini akan membahas jenis yang kedua. Oleh karena itu, seterusnya dalam makalah ini, kecuali pada bagian yang membahas konsep-konsep dasar, terminologi ‘tandatangan digital’ akan selalu mengacu kepada konsep serta teknik tandatangan digital yang menggunakan fungsi *hash*.

Komponen paling penting dalam tandatangan digital ialah fungsi *hash*. Fungsi *hash* memetakan dokumen atau data ke dalam rangkaian bit yang unik dan terbatas. Dengan kata lain, fungsi *hash* mampu menerima masukan berupa *string* dengan panjang berapapun dan menghasilkan *string* yang panjangnya tetap.

Kerangka atau konstruksi fungsi *hash* yang sering dipakai dalam aplikasi tandatangan digital ialah Merkle-Damgård *Structure*, yang dinamai menurut penemunya, Ralph Merkle dan Ivan Damgård.

2. Konsep-Konsep Dasar

2.1. Layanan Kriptografi

Sampai saat ini, kriptografi sebagai sebuah teknologi telah berhasil menyediakan beberapa layanan atau fasilitas terhadap penggunanya, terkait dengan aspek keamanan data yang disimpan atau ditransmisikan secara elektronik atau digital. Berikut ialah definisi dari keempat layanan kriptografi tersebut [1].

Definisi 1

Kerahasiaan (*confidentiality*) adalah layanan kriptografi yang dapat menjaga isi dari suatu dokumen sehingga orang-orang yang tidak berhak tidak mampu membaca dan memahami isi dokumen tersebut.

Definisi 2

Integritas Data (*data integrity / data athentication*) adalah layanan kriptografi yang berguna untuk menjaga keaslian isi dokumen serta mampu mendeteksi manipulasi terhadap data oleh pihak yang tidak berhak.

Definisi 3

Otentikasi (*user / entity / data origin authentication*) ialah layanan kriptografi yang mampu menyediakan identifikasi pemilik atau pengirim dokumen sehingga mampu mendeteksi dokumen 'asli' yang dikirim oleh pihak yang berpura-pura sebagai pihak yang berhak.

Definisi 4

Nirpenyangkalan (*non-repudiation*) ialah layanan kriptografi yang mampu mencegah seseorang menyangkal pengiriman atau kepemilikan terhadap suatu dokumen dengan cara memberikan karakteristik unik pada dokumen tersebut yang mengidentifikasi pemilik atau pengirim dokumen serta tidak dapat disangkal.

2.2. Tandatangan Digital

Seperti telah kita ketahui bersama, tandatangan merupakan perangkat yang sering kita gunakan sehari-hari sebagai identifikasi diri yang unik, terutama dalam hal-hal yang terkait dengan dokumen tertulis. Karena keunikannya itu, tandatangan memiliki fungsi utama sebagai bukti yang otentik terhadap keterkaitan pemilik tandatangan dengan dokumen yang bertandatangan tersebut, baik dalam bentuk kepemilikian, persetujuan, maupun pengetahuan (*document acknowledgement*). Agar mampu memenuhi fungsinya tersebut, maka tandatangan seharusnya memiliki sifat-sifat berikut [1].

1. Tidak dapat dilupakan
2. Tidak dapat dipindahkan
3. Tidak dapat digunakan ulang
4. Tidak dapat disangkal
5. Dokumen bertandatangan tidak dapat diubah

Definisi 5

Tandatangan (*signature*) adalah tulisan atau rangkaian aksara / tanda yang bersifat unik terhadap setiap orang sehingga dapat digunakan untuk melakukan identifikasi terhadap pemilik tandatangan dan melakukan otentikasi terhadap dokumen yang ditandatangani.

Bila kita bandingkan konsep tandatangan konvensional di atas dengan layanan kriptografi, terlihat bahwa tandatangan konvensional secara teoretis mampu mewujudkan integritas data, otentikasi, dan nirpenyangkalan sekaligus. Tandatangan mampu menjaga integritas data melalui konvensi yang menyatakan bahwa sebuah dokumen yang telah ditandatangani tidak dapat dimanipulasi lebih lanjut sehingga meninggalkan cacat pada dokumen seperti coretan, tulisan yang ditindih atau diganti, robek, dan sebagainya (sifat ke-5). Bila cacat dokumen terdeteksi, tandatangan berikut dokumen tersebut dianggap tidak sah.

Tandatangan juga mampu berperan sebagai perangkat otentikasi pihak yang terkait dengan dokumen tersebut. Hal ini disebabkan oleh sifat tandatangan yang 'melekat' pada setiap orang secara personal dan unik. Terkait dengan hal itu, tandatangan juga mampu mewujudkan nirpenyangkalan (sifat ke-1, ke-4). Namun, saat ini fungsi otentikasi dan nirpenyangkalan dari tandatangan konvensional semakin tidak dapat diandalkan mengingat pemalsuan tandatangan, beserta teknik lain yang tergolong rekayasa dokumen (*document forgery*), semakin umum dijumpai dan semakin mudah dilakukan.

Beralih ke Abad Informasi saat ini, kertas sebagai media dokumen sedikit demi sedikit mulai digantikan oleh media elektronik serta dokumen yang disimpan di dalamnya (*e-document*). Namun demikian, peran tandatangan tetap tidak berubah. Sesuai dengan medianya, tandatangan bertransformasi menjadi tandatangan digital. Ia bukan lagi berupa sebentuk tulisan unik yang yang dibubuhkan pada sebuah dokumen. Tandatangan digital merupakan selarik kode dalam *string* yang

berfungsi seperti tandatangan biasa bagi dokumen digital tersebut. Kode ini dihasilkan melalui teknik kriptografi yang diterapkan kepada dokumen tersebut. Jadi, sedikit berbeda dengan tandatangan kovensional, nilai suatu tandatangan digital juga bergantung pada isi dokumen yang ditandatangani, selain bergantung kepada pemilik tandatangan.

Definisi 6

Tandatangan digital (*digital signature*) adalah rangkaian kode yang dihasilkan melalui pemrosesan kriptografis terhadap dokumen digital sedemikian hingga ia dapat digunakan untuk melakukan identifikasi terhadap pemilik tandatangan dan melakukan otentikasi terhadap dokumen yang ditandatangani.

Sampai saat ini, teknik kriptografis yang dimanfaatkan untuk aplikasi tandatangan digital terdiri dari dua macam, yaitu teknik enkripsi dan teknik *hashing* [1]. Teknik yang pertama dijalankan dengan mengenkripsi dokumen, baik menggunakan algoritma kunci simetri maupun menggunakan algoritma kunci publik / nirsimetri. Melalui teknik ini, aplikasi tandatangan digital mampu menyediakan keempat aspek keamanan yang disediakan kriptografi (kerahasiaan, integritas data, otentikasi, dan nirpenyangkalan).

