KELEMAHAN FUNGSI *MESSAGE DIGEST 5*

Satrio Dewantono 13509051
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509051@std.stei.itb.ac.id

ABSTRAK

Message Digest 5 ***(MD5) adalah suatu fungsi kriptografik berbasis*** **hash** ***yang sempat populer sejak pengembangannya pada tahun 1991 oleh profesor MIT bernama Ronald Rivest sebagai pengembangan dari MD4. Hingga waktu yang tak lama dari saat ini MD5 telah digunakan secara ekstensif sebagai salah satu algoritma enkripsi standar dalam berbagai keperluan autentikasi dalam dunia internet, seperti contohnya pembuatan kunci sertifikat SSL. Bahkan berbagai bahasa pemrograman yang sering digunakan untuk keperluan*** **web-based** ***seperti PHP dan PERL memiliki fungsi MD5 otomatis yang langsung menghasilkan*** **hash** ***MD5 dari suatu nilai yang dimasukkan.***

Kelemahan dari MD5 sendiri mulai ditemukan tidak lama setelah peluncurannya – pada tahun 1996. Sejak saat itu adalah suatu fakta yang dapat diterima bahwa MD5 cenderung rentan terhadap serangan **collision** ***yaitu suatu peristiwa di mana dua nilai yang berbeda dapat memiliki nilai*** **hash** ***yang sama. Baru setelah tahun 2008 - di mana telah ditemukan cara untuk memanfaatkan*** **collision** ***ini untuk memalsukan sertifikat SSL jejaring palsu – MD5 pun akhirnya divonis tidak cocok untuk dipakai sebagai fungsi enkripsi yang membutuhkan ketahanan dari serangan*** **collision.**

Untuk alternatif dari MD5, telah dianjurkan penggunaan algoritma "saudara"nya yaitu SHA1 **(Secure Hash Algorithm-1)** ***atas ketahanan algoritma ini terhadap serangan*** **collision** ***yang relatif lebih mumpuni dibandingkan dengan MD5 . Namun apakah dengan ini berarti MD5 sudah pantas ditinggalkan dan tidak digunakan sama sekali?***

Makalah ini bertujuan untuk memberi analisis mengenai kelemahan-kelemahan yang telah ditemui pada MD5, dampak-dampak yang ada, serta menganalisis beberapa alternatif.

1. PENDAHULUAN

Sejak permulaan peluncurannya pada tahun 1991 hingga saat makalah ini dibuat, algoritma *Message Digest 5* yang dirancang untuk memperbaiki kelemahan pendahulunya, MD4, oleh Ronald Rivest, telah menjadi salah satu fungsi kriptografik populer untuk berbagai macam keperluan autentikasi, baik enkripsi maupun autentikasi pesan. Penggunaan ekstensif ini semakin benar adanya setiap tahunnya, hingga kemudian MD5 menjadi salah satu algoritma standar *de facto* dalam dunia internet. Hal ini dapat dilihat salah satunya dari penggunaan MD5 sebagai fungsi enkripsi sertifikat SSL *(Secure Sockets Layer)* dari sejumlah besar situs web, tidak terkecuali situs resmi beberapa pemerintahan termasuk Amerika Serikat (SSL sendiri merupakan suatu protokol yang dikembangkan oleh Netscape yang memungkinkan komunikasi terautentikasi dan terenkripsi antara pengguna dan server).

.

Beberapa contoh penggunaan MD5 lainnya antara lain adalah untuk pembuatan *checksum* dalam penyimpanan berkas pada suatu server dan untuk enkripsi tambahan pada penyimpanan kata sandi (meskipun MD5 sendiri bukanlah merupakan fungsi enkripsi).

Cara kerja dari fungsi kriptografis ini pada dasarnya adalah pemrosesan suatu masukan dengan panjang *n* dan mengembalikan keluaran dengan ukuran 128 bit sesuai dengan masukan itu. Berikut langkah-langkah yang dilakukan oleh algoritma MD5 dalam pemrosesan masukan menjadi keluaran yang diinginkan:

1. Masukan dengan panjang *n* yang diterima diberikan *padding* (tambahan karakter) agar panjangnya dapat habis dibagi dengan 512. Langkah awal *padding* ini adalah dengan melakukan masukan nilai 1 diikuti dengan 0 sejumlah yang diperlukan (*npadding*) sehingga panjang masukan kongruen dengan 448 (mod 512), yang kemudian diikuti dengan 64 bit yang digunakan untuk menyimpan panjang masukan sesungguhnya sehingga (n + *npadding* + 64) (mod 512) = 0. Apabila bit untuk menyimpan nilai *n* melampaui 64 (kemungkinan hal ini terjadi sangat kecil, di mana $n > 2^{64}$), orde terkecil dari bilangan ini yang masih bisa dimuat dalam 64 bit-lah yang digunakan. Pada akhirnya, sebagai konsekuensi hasil 0 dari modulo 512, masukan akan dapat dibagi menjadi 16 *word* yang masing-masing berukuran 32 bit.
2. *Word* konstanta sejumlah 4 berukuran 32 bit diinisialisasikan sebagai *buffer* dalam bentuk

heksadesimal. *Buffer* ini digunakan sebagai tempat penyimpanan keluaran hasil algoritma yang akan berukuran 128 bit. Contoh inisialisasi (dalam heksadesimal):

```
word A: 01 23 45 67
word B: 89 ab cd ef
word C: fe dc ba 98
word D: 76 54 32 10
```

3. Masukan kemudian diproses per 3 blok 32 bit dengan memasukkannya masing-masing ke empat fungsi berikut (X,Y,Z merujuk pada masing-masing dari 3 blok tersebut):

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$H(X,Y,Z) = X \otimes Y \otimes Z$$
$$I(X,Y,Z) = Y \otimes (X \vee \neg Z)$$

Proses ini dilakukan dalam 4 "ronde", di mana setiap satu ronde terdapat enam belas kali operasi salah satu fungsi di atas terhadap tiga blok tertentu sehingga menghasilkan suatu *state* dengan ukuran 16 bit.

4. Hasil dari setiap ronde ini dimasukkan ke *buffer* A,B,C,D yang telah dibuat sebelumnya dengan tidak melupakan penambahan dengan nilai A,B,C,D sebelumnya yang telah dihasilkan dari ronde terdahulu.

5. Keluaran pada saat ini merupakan konkatenasi antara A,B,C,D yang masing-masing berukuran 32 bit, sehingga menghasilkan keluaran berukuran 128 bit.

Dari pengamatan proses di atas dapat disimpulkan bahwa untuk masukan bernilai berapapun, *hash* MD5 yang akan dihasilkan akan selalu berukuran 128 bit. Hal ini merupakan salah satu dari beberapa hal yang menjadi potensial kelemahan dari fungsi MD5 yang populer ini.

2. KELEMAHAN MD5

2.1.Sekilas Mengenai *Collision*

Seperti telah dibahas di atas, proses MD5 akan menghasilkan keluaran yang ukurannya selalu 128 bit. Ini berarti bahwa untuk kemungkinan panjang masukan yang tak terhingga, hanya akan ada 2^{128} kemungkinan nilai *hash* yang akan dihasilkan MD5. Bahkan untuk nilai kosong:

X = ""

Hash MD5 akan menghasilkan:

H(X) = 7215ee9c7d9dc229d2921a40e899ec5f

Yang berukuran 128-bit.

Hal ini memberikan indikasi awal bahwa MD5 akan rentan terhadap suatu situasi di mana *f(x1)=f(x2)*, atau suatu masukan akan memiliki nilai keluaran *hash* yang sama dengan suatu nilai masukan yang lain. Situasi seperti ini disebut *collision* ("tabrakan").

Tak dinyana jumlah 2^{128} ini sendiri adalah jumlah yang relatif besar, dan apabila digunakan metode *brute force* murni untuk mencoba mencari situasi *collision* ini di mana akan dicoba masukan-masukan acak sehingga nilai *hash* masukan tersebut akan sama dengan nilai *hash* yang dibandingkan maka secara teoretis dan tanpa memperhatikan teori probabilitas kasus terburuk dari algoritma *brute force* ini, diberikan satu proses perbandingan masukan memakan waktu 1 milisekon, akan memiliki t=2^{128} milisekon atau 1,07 x 10^{28} tahun.