Teknik yang kedua, teknik *hashing*, diterapkan dengan cara menjalankan fungsi *hash* pada dokumen sehingga dihasilkan sebuah kode ringkas (*hash value*) dari dokumen tersebut. Kode *hash* tersebut kemudian 'ditempelkan' (*appended*) ke dokumen. Dengan teknik ini, dokumen tetap dapat dibaca siapapun (tidak ada kerahasiaan). Di sisi lain, integritas data terjamin karena jika dokumen berubah sekecil apapun, nilai *hash* pasti akan berubah sehingga manipulasi akan terdeteksi.

Teknik *hashing* di atas masih kurang lengkap sebagai aplikasi tandatangan digital karena belum mampu menyediakan fasilitas otentikasi dan nirpenyangkalan. Hal itu disebabkan oleh kenyataan bahwa fungsi *hash* hanya menerima masukan berupa dokumen, tanpa ada masukan lain yang berfungsi sebagai identifikasi terhadap pihak penandatangan. Solusi untuk masalah ini dapat dilakukan dengan dua macam cara yakni menggunakan teknik MAC (*Message Authentication Code*) dan teknik *hash-and-sign*.

Teknik MAC dijalankan dengan menggunakan algoritma MAC yang merupakan varian dari algoritma / fungsi *hash* standar. Alih-alih hanya menerima sebuah masukan yakni dokumen yang akan ditandatangani, algoritma MAC juga menerima sebuah masukan lain yang merupakan kunci rahasia. Dengan adanya kunci rahasia ini, sembarang orang, kecuali pengirim dan penerima dokumen yang sama-sama mengetahui kunci rahasia, tidak dapat memanipulasi dokumen sambil tetap mempertahankan kode *hash* yang valid. Dengan demikian otentikasi serta nirpenyangkalan dapat tercapai.

Teknik yang kedua, *hash-and-sign*, merupakan kombinasi antara teknik *hashing* biasa dengan teknik enkripsi. Suatu fungsi *hash* digunakan untuk menghitung kode *hash* dari dokumen (tahap *hash*). Sebelum ditempelkan pada dokumen, nilai *hash* tersebut dienkripsi menggunakan kunci (tahap *sign*). Teknik ini dapat dilakukan dengan memanfaatkan algoritma kunci simetri ataupun algoritma kunci nirsimetri, meskipun terdapat perbedaan pada prosedur penandatanganan dan otentikasi dalam menggunakan kedua macam algoritma tersebut. Apapun jenis algoritma enkripsi yang dipakai, jika prosedur yang sesuai dijalankan dengan benar, maka layanan otentikasi dan nirpenyangkalan dapat tercapai karena adanya kunci enkripsi dan/atau dekripsi yang berperan sebagai 'identitas' pihak-pihak yang terkait dengan dokumen.

2.3. Fungsi *Hash*

Secara umum, fungsi *hash* merupakan jenis fungsi yang mampu menghasilkan kode ringkas dari sebuah *string*. Jika dari kode ringkas tersebut dapat diperoleh kembali *string* awal melalui *fungsi rekonstruksi*, maka ia disebut fungsi *hash* dua arah. Fungsi-fungsi atau algoritma kompresi data seperti Kode Huffman dapat digolongkan ke dalam jenis ini. Sebaliknya, jika *string* awal tidak dapat direkonstruksi dari kode ringkasnya, maka ia disebut fungsi *hash* satu arah.

Dalam kriptografi, khususnya aplikasi tandatangan digital, fungsi *hash* yang digunakan adalah yang bersifat satu arah. Secara lebih spesifik, kode ringkas yang dihasilkan memiliki panjang yang tetap dan secara umum lebih pendek dari panjang *string* awal, berapapun ukuran *string* masukan. Dengan adanya spesifikasi ini, maka fungsi *hash* akan bersifat

many-to-one karena domain fungsi berukuran jauh lebih besar daripada daerah hasilnya.

Definisi 7

Fungsi *hash* (kriptografi) adalah fungsi yang menerima masukan berupa rangkaian bit yang panjangnya sembarang dan menghasilkan rangkaian bit yang panjangnya tetap, atau dapat diekspresikan sebagai fungsi H, dimana

$H(\{0,1\}^*) \rightarrow \{0,1\}^k$, k konstan

Agar dapat digunakan dalam aplikasi tandatangan digital, suatu fungsi *hash* harus memiliki properti sebagai berikut [1][5].

1. Fungsi *hash* harus mampu bekerja pada blok data berukuran sembarang.
2. Fungsi *hash* harus menghasilkan keluaran yang panjangnya tetap (*fixed-length output*).
3. Nilai fungsi *hash* $H(x)$ untuk sembarang masukan x mudah dikomputasi (kompleksitas fungsi tidak terlalu tinggi).
4. (***Preimage attack***) Sebaliknya, tidak mungkin melakukan rekonstruksi nilai x dari nilai $H(x)$. Properti ini menegaskan bahwa fungsi *hash* dalam tandatangan digital harus bersifat satu arah.
5. (***Second preimage attack / weak collision attack***) Untuk sembarang x, tidak mungkin mencari $y \neq x$ sedemikian hingga $H(y) = H(x)$.
6. (***Single collision attack***) Secara komputasi, tidak mungkin mencari pasangan x dan y sedemikian hingga $H(x) = H(y)$, meski secara teoretis pasangan seperti itu pasti ada.

Lebih jauh, properti fungsi *hash*, khususnya tiga yang terakhir, mengalami generalisasi sebagai berikut [5].

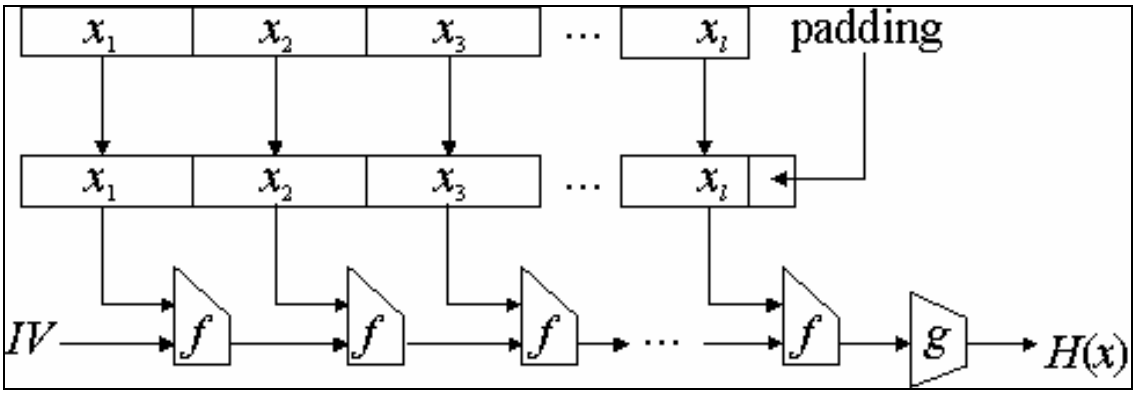
7. (***K-way preimage attack***) Jika diketahui sembarang nilai *hash* $H(x)$, tidak mungkin mencari $x_1, x_2, x_3, \ldots, x_k$, sedemikian hingga $H(x_1) = H(x_2) = H(x_3) = \ldots = H(x_k)$.
8. (***K-way second preimage attack***) Jika diketahui sembarang nilai y, tidak mungkin mencari $x_1, x_2, x_3, \ldots, x_k$, sedemikian hingga $H(y) = H(x_1) = H(x_2) = H(x_3) = \ldots = H(x_k)$.
9. (***K-way / multi collision attack***) Tidak mungkin mencari nilai-nilai $x_1, x_2, x_3, \ldots, x_k$, sedemikian hingga $H(x_1) = H(x_2) = H(x_3) = \ldots = H(x_k)$

Pada kenyataannya, syarat-syarat di atas merupakan celah yang selalu dicoba untuk dimanfaatkan oleh pihak-pihak yang ingin menyerang atau memecahkan suatu fungsi *hash*. Terlihat bahwa serangan pada fungsi *hash* sebagian besar mengeksploitasi sifat alamiahnya yakni *many-to-one mapping* yang berimplikasi pada kepastian terjadinya *collision*. Adapun motivasi serangan kepada fungsi *hash*, dalam konteks ini khususnya serangan kepada tandatangan digital, tidak jauh berbeda dengan motivasi dalam *document forgery* konvensional yaitu memanipulasi dokumen dan/atau tandatangan sedemikian hingga dokumen tersebut tetap tampak 'asli' (valid) namun dengan isi yang sudah berubah sesuai keinginan pelaku. Di sisi lain, serangan terhadap fungsi *hash*, atau secara umum terhadap sembarang teknik kriptografi juga dilakukan dengan motivasi yang baik, yakni mencari kelemahan suatu teknik kriptografi sehingga teknik itu bisa disempurnakan ke tingkat keamanan yang lebih baik. Apapun motivasinya, serangan semacam ini disebut sebagai kriptanalisis.

Definisi 8

Kriptanalisis adalah aktivitas menganalisis suatu teknik kriptografi untuk mencari kelemahan teknik tersebut sehingga dapat dimanfaatkan untuk tujuan tertentu, baik ataupun buruk.

Selain properti di atas, suatu fungsi *hash* juga harus dirancang sedemikian rupa sehingga tidak mungkin menentukan hubungan antara sembarang partisi pada masukan dan sembarang partisi pada keluaran, atau sebaliknya. Secara praktis, hal ini sama dengan kemampuan untuk memodifikasi dokumen diikuti dengan modifikasi manual pada nilai *hash*-nya, namun tetap mempertahankan 'validitas' dokumen. Kriteria ini dapat disebut sebagai daya tahan terhadap ***local preimage attack***. Fungsi *hash*, dan teknik kriptografis lain secara umum, memenuhi kriteria ini dengan mewujudkan apa yang disebut sebagai ***avalanche effect***, yaitu perubahan minimum pada masukan mampu menghasilkan perubahan drastis pada keluaran, analog dengan terjadinya *avalanche* (longsor salju) dimana satu gerakan kecil di puncak dapat menimbulkan longsor yang semakin membesar dan merusak di lembah. Efek ini sejalan dengan konsep *diffusion* dalam Teori Informasi oleh Shannon yang menjadi landasan umum bagi kriptografi.



Gambar 1

3. Merkle-Damgård *Structure*

Telah dipaparkan sebelumnya, bahwa kegunaan utama suatu fungsi *hash* adalah untuk menghasilkan representasi yang ringkas dari suatu dokumen (*string*). Representasi ini harus memiliki panjang yang tetap, berapapun panjang *string* masukan. Bagian ini akan menjelaskan cara kerja Merkle-Damgård *Structure* sehingga mampu memenuhi *requirement* tersebut. Bagian ini juga akan memaparkan sifat-sifat metode ini, kelebihan, serta kekurangannya.

3.1. Cara Kerja [2]

Secara garis besar, Merkle-Damgård *Structure* bekerja dengan memecah *string* dengan panjang sembarang ke dalam blok-blok yang ukurannya telah ditentukan. Struktur ini kemudian bekerja secara iteratif, setiap iterasinya mengkombinasikan sebuah blok dengan hasil iterasi sebelumnya, menggunakan sebuah fungsi kompresi.

Merkle-Damgård *Structure* mensyaratkan blok-blok masukan memiliki panjang yang seragam. Namun, karena *string* masukan dapat memiliki panjang sembarang, ada kemungkinan bahwa blok terakhir memiliki panjang yang berbeda. Merkle-Damgård *Structure* memberi solusi dengan melakukan *padding* (penambahan) 0 (sebenarnya nilai apapun dapat digunakan sebagai *padding*, namun untuk menyederhanakan kerja algoritma biasanya digunakan nilai 0 yang memiliki representasi rangkaian bit 0) sebanyak yang diperlukan supaya panjang blok terakhir menjadi seragam. Solusi ini belum memadai karena *padding* nilai 0 dapat mengakibatkan dua buah *string* yang sebenarnya berbeda menjadi identik setelah proses *padding*.

Contoh:

Misalnya, telah ditetapkan bahwa suatu Merkle-Damgård *Structure* bekerja pada panjang blok sebesar tujuh karakter. Jika *string* masukan ialah “kriptografi”, ia akan dipecah menjadi

```
kriptog     rafi
```

Blok terakhir akan dikenakan *padding* menjadi

```
kriptog     rafi000
```

Namun, misalkan terdapat *string* kedua yaitu “kriptografi00” (“00” di akhir merupakan bagian asli dari *string* tersebut). Ia akan dipecah menjadi

```
kriptog     rafi00
```

Blok terakhir akan dikenakan *padding* menjadi

```
kriptog     rafi000
```

Terlihat bahwa dua buah *string* yang berbeda menjadi “sama”.

Untuk mengatasi masalah ini, sebuah nilai 1 (sebenarnya nilai apapun bisa dipakai sebagai ‘pembatas’) ditambahkan di akhir *string* sebelum *padding* nilai 0.

Contoh:

Pada contoh sebelumnya, *string* pertama akan dipecah dan dikenai *padding* menjadi

```
kriptog     rafi100
```

Sedangkan *string* kedua menjadi

```
kriptog     rafi001
```

Lebih jauh lagi, para pencipta Merkle-Damgård *Structure* mengusulkan teknik yang dinamakan Merkle-Damgård *Strengthening*, yakni melakukan *padding* nilai representasi panjang *string* asli setelah *padding* nilai 0.

Contoh:

Pada contoh sebelumnya, *string* pertama awalnya (“kriptografi”) memiliki panjang 11, sehingga hasil akhir *padding* menjadi

```
kriptog rafi100 0000011
```

Sedangkan *string* kedua yang aslinya ("kriptografi00") memiliki panjang 13, akan menjadi

```
kriptog rafi001 0000013
```

Namun demikian, algoritma *hash* yang mengimplementasikan Merkle-Damgård *Structure* biasanya menetapkan besarnya representasi panjang *string* awal. Hal ini berguna untuk melakukan sedikit efisiensi selain secara implisit membatasi ukuran *string* awal yang dapat diproses algoritma tersebut.