Dengan memperhatikan probabilitas, maka akan ditemui suatu paradoks yang disebut *birthday paradox* di mana dianalisis kemungkinan sejumlah orang memiliki tanggal lahir yang sama:

Diberikan *n* jumlah orang, maka berdasarkan prinsip *pigeonhole* kemungkinan terdapat dua orang yang memiliki tanggal lahir yang sama (*p(n)*) dapat ditemukan dengan (asumsi 1 tahun = 365 hari)

$$p(n) = 1 - \bar{p}(n)$$
$$\bar{p}(n) = 1 \times \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \times \ldots \times \left(1 - \frac{n-1}{365}\right)$$
$$= \frac{365 \times 364 \times \ldots \times (365 - n + 1)}{365^n}$$
$$= \frac{365!}{365^n (365-n)!}$$
$$= \frac{n! \binom{365}{n}}{365^n}$$

Dengan aproksimasi Taylor:

$$\bar{p}(n) = e^{-(n(n-1)/2)/365}$$
$$p(n) \approx 1 - e^{-n^2/(2\times 365)}$$

Apabila 365 disubstitusikan dengan *H* yang berlaku sebagai peubah, maka dari persamaan di atas bisa didapat pula *n* yang diperlukan untuk mendapat probabilitas *p* tertentu:

$$p(n) \approx 1 - e^{-n^2/(2\times H)}$$
$$\rightarrow n(p;H) \approx \sqrt{2H \cdot \ln\left(\frac{1}{1-p}\right)}$$

Persamaan di atas berlaku bagi situasi di mana *n* adalah jumlah percobaan masukan pada suatu upaya pencarian *collision*, *H* adalah jumlah nilai *hash* keluaran yang mungkin dari suatu fungsi kriptografik, dan *p* adalah kemungkinan terjadinya *collision* di mana *f(x1)=f(x2)*.

Dari aproksimasi persamaan di atas, jika diberikan

Q(H) adalah jumlah percobaan masukan yang diharapkan hingga penemuan *collision* pertama maka:

$$Q(H) \approx \sqrt{\frac{\pi}{2}H}$$

Dari aproksimasi nilai yang berkelanjutan dapat disimpulkan lebih jauh lagi bahwa untuk fungsi yang menghasilkan keluaran *m* bit jumlah yang harus dicoba "hanya" sejumlah $2^{m/2}$ hingga *collision* pertama.

Dalam kasus MD5 (128 bit), maka dari itu, jumlah percobaan yang diperlukan bukan 2^{128} kali, namun hanya setengah pangkatnya dari ini (2^{64} kali). Jumlah percobaan $2^{n/2}$ ini disebut *birthday bound* dan merupakan batasan bawah keamanan ideal dari fungsi kriptografik secara probabilitas (jika dilakukan percobaan sejumlah ini pada suatu fungsi maka hampir dapat dijamin akan terjadi *collision*). Apabila suatu fungsi membutuhkan percobaan yang lebih sedikit daripada *birthday bound* nya untuk mengalami *collision*, maka fungsi itu dianggap lemah.

Terdapat berbagai macam faktor lain yang dapat mempengaruhi ketahanan suatu fungsi terutama terhadap *collision*. Salah faktor lain itu adalah distribusi dari hasil. Analisis probabilitas *birthday bound* di atas dilakukan dengan asumsi bahwa hasil keluaran dari fungsi terdistribusi normal seperti halnya tanggal pada kalender. Dapat dengan mudah dilihat bahwa peristiwa penemuan *collision* akan lebih mudah didapat apabila distribusi hasil tidak merata, seperti misalnya apabila keluaran fungsi itu cenderung memiliki nilai tertentu.

2.2.Kriptanalisis Lebih Lanjut terhadap MD5

Dalam kasus MD5, kriptanalisis lebih lanjut menunjukkan bahwa MD5 jauh lebih lemah daripada probabilitas teoritis ini. Secepat tahun 1996, sekitar 5 tahun setelah peluncurannya, *collision* sudah mampu ditemukan pada sistem kompresi MD5 oleh H.Dobbertin. Walaupun *collision* yang ditemukan ini adalah *pseudo-collision* dalam arti suatu *collision* dengan nilai awal tertentu dan merupakan suatu peristiwa terisolasi, peringatan kemudian dikeluarkan untuk tidak mengandalkan MD5 untuk kegunaan-kegunaan yang memerlukan ketahanan terhadap *collision*.