Contoh:

Misalkan sebuah algoritma *hash* memiliki spesifikasi ukuran representasi panjang *string* sebesar dua karakter. Jika *string* pada contoh sebelumnya diproses menggunakan algoritma ini, maka *string* pertama akan dipecah dan di-*padding* menjadi

```
kriptog     rafi111
```

Sedangkan *string* kedua akan menjadi

```
kriptog rafi001 0000013
```

Terlihat bahwa algoritma dapat 'menghemat' satu blok. Selain itu, secara implisit, spesifikasi tersebut juga mensyaratkan panjang maksimum *string* masukan ialah (2^{16} – 1) bit, dengan asumsi sebuah karakter memiliki representasi 8 bit.

Manfaat Merkle-Damgård *Strengthening* akan ditelaah lebih lanjut pada bagian-bagian selanjutnya.

Proses inisialisasi menghasilkan blok-blok *string* yang siap dikompresi secara iteratif. Dalam proses utama ini, diperlukan sebuah fungsi kompresi yang menerima masukan berupa sebuah blok *string* serta blok *string* hasil (antara) dari iterasi sebelumnya, dan menghasilkan sebuah blok dengan panjang tertentu, yang tidak perlu sama besarnya dengan panjang blok masukan. Khusus untuk iterasi pertama, Merkle-Damgård *Structure* memerlukan sebuah blok yang disebut *initialization vector* (IV). Ukuran IV inilah yang akan menentukan ukuran keluaran akhir dari algoritma *hash* yang memakai Merkle-Damgård *Structure* ini.

Untuk fungsi kompresi dalam setiap iterasi, biasanya digunakan fungsi yang didasarkan pada algoritma *block cipher*. Hal ini dimaksudkan agar kompleksitas Merkle-Damgård *Structure* secara keseluruhan dapat dijaga tetap sederhana (fungsi kompresi dapat dibuat cukup sederhana, hanya perlu 'mencampur' sebuah blok dengan hasil antara sebelumnya). Namun, algoritma *hash* yang populer biasanya menggunakan fungsi kompresi yang perilakunya dibuat berbeda pada setiap iterasi, yang bertujuan untuk meningkatkan aspek keamanan algoritma secara keseluruhan.

Setelah proses iterasi selesai, hasil akhirnya dapat diproses lebih lanjut oleh sebuah fungsi yang disebut *finalization function*, yang bertujuan untuk semakin meningkatkan keamanan hasil *hash*. Fungsi ini secara umum dapat berupa sebuah fungsi enkripsi yang menerima sebuah masukan berupa hasil akhir iterasi dan menghasilkan keluaran akhir dari algoritma *hash*. Fungsi final ini juga dapat berperan sebagai fungsi kompresi akhir, yang mengubah ukuran hasil akhir *hashing*. Perlu diperhatikan bahwa fungsi finalisasi ini tidak terdapat dalam rancangan awal Merkle-Damgård *Structure*. Ia merupakan solusi yang sering digunakan oleh algoritma *hashing* berbasis struktur ini, bagi sebuah jenis serangan terhadap kelemahan Merkle-Damgård *Structure.*

Skema Merkle-Damgård *Structure* dapat dilihat pada **Gambar 1** [2]. $\boldsymbol{x_i}$ adalah blok masukan, ***IV*** adalah *initialization vector*, $\boldsymbol{f}$ ialah fungsi kompresi, $\boldsymbol{g}$ ialah *finalization function*, sedangkan ***H(x)*** ialah hasil akhir (nilai *hash*).

3.2. Kelebihan

Alasan utama yang menyebabkan konstruksi Merkle-Damgård *Structure* sangat populer adalah salah satu sifatnya yakni bahwa jika fungsi kompresi $\boldsymbol{f}$ *collision-resistant*, maka keseluruhan konstruksi pasti akan bersifat *collision-resistant* juga [2]. Hal ini telah dibuktikan sendiri oleh para penciptanya (Ralph Merkle dan Ivan Damgård). Bukti sifat ini juga tercantum dalam [8], sebagai berikut. Misalkan sebuah algoritma *hashing* H bekerja dengan Merkle-Damgård *Structure*, fungsi kompresi h, serta *initialization vector* IV. Misalkan terdapat sepasang *string* masukan yang berbeda, x dan y, sedemikian hingga H(x) = H(y). Jika panjang x sama dengan panjang y, maka dapat disimpulkan bahwa pasti terjadi *collision* pada h sebab bangun konstruksi dan jumlah iterasi pada komputasi H(x) dan H(y) identik akibat kesamaan panjang masukan. Jika panjang x tidak sama dengan panjang y, maka terjadinya *collision* pada H ini berimplikasi pada salah satu dari dua hal: h mengalami *collision* atau h tidak bersifat satu-arah sebab dari sembarang keluaran dan masukan dapat diperoleh *pre-image* dari IV.

Sifat satu-arah atau disebut juga sifat *pre-image resistant* dianggap trivial sebab dalam konteks makalah ini, serta dalam konteks pembahasan tandatangan digital secara umum, fungsi *hash* pasti bersifat satu-arah, sehingga syarat itu dapat dihapus dari bukti. Jadi, jika dapat dibuktikan bahwa h bersifat *collision resistant*, maka Merkle-Damgård *Structure* H yang menggunakannya pun pasti bersifat *collision resistant*. QED

Merkle-Damgård *Structure* juga dianggap sebagai sebuah konstruksi yang generik. Artinya, konstruksi ini dapat diimplementasikan tanpa modifikasi, apapun algoritma yang dipakai sebagai ***f*** (dan ***g***).

Seperti terlihat pada bagian sebelumnya, Merkle-Damgård *Structure* bekerja secara sekuensial pada dokumen atau *string* yang diproses, hanya membutuhkan sekali iterasi terhadap keseluruhan dokumen. Kenyataan ini menyebabkan Merkle-Damgård *Structure* dapat diimplementasikan sebagai *stream cipher/encryption*, sehingga dapat bekerja pada pemrosesan secara *online* dimana dokumen atau *string* masukan datang berupa paket-paket kecil data alih-alih sebuah dokumen utuh.

3.3. Kekurangan

Beberapa kelemahan pada Merkle-Damgård *Structure* telah berhasil ditemukan dan diusulkan oleh beberapa pakar kriptografi. Bagian ini memberi daftar kelemahan-kelemahan tersebut, yang dikumpulkan dari [2], [5], dan [6]. Kelemahan dalam aspek kriptanalisis akan dibahas lebih mendalam pada bagian tersendiri.