Pada tahun 2004, akhirnya ditemukanlah *collision* yang sesungguhnya oleh Xiayoun Wang et.al. dari Shandong University, di mana struktur iteratif dari MD5 dan metode operasinya yang membagi pesan masukan menjadi blok-blok untuk diproses dapat dimanfaatkan untuk mencari kesamaan *hash* per blok dengan suatu nilai permulaan.

Rinciannya, apabila diberikan nilai permulaan IV_0 yang terdiri dari A_0,B_0,C_0,dan D_0 yang masing-masing berukuran 32 bit, maka dapat ditemukan blok {*M*,*M*'} dan {*Ni*, *Ni*'}:

$$M' = M + \Delta C_1$$

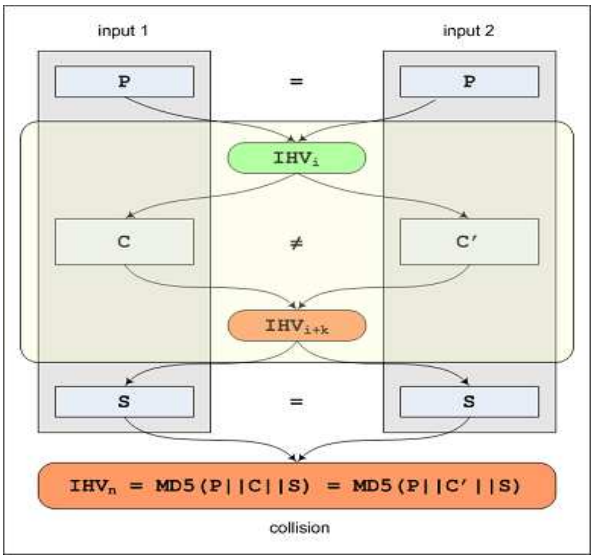
$$N_i' = N_i + \Delta C_2$$

Sehingga

$$MD5(M, N_i) = MD5(M', N_i')$$

Proses ini dilaporkan memakan waktu 1 jam untuk menemukan M dan M', dan selanjutnya sekitar 15 menit untuk menemukan N_i dan N_i' menggunakan komputer IBM P690.

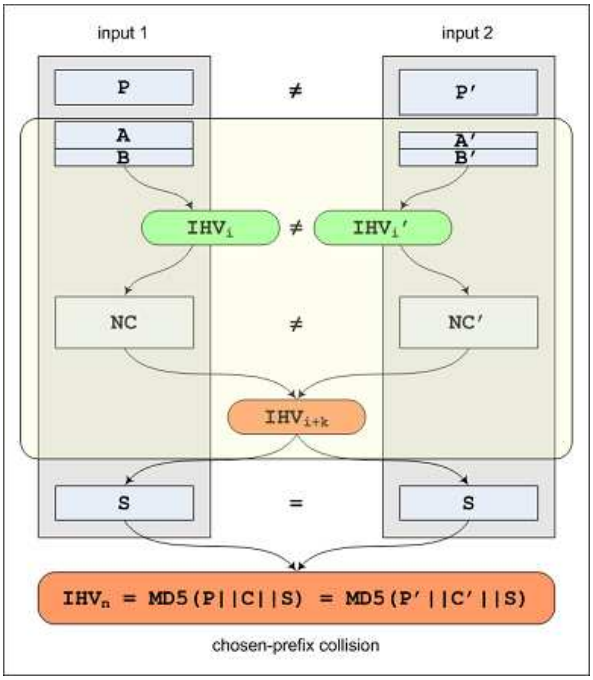
Hal yang lebih berbahaya adalah fakta bahwa karena pemrosesan MD5 bersifat iteratif, jika *MD5(C)=MD5(C')* maka *MD5(P||C||S)=MD5(P||C'||S)*. Di mana *P* dan *S* adalah nilai tambahan bagi masukan dan || merupakan proses konkatenasi. Peristiwa ini disebut *prefix* dan *suffix collision* (*P* dan *S* merupakan *prefix* dan *suffix* secara berturut-turut).



Gambar 2.2.1 Pencocokan *prefix* dan *suffix*

Hal ini berarti bahwa masukan yang sama sekali berbeda isinya pun dapat dengan relatif mudah disesuaikan sehingga nilai *hash*-nya sama dengan suatu pesan lain, asalkan terdapat potongan dari kedua pesan itu yang *collision*-nya sudah ditemukan dengan metode dari Wang et.al.