Secara umum, Merkle-Damgård *Structure* sudah memenuhi kriteria fungsi *hash* yang baik. Namun demikian, dalam [5] disebutkan bahwa Merkle-Damgård *Structure* memiliki sifat *failure-unfriendly*. Artinya, jika seandainya fungsi kompresi ***f*** gagal memenuhi kriteria *collision resistance*, maka konstruksi ini akan runtuh dan menjadi sangat rentan (*vulnerable*). Ini merupakan kebalikan dari kekuatan utama Merkle-Damgård *Structure*, yang menunjukkan bahwa 'nyawa' struktur ini memang berada pada ***f***. Peran Merkle-Damgård *Structure* sesungguhnya hanyalah sebagai ***domain extender***, yakni memperluas kapabilitas ***f*** agar mampu menerima masukan dengan panjang sembarang [6].

Beberapa kelemahan yang disebabkan oleh bentuk Merkle-Damgård *Structure* itu sendiri (kelemahan struktural) juga telah ditemukan. Mereka diacu sebagai suatu serangan terhadap titik-titik lemah tersebut. Karena itu, pembahasan lebih jauh ditempatkan pada bagian yang membahas kriptanalisis terhadap Merkle-Damgård *Structure*. Serangan-serangan tersebut ialah:

1. *Length Extension Attack*, yakni bila telah diketahui
   `h = H(x)`
   maka dengan mudah dapat dicari *h'* untuk sembarang *y* sedemikian hingga
   `h' = H(x||y)`
   tanpa perlu mengetahui *x*. *x*||*y* berarti x dikonkatenasi dengan y.
2. Berdasarkan serangan di atas, dapat dilakukan *Second Collision Attack*, yakni bila telah ditemukan sepasang masukan yang mengalami *collision*, sangat mudah untuk menemukan banyak *collision* lainnya. Misalkan terjadi *collision* antara *x* dan *y*, yaitu
   `H(x) = H(y)`
   maka dapat ditemukan dengan mudah
   `H(x||z) = (y||z)`
   untuk sembarang ekstensi masukan *z*.
3. Beberapa serangan lain,seperti *Multiple Collision Attack (Joux Attack)* dan *Multiple Preimage Attack*, juga telah berhasil dilakukan dalam waktu yang lebih cepat daripada ekspektasi. Semakin cepat suatu serangan menemukan hasil, berarti semakin lemah suatu teknik kriptografi.

4. Analisis

Bagian ini akan membahas secara lebih mendalam aspek komputasi dan aspek keamanan (kriptanalisis) dari Merkle-Damgård *Structure*. Beberapa kemungkinan solusi, pengembangan atau penyempurnaan terhadap kelemahan-kelemahan konstruksi juga dipaparkan. Bagian ini akan diakhiri dengan pembahasan tentang hubungan antara hasil analisis teoretis dengan aplikasi Merkle-Damgård *Structure* dalam teknik kriptografi, khususnya tandatangan digital.

4.1. Komputasi

Seperti telah dijelaskan sebelumnya, Merkle-Damgård *Structure* hanya merupakan sebuah *domain extender* bagi sembarang fungsi *hash* ***f***. Oleh karena itu, kompleksitas komputasi fungsi *hash* yang menggunakan konstruksi ini sangat

bergantung pada kompleksitas algoritma ***f*** yang digunakan. Jadi, dapat dirumuskan bahwa jika terdapat sebuah fungsi *hash* H yang menggunakan Merkle-Damgård *Structure*:

$$H(\{0,1\}^m) \rightarrow \{0,1\}^n$$

dengan m adalah panjang bit masukan dan n adalah panjang bit keluaran, serta menggunakan fungsi kompresi ***f***:

$$\boldsymbol{f}(\{0,1\}^n,\{0,1\}^k) \rightarrow \{0,1\}^n$$

dengan k ialah panjang blok masukan dimana ***f*** beroperasi dan n ialah panjang blok keluaran, maka kompleksitas algoritma H dari segi waktu komputasi adalah

$$O(H) = (m/k)*O(f)$$

dengan *O(f)* menyatakan kompleksitas algoritma fungsi kompresi ***f***.

Mempertimbangkan rumusan di atas, kemungkinan optimasi Merkle-Damgård *Structure* dapat dilakukan dengan tiga cara:

1. Melakukan optimasi ***f***.
2. Memangkas jumlah iterasi pada konstruksi.
3. Membatasi panjang masukan.

Solusi pertama merupakan solusi yang paling masuk akal mengingat ***f*** adalah inti dari Merkle-Damgård *Structure*. Optimasi terhadap fungsi kompresi biasanya dilakukan dengan cara menyederhanakan algoritma di dalamnya. Namun, penyederhanaan tersebut harus mempertimbangkan kompromi (*trade-off*) antara kualitas atau kekuatan algoritma dengan kompleksitasnya. Kompleksitas algoritma yang kecil secara implisit menyatakan bahwa operasi-operasi yang terlibat di dalamnya bersifat sederhana dan, akibatnya, cenderung lemah. Mengingat kualitas keseluruhan fungsi *hash* bergantung kepada kualitas fungsi kompresi yang digunakan, optimasi ***f*** harus dilakukan dengan cermat meski tidak ada panduan yang ketat mengenai bagaimana mendesain suatu algoritma kompresi ataupun, secara lebih umum, algoritma kriptografi apapun. Ilmu kriptografi hanya menyediakan panduan tentang bagaimana seharusnya kualitas hasil dari suatu algoritma, bukan prosesnya. Hal inilah yang menyebabkan aktivitas desain algoritma kompresi, algoritma *hashing*, ataupun algoritma kriptografi lainnya lebih cenderung merupakan seni (*work of art*) daripada sebuah pekerjaan ilmiah [5].

Solusi kedua, memangkas jumlah iterasi, mau tidak mau berakibat pada modifikasi signifikan pada konstruksi global dan juga modifikasi pada fungsi kompresi. Modifikasi itu perlu karena, sebagaimana telah dijelaskan, jumlah iterasi berbanding lurus dengan panjang masukan, sehingga bilamana jumlah iterasi akan dikurangi, konstruksi harus tetap dapat mempertahankan fungsinya sebagai *domain extender* yang mampu 'menjangkau' keseluruhan rangkaian bit masukan.

Penulis menawarkan solusi semacam ini dengan cara 'melipat' Merkle-Damgård *Structure* sebagai berikut: Misalkan masukan terdiri atas n buah blok. Maka setiap iterasi ke-i memproses blok ke-i dan blok ke-(n-i+1), dengan i = 1,2,3,...,n/2. Dengan kata lain, iterasi pertama memproses bersama-sama blok pertama dan terakhir, iterasi kedua untuk blok kedua dan blok ke-(n-1), demikian seterusnya sampai iterasi bertemu di tengah. Jika jumlah blok ganjil, maka iterasi terakhir memproses sepasang blok identik yaitu blok tengah tersebut. Adapun fungsi kompresi dalam setiap iterasi dapat dijalankan dengan dua alternatif cara.

Cara pertama, dua buah blok masukan diproses secara independen sehingga keluaran setiap iterasi adalah sepasang hasil antara yang akan menjadi masukan bagi iterasi berikutnya. Jadi, melipat Merkle-Damgård *Structure* dengan cara ini sama halnya dengan melakukan sepasang iterasi secara paralel. Cara ini tidak mengakibatkan perlunya modifikasi pada fungsi kompresi, namun karena pelipatan ini hanya merupakan paralelisasi proses, ia tidak berpengaruh apa-apa pada kompleksitas algoritma sehingga optimasi yang diinginkan tidak tercapai. Bagaimanapun, implementasi teknis pada mesin dan/atau bahasa pemrograman tertentu dapat membangkitkan keuntungan cara pelipatan ini (misalnya *threading* pada bahasa pemrograman tertentu), namun hal itu berada di luar bahasan makalah ini yang bersifat teoretis.

Alternatif kedua dalam melipat Merkle-Damgård *Structure* ialah dengan memodifikasi fungsi kompresi agar mampu menerima tiga buah masukan yakni sepasang blok masukan dan hasil iterasi sebelumnya. Meski tidak dapat membuktikannya, penulis merasa optimis bahwa kompleksitas algoritma kompresi hasil modifikasi

$$\boldsymbol{f'}(\{0,1\}^{2k},\{0,1\}^k,\{0,1\}^k) \rightarrow \{0,1\}^{2k}$$

dapat dibuat seoptimal mungkin sedemikian hingga

$$O(H') = (m/2k)*O(f')$$
$$\text{dan}$$
$$O(H') < O(H)$$

Selain memberikan keuntungan di sisi komputasi, pelipatan struktur ini juga dapat mengatasi kelemahan Merkle-Damgård *Structure* yang diakibatkan oleh skemanya yang sekuensial. Pelipatan mengakibatkan skema konstruksi tidak lagi sekuensial (murni) dari awal masukan sampai akhir sehingga beberapa jenis serangan terhadap kelemahan struktural ini tidak lagi berlaku (dibahas lebih lanjut pada bagian selanjutnya). Akan tetapi, perubahan struktur menjadi non-sekuensial juga meniadakan kelebihan Merkle-Damgård *Structure* yang dapat berperilaku seperti *stream cipher*. Merkle-Damgård *Structure* terlipat, misalnya, tidak dapat lagi dimanfaatkan untuk proses *hashing* secara *online* sebab sejak iterasi pertama sudah meminta blok terakhir yang belum terkirim / tersedia.

Solusi ketiga dalam rangka optimasi komputasi Merkle-Damgård *Structure* ialah dengan cara membatasi panjang masukan, yang berakibat langsung pada terbatasnya jumlah iterasi. Cara ini merupakan yang paling sederhana dan, pada prakteknya, paling sering digunakan. Tapi, pembatasan ukuran masukan itu bukan dilakukan demi optimasi komputasi fungsi *hash*, namun karena alasan-alasan lain yang terkait implementasi teknis, seperti pembatasan jumlah memori yang digunakan oleh fungsi serta alasan yang terkait dengan spesifikasi representasi ukuran masukan (*length padding*). Hal itu disebabkan oleh kenyataan bahwa ketika masukan dibatasi, ia tidak akan berpengaruh apa-apa pada kompleksitas algoritma, melainkan berpengaruh pada waktu komputasi total. Jadi, solusi ini menawarkan minimalisasi, bukan optimasi.

4.2. Keamanan (kriptanalisis)

Sebagaimana umumnya sebuah teknik kriptografi, Merkle-Damgård *Structure* juga tidak pernah berhenti ‘diserang’ atau menjadi sasaran kriptanalisis. Karena Merkle-Damgård *Structure* sudah berusia cukup tua dalam skala teknologi kriptografi dan teknologi informasi (diusulkan pertama kali pada 1989), maka ia sudah sangat sering dijadikan objek analisis oleh para insan kriptografi. Oleh karena itu juga, telah terdapat beberapa jenis serangan terhadap kelemahan struktur ini, serta bermacam solusi untuk mengoreksi kelemahan tersebut. Beberapa jenis serangan tesebut akan dipaparkan disini, terutama yang lebih relevan dengan tingkat keamanan aplikasi tandatangan digital.

4.2.1. Kriptanalisis Dasar

Setiap fungsi *hash* memiliki dua properti utama: *collision resistance* dan *pre-image resistance (one-wayness)*. Karena itu kriptanalisis terhadap Merkle-Damgård *Structure* juga terbagi menjadi dua kelompok besar, masing-masing melakukan eksploitasi pada properti tersebut.

Sebagaimana sembarang usaha komputasi, kriptanalisis terhadap Merkle-Damgård *Structure* juga dapat dilakukan dengan cara yang paling ‘primitif’ yaitu *brute force* / *exhaustive search*. Dengan demikian, suatu fungsi *hash* yang ideal harus memiliki *collision resistance* dan *pre-image resistance* pada tingkat tertentu sedemikian hingga kriptanalisis paling mangkus dilakukan secara *brute force* / *exhaustive search*, dengan kompleksitas yang cukup rumit / besar.

Pada Merkle-Damgård *Structure* yang dirumuskan sebagai

$$H(\{0,1\}^m) \rightarrow \{0,1\}^n$$

rumusan kompleksitas *exhaustive search* ialah

$$2^m * O(H)$$

Bagaimanapun juga, suatu fungsi *hash* yang baik tidaklah perlu ideal. Jika kompleksitas untuk metode kriptanalisis terbaik sudah lebih kecil daripada rumusan di atas, namun masih cukup besar sehingga komputasinya sangat sulit (*practically infeasible*), maka fungsi *hash* tersebut sudah dapat dikatakan cukup baik atau aman. Pertimbangan lain dalam mengukur *feasibility* dari sebuah usaha kriptanalisis adalah validitas secara semantik. Konsep ini dapat diilustrasikan dengan mudah: mencari nilai-nilai yang mengalami *collision* adalah satu hal, memilih nilai yang bermakna sehingga bisa dimanfaatkan untuk manipulasi masukan adalah hal lain, yang bahkan lebih sulit. Komputasi dilakukan dalam representasi rangkaian bit, sehingga hasil kriptanalisis juga dalam representasi bit yang, hampir pasti, acak. Hasil

tersebut tidak memiliki representasi *string* yang bermakna dilihat dari sudut pandang bahasa manusia, sehingga praktis tidak menimbulkan ancaman berarti bagi keamanan dokumen.

4.2.2. *Length Extension Attack*

Seperti telah tertulis di bagian lain makalah ini, jenis serangan ini dilakukan sebagai berikut. Bila telah diketahui

$h = H(x)$

maka dengan mudah dapat dicari *h'* untuk sembarang *y* sedemikian hingga

$h' = H(x||y)$

tanpa perlu mengetahui *x*. Namun, tahapan *length padding* yang diusulkan langsung oleh Merkle dan Damgård sendiri dan dinamai Merkle-Damgård *Strengthening*, membuat serangan menjadi sedikit lebih sulit. Agar dapat memberikan *length padding* yang valid bagi string $<x||y>$, panjang x harus diketahui.