Lebih jauh lagi, dapat digunakan metode yang disebut *chosen-prefix collision* di mana suatu blok *collision* {*C*,*C*'} dapat ditemukan dengan masukan dua pasangan *prefix* {*P*,*P*'} dan *suffix S*.



Gambar 2.2.2 *Chosen-Prefix Collision*

2.3 *Preimage Attack* pada MD5

Preimage attack merupakan serangan terhadap fungsi *hash* yang menyerupai *collision attack*, namun dengan tujuan mencari masukan *m2* apabila masukan *m1* sudah diketahui, sehingga *f(m1) = f(m2)*. Tidak seperti *collision attack* di mana tujuannya adalah mencari kedua masukan *m1* dan *m2*. Serangan *preimage* pada umumnya lebih kompleks dibandingkan serangan *collision* dan tidak jarang alternatif satu-satunya untuk melakukan serangan ini adalah dengan menggunakan *brute force.*

Pada April 2009, MD5 telah berhasil diserang *preimage* secara teoritis, dengan masukan yang dikendalikan. Serangan *preimage* ini memiliki kompleksitas $2^{123,4}$.

2.4 Aplikasi dan Efek Kelemahan MD5

Telah disinggung berkali-kali sebelumnya bahwa MD5 sebagai fungsi populer memiliki banyak kegunaan. Tentunya berbagai kegiatan yang menggunakan MD5 ini telah mengalami efek dari ditemukannya kelemahan-kelemahan fatal pada MD5.

Contoh pertama adalah *private key* yang menjadi mudah dimanipulasi. *Private key* biasa digunakan sebagai autentikasi pesan yang disalurkan dari satu individu ke individu yang lain. Pada prakteknya *private key* dapat dianalogikan sebagai tanda tangan untuk dokumen elektronik, dengan bentuk *hash* yang dihasilkan dari fungsi kriptografik untuk menghasilkan suatu datum yang unik. Apabila dokumen ini kemudian dirubah, maka hasil pemrosesan dokumen yang telah dirubah tersebut diharapkan tidak akan sesuai lagi dengan *private key* yang dihasilkan dari dokumen asli, sehingga setiap upaya perubahan terhadap dokumen dapat terdeteksi.

Apabila fungsi yang digunakan untuk menghasilkan *private key* ini rentan terhadap *collision*, maka perubahan isi dokumen akan mungkin, selama hasil *hash* dari dokumen yang telah dirubah mengalami *collision* dengan *hash* dokumen sebelumnya.

Kasus manipulasi ini dapat diilustrasikan sebaga berikut: A adalah seorang agen yang bekerja pada suatu badan intelijen X, namun sebenarnya A adalah seorang mata-mata bagi negara musuh. A ditugaskan untuk mendapatkan dokumen-dokumen paling rahasia yang tersimpan pada badan intelijen X.

Untuk mendapatkan dokumen-dokumen tersebut, A kemudian berpura-pura hendak keluar dari pekerjaannya di badan intelijen tersebut dan meminta atasannya di badan intelijen X bernama B untuk membuatkan surat rekomendasi untuknya agar bisa digunakan untuk melamar di tempat kerja lainnya. B pun membuatkan surat tersebut lengkap dengan *private key*-nya, yang dihasilkan dari masukan surat tersebut ke fungsi kriptografik, untuk memastikan keaslian surat tersebut.

Dengan berbekal surat rekomendasi ini, A kemudian membuat suatu surat lagi berisi perintah dari B untuk memberi akses untuk dokumen-dokumen paling rahasia badan intelijen X pada A.

A membuat variasi dari surat perintah ini dalam jumlah besar, dan kemudian mencari variasi yang nilai *hash*-nya sama dengan *private key* surat rekomendasi yang ia dapat. Setelah A berhasil membuat surat perintah yang sesuai dengan *private key* surat rekomendasi, A pun dapat menggunakan surat perintah ini seolah-olah surat tersebut benar-benar dibuat oleh atasannya.

Contoh kedua dari efek kelemahan MD5 adalah pembuatan sertifikat SSL palsu.

Sertifikat SSL digunakan untuk menanggulangi tindakan *phishing*, di mana seseorang membuat halaman web yang tampilannya sama persis dengan yang ingin didatangi oleh pengguna (biasanya halaman *login* atau pembayaran yang meminta si pengguna memasukkan data tertentu), dengan tujuan membuat pengguna itu memasukkan data pribadi seperti nomor kartu kredit atau *password* ke halaman web tiruan tersebut, sehingga bisa dimanfaatkan oleh si pembuat halaman tiruan.