Length Extension Attack merupakan salah satu serangan terhadap kelemahan Merkle-Damgård *Structure* yang paling awal ditemukan. Serangan ini didasarkan pada sifat sekuensial pada struktur. Secara sederhana, serangan ini dilakukan dengan cara menggunakan nilai h sebagai *initialization vector* saat menghitung h'. Dengan kata lain, Sifat sekuensial Merkle-Damgård *Structure* memungkinkan perhitungan h' = H(z) dengan cara pertama-tama menghitung nilai h = H(x), dimana x merupakan *prefix* dari z.

Karena *Length Extension Attack* dapat dikatakan berusia hampir setua Merkle-Damgård *Structure* itu sendiri, maka solusi untuk menahan serangan ini pun sudah umum. Bahkan, skema Merkle-Damgård *Structure* yang dikenal saat ini sudah bebas dari serangan ini, berkat keberadaan *finalization function*. Dengan adanya fungsi finalisasi ini, sifat operasi *hashing* menjadi tertutup. Pengertian tertutup disini bukanlah seperti makna operasi tertutup secara matematis, melainkan tertutup (*final*) sehingga tidak dapat dilanjutkan untuk menghitung nilai *hash* masukan lain yang merupakan perpanjangan (*extension*) dari masukan semula. Sembarang solusi lain, seperti pelipatan Merkle-Damgård *Structure*, yang membuat struktur tersebut kehilangan sifat sekuensial-terbuka juga dapat digunakan untuk mengatasi serangan ini.

Berdasarkan fakta di atas (pun fakta bahwa serangan ini hampir tidak bermanfaat dalam kriptanalisis tandatangan digital), *Length Extension Attack* saat ini tidak dieksploitasi lagi. Namun, ia masih berguna dalam kriptanalisis Merkle-Damgård *Structure*, seperti yang akan dijelaskan pada bagian berikutnya.

4.2.3. *Second Collision Attack*

Berdasarkan *Length Extension Attack*, dapat dilakukan *Second Collision Attack*, yakni bila telah ditemukan sepasang masukan yang mengalami *collision*, sangat mudah untuk menemukan banyak *collision* lainnya. Misalkan terjadi *collision* antara *x* dan *y*, yaitu

$H(x) = H(y)$

maka dapat ditemukan dengan mudah

$H(x||z) = (y||z)$

untuk sembarang ekstensi masukan *z*.

Dalam serangan ini, terlihat bahwa *Length Extension Attack* masih berperan dalam kriptanalisis terhadap Merkle-Damgård *Structure*. Jika suatu skema pengembangan dari Merkle-Damgård *Structure* orisinal memiliki daya tahan terhadap *Second Collision Attack*, maka ketahanan terhadap *Length Collision Attack* adalah syarat perlu.

Tidak seperti *Length Extension Attack* yang *practically obsolete*, *Second Collision Attack* masih cukup relevan dengan kriptanalisis tandatangan digital karena sembarang jenis *collision attack* dapat di(salah)gunakan untuk aktivitas *document forgery*.

4.2.4. *Multiple Collision Attack* [5]

Berbeda dengan *Second Collision Attack* yang harus bermodalkan pasangan masukan yang *collided*, yakni memiliki nilai *hash* yang sama, *Multiple Collision Attack* dimulai tanpa pengetahuan apapun (*start from scratch*). Sepintas, melakukan serangan ini merupakan tugas yang berat. Hal ini tampak dari ekspektasi awal terhadap Merkle-Damgård *Structure*, yakni mengharuskan pelaku serangan ini menghabiskan waktu sebanyak

$$2^{n(2^k-1)/2^k}$$

untuk menemukan sejumlah 2^k nilai masukan yang *collided* satu sama lain. Namun, Antoine Joux menemukan metode *Multiple Collision Attack* yang lebih mangkus, yakni untuk tugas yang sama hanya memerlukan waktu

$$k * 2^{n/2}$$

dengan n ialah jumlah bit keluaran dan k adalah jumlah blok masukan.

Prinsip kerja *Joux Attack* adalah *divide and conquer*. Serangan ini bekerja dengan cara menemukan sejumlah k pasang blok yang *collided* oleh fungsi kompresi (serangan ini mengasumsikan terdapat algoritma yang cukup mangkus untuk menemukan *collision* dalam skala blok yang cukup kecil). Pasangan-pasangan blok
$\{<M_{a1}, M_{b1}>, <M_{a2}, M_{b2}>, <M_{a3}, M_{b3}>, \ldots, <M_{ak}, M_{bk}>\}$
dikatakan mengalami *local collision*. Kemudian, pasangan-pasangan tersebut disusun menjadi 2^k macam masukan lengkap yang nilai *hash*-nya sama, yaitu

$(M_{a1}||M_{a2}||M_{a3}||\ldots||M_{ak})$
$(M_{a1}||M_{a2}||M_{a3}||\ldots||M_{a(k-1)}||M_{bk})$
.
.
.
.
$(M_{b1}||M_{b2}||M_{b3}||\ldots||M_{bk})$

Salah satu solusi untuk menghindari *Joux Attack* ialah dengan meningkatkan kompleksitas fungsi kompresi sehingga usaha menemukan *local collision* menjadi *infeasible* secara komputasi.

Multiple Collision Attack juga relevan dengan kriptanalisis tandatangan digital, bahkan memberikan ancaman yang cukup besar bagi keamanan tandatangan digital karena semakin memperbesar peluang ditemukannya sepasang dokumen yang mengalami *collision* dan keduanya valid secara semantik.

4.2.5. Rumusan Solusi: Merkle-Damgård *Structure* Terlipat

Pada pembahasan tentang komputasi Merkle-Damgård *Structure*, penulis telah memaparkan secara ringkas salah satu contoh solusi untuk optimasi komputasi. Di luar motivasi tersebut, ternyata skema Merkle-Damgård *Structure* terlipat tersebut juga memiliki keuntungan pada aspek keamanannya. Seperti telah dijelaskan, kebanyakan serangan terhadap Merkle-Damgård *Structure* dilakukan dengan memanfaatkan strukturnya yang sekuensial. Karena pelipatan menghapus sifat sekuensial tersebut, jenis-jenis serangan yang mengandalkan hal itu juga gugur dengan sendirinya.

Terdapat dua alternatif solusi yang telah penulis usulkan. Dari segi optimasi komputasi, alternatif kedua yang telah dipaparkan lebih banyak, memiliki potensi yang lebih baik. Namun, jika kita abaikan aspek komputasi tersebut, alternatif pertama justru lebih menarik untuk dikembangkan dalam rangka meningkatkan aspek keamanan algoritma karena strukturnya yang masih sederhana. Salah satu kemungkinan pengembangan adalah dengan mengaplikasikan struktur Jaringan Feistel yang telah memiliki reputasi dalam aplikasi *block cipher*. Jaringan Feistel dianggap cocok untuk diterapkan karena pelipatan Merkle-Damgård *Structure* akan membagi masukan menjadi dua lajur blok masukan, identik dengan skema Jaringan Feistel yang juga melibatkan sepasang lajur blok.

4.3. Implikasi Praktis

Telah dikemukakan pada bagian sebelumnya bahwa terlepas dari bermacam teori yang melandasi kriptanalisis, faktor semantik sangat berpengaruh terhadap ‘manfaat’ yang diperoleh dari hasil kriptanalisis tersebut. Motivasi praktis dalam melakukan kriptanalisis terhadap Merkle-Damgård *Structure*, yang juga mengarah pada kriptanalisis terhadap aplikasi tandatangan digital, ialah *document forgery*. Ketika sampai di tataran praktek ini, pelaku *document forgery* mempunyai lebih banyak lagi pertimbangan, diantaranya:

1. (Faktor Semantik) Apakah mungkin melakukan rekayasa sehingga dihasilkan dokumen palsu sesuai keinginan?
2. (Faktor Sumber Daya) Seberapa banyak sumber daya komputasi (tenaga, mesin, waktu) yang diperlukan untuk melakukan rekayasa itu?
3. (Faktor Perolehan) Apakah manfaat yang diraih dari rekayasa itu nilainya sepadan dengan semua usaha yang telah dilakukan?

Untuk faktor pertama, kriptanalisis Merkle-Damgård *Structure* masih jauh dari tingkat yang diharapkan tersebut. Pada tingkatan yang paling sederhana, faktor semantik memicu penggunaan teori peluang untuk mencari validitas semantik tersebut, sedangkan dari semua teknik kriptanalisis terhadap Merkle-Damgård *Structure* yang ada saat ini, belum ada teknik yang mampu menjanjikan peluang keberhasilan menemukan validitas semantik yang cukup besar sampai layak untuk dicoba. Pada titik ekstrem lain, percobaan untuk meningkatkan validitas

semantik akan melibatkan *Natural Language Processing* (NLP). Saat ini, NLP merupakan salah satu masalah komputasi yang paling rumit sehingga tidak memberikan banyak harapan pula.

Faktor kedua berhubungan dengan kompleksitas teknik kriptanalisis. Pelaku kriptanalisis pasti mempunyai batas maksimum usaha yang dapat ia keluarkan, baik dalam bentuk tenaga, biaya, maupun waktu. Jika secara teoretis suatu teknik kriptanalisis membutuhkan sumber daya yang melebihi batas tersebut (misalnya kebutuhan pemrosesan dengan superkomputer, waktu komputasi yang sangat lama, dsb.), seberapapun besarnya manfaat yang dijanjikan, teknik tersebut akan dianggap tidak layak (*infeasible*) dan tidak akan dilakukan. Dari analisis bagian sebelumnya dapat ditarik hipotesis bahwa sampai saat ini hampir semua teknik kriptanalisis terhadap Merkle-Damgård *Structure* masih berstatus *practically infeasible*.

Faktor ketiga merupakan perwakilan dari motif ekonomi dasar yang melandasi semua aktivitas manusia. Jika manfaat yang akan diperoleh nilainya lebih kecil dari prospek nilai usaha yang dikeluarkan, maka suatu teknik kriptanalisis juga dianggap tidak praktis. Faktor ini juga dapat mengambil bentuk lain dalam aspek waktu: bilamana suatu teknik kriptanalisis secara potensial memerlukan waktu komputasi yang melebihi masa keberlakuan objek kriptanalisis itu sendiri. Berhubungan dengan faktor kedua, secara umum kriptanalisis terhadap Merkle-Damgård *Structure* masih bersifat tidak praktis dan tidak ekonomis.

5. Simpulan

Merkle-Damgård *Structure* merupakan konstruksi fungsi *hash* yang sangat sederhana namun dapat memiliki kapabilitas serta tingkat keamanan yang cukup memadai sebagai modal dasar. Setelah melalui berbagai ujian dan serangan kriptanalisis serta penyempurnaan skema, Merkle-Damgård *Structure* menjadi cukup tangguh sehingga menjadi basis bagi hampir semua algoritma *hashing* yang ada saat ini.

Kekuatan konstruksi ini terbukti secara praktis belum dapat memotivasi pihak-pihak yang ingin memecahkannya dengan tujuan negatif. Oleh karena itu, Merkle-Damgård *Structure* masih layak dipertahankan sebagai basis algoritma tandatangan digital.

Bagaimanapun, selalu ada potensi untuk mengembangkan Merkle-Damgård *Structure* lebih lanjut. Penulis telah menawarkan sebuah varian baru dari Merkle-Damgård *Structure* yang melibatkan 'pelipatan' dan pemanfaatan Jaringan Feistel. Namun, rumusan solusi tersebut masih dalam bentuk teori yang 'mentah' dan sangat membutuhkan analisis serta pengujian yang mendalam: hal-hal yang dengan sangat menyesal belum mampu penulis tuangkan dalam makalah ini.

Pustaka

1. *Kriptografi, Diktat Kuliah IF5054.* Rinaldi Munir.
2. *Merkle-Damgård Hash Function: from Wikipedia, the free encyclopedia*. NN. (http://en.wikipedia.org/wiki/Merkle-Damgård hash function)
3. *Merkle-Damgård Revisited : How to Construct a Hash Function*. Jean-Sebastien Coron, Yevgeniy Dodis, Cecile Malinaud, dan Prashand Puniya. (http://cs.nyu.edu/~puniya/papers/merkle.pdf)
4. *Constructing Secure Hash Functions by Enhancing Merkle-Damgård Construction.* Praveen Gauravaram, William Millan, Ed Dawson, dan Kapali Viswanathan. (http://www.isi.qut.edu.au/people/subramap/ACISP06_final_version.pdf)
5. *Cryptographic Hash Functions, Recent Results on Cryptanalysis and their Implication on System Security.* Rüdiger Weis dan Stefan Lucks. (http://www.sane.nl/sane2006/program/final-papers/R10.pdf)
6. *Hash Functions: Theory, Attacks, and Applications*. Ilya Mironov. (http://research.microsoft.com/users/mironov/papers/hash_survey.pdf)
7. *The Design Principle of Hash Function with Merkle-Damgård Construction.* Duo Lei, Feng Guozhu, Li Chao, Feng Keqin, Longjiang Qu. (http://eprint.iacr.org/2006/135.pdf)
8. *Domain Extender for Collision Resistant Hash Functions: Improving Upon Merkle-Damgård Iteration*. Palash Sarkar. (http://eprint.iacr.org/2003/173.pdf)
9. *Secrecy, Authentication, and Public Key Systems*. Ralph Charles Merkle. (http://www.merkle.com/papers/Thesis1979.pdf)