Dengan adanya sertifikat SSL, *browser* dari pengguna akan memastikan bahwa halaman web yang dikunjungi memang benar adalah halaman web asli yang dimaksudkan untuk dikunjungi oleh si pengguna.

Pada sertifikat SSL biasanya terdapat juga *key* yang

berupa nilai heksadesimal yang dihasilkan oleh fungsi kriptografis. Apabila fungsi kriptografis yang digunakan untuk menghasilkan *key* SSL ini lemah, maka akan mudah untuk membuat sertifikat SSL palsu yang dapat mengelabui *browser* dan membuat pengguna semakin percaya bahwa halaman web tiruan yang ia kunjungi adalah halaman web asli.

Khususnya dalam konteks SSL, MD5 sudah dinilai tidak layak lagi untuk digunakan. Pada *Chaos Communication Congress* tanggal 30 Desember 2008, sekelompok peneliti telah berhasil membuat sertifikat SSL palsu yang *hash* MD5-nya menyerupai sertifikat asli dengan menggunakan serangkaian unit *Sony Playstation 3*.

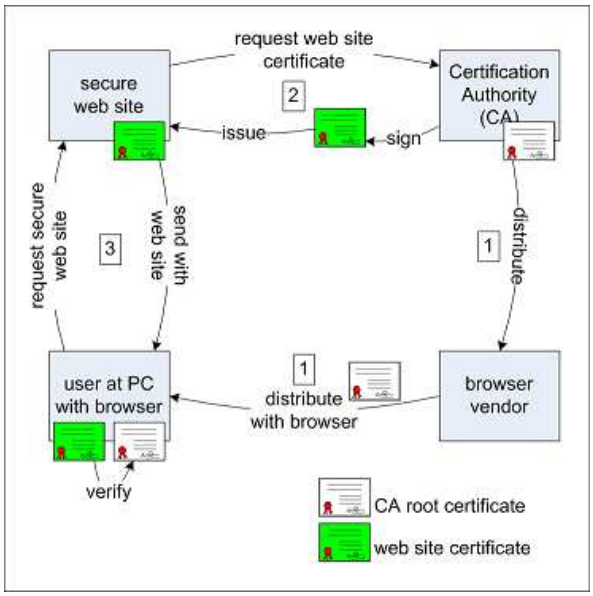


Diagram 2.4.1 Proses normal autentikasi sertifikat

Diagram 2.4.1 menunjukkan proses autentikasi normal yang terjadi antara *website* asli dan pengguna.

1. Pihak yang berwenang mendistribusikan sertifikat (*Certificate Authority/CA)* mendistribusikan sertifikat ke *browser* yang kemudian menyimpan datar *website* yang dapat dipercaya pada komputer pengguna.
2. *Website* yang ingin menggunakan sertifikat untuk memastikan keamanan akses memesan sertifikat dari CA, dan mendapatkan sertifikat yang sudah ditandatangani oleh pihak CA.
3. Ketika pengguna mengakses *website* tersebut, sertifikat dari *website* akan dicocokkan dengan daftar yang dimiliki *browser* pengguna. Apabila cocok, *browser* akan mengakses *website*.

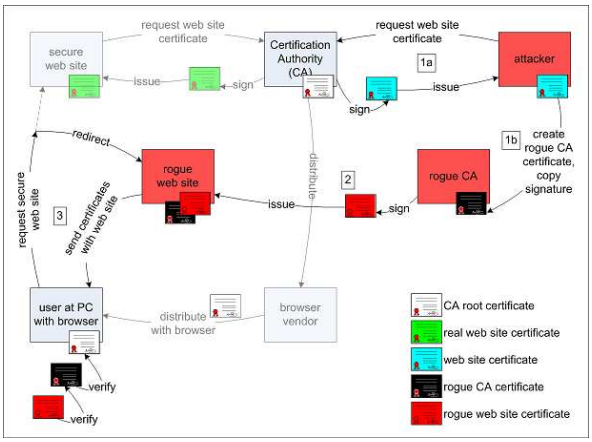


Diagram 2.4.2 Proses autentikasi sertifikat ketika terjadi serangan *phishing*

Diagram 2.4.2 menunjukkan skenario serangan yang mungkin terjadi.

1. a. *Website* penyerang memesan sertifikat dari CA dan menerima sertifikat yang ditandatangani oleh CA.
   b. *Website* penyerang membuat sertifikat palsu yang memiliki *key* serupa dengan sertifikat yang ia miliki melalui CA yang juga tiruan.
2. *Website* tiruan dibuat, dan diberikan sertifikat yang ditandatangani oleh CA palsu.
3. Ketika pengguna hendak mengakses *website*, ia dialihkan ke *website* tiruan yang kemudian memberikan kedua sertifikat palsu yang mensimulasikan sertifikat *browser* dan sertifikat *website*. Karena *key* nya serupa, *browser* tidak akan memperingatkan pengguna.

Contoh ketiga adalah pemecahan enkripsi *password*. *Password* yang hanya dienkripsi MD5 saat ini sudah dianggap tidak aman, antara lain karena MD5 sendiri bukanlah fungsi enkripsi dan pendekripsian *hash* MD5 sudah dapat dengan mudah didapat. Pencarian dengan *Google* akan langsung menghasilkan *website* yang dapat mendekripsi *hash* MD5 menjadi string yang menghasilkannya. Bahkan kotak URL dari beberapa *browser* dapat digunakan sebagai dekriptor MD5 karena *browser* diharuskan mampu untuk mendekripsi alamat *website* yang dienkripsi dengan MD5.

3. ALTERNATIF UNTUK MD5

Untuk menggantikan MD5 yang sudah dipastikan lemah dan berbahaya untuk digunakan, telah disarankan berbagai alternatif. Dalam penggunaannya sebagai fungsi kriptografi SSL, MD5 dapat digantikan dengan fungsi-fungsi lain seperti SHA1 atau SHA256. *Website-website* resmi pemerintahan Amerika Serikat sudah diharuskan untuk mengganti fungsi SSL yang digunakannya dari MD5 menjadi SHA1 pada tahun 2011.

Sedangkan untuk penanggulangan kelemahan MD5 pada enkripsi *password*, dianjurkan untuk menggunakan MD5 dengan *salting*, yaitu konkatenasi nilai string password yang dimasukkan dengan *key* tambahan yang dapat berupa nilai konstan atau nilai yang dibentuk secara acak.

Dalam penggunaannya untuk konteks autentikasi pesan, MD5 yang dimasukkan sebagai inti fungsi HMAC (*Hash-Based Message Authentication Code*), yang kemudian disebut HMAC-MD5, pun masih layak digunakan, karena fungsi HMAC sendiri telah dirancang sedemikian rupa sehingga tidak bergantung pada ketahanan *collision*, termasuk properti ketahanan fungsi intinya. HMAC merupakan pengembangan dari fungsi kriptografik, dengan menambahkan fitur *secret key* pada string yang akan dimasukkan:

$$\mathbf{HMAC}(K,m) = \mathbf{H}\Big((K \oplus \text{opad}) \parallel \mathbf{H}\big((K \oplus \text{ipad}) \parallel \text{m}\big)\Big)$$

Dengan *K=secret key*, opad = *padding luar*, ipad = *padding dalam*, dan || adalah fungsi konkatenasi.

4. AKHIR DARI MD5?

Dengan kelemahan-kelemahan dan alternatifnya seperti di atas, apakah berarti MD5 sudah tidak memiliki kegunaan lagi dan layak untuk ditinggalkan sama sekali? Pada kenyataannya, MD5 menjadi fungsi yang populer karena performanya yang mangkus dibandingkan dengan fungsi-fungsi lain. Berikut tabel performa kecil yang didapat dari pengetesan MD5 terhadap SHA1 dalam beberapa sistem yang berbeda:

	Pentium P5 90MHz	Power Mac 80 MHz	SPARC 4 110 MHz	DEC Alpha 200 MHz
MD5	13.1	3.1	5.1	8.5
SHA1	2.5	1.2	2.0	3.3

Tabel 2.6.1 Perbandingan performa MD5 dan SHA1 (dalam MB/s)

Dapat dilihat perbedaan performa MD5 dan SHA1 yang sangat besar – lebih dari 2 kali lipat. Hal ini dapat menjadi sangat berarti apabila data masukan berukuran sangat besar. Dari sini dapat ditarik kesimpulan bahwa MD5 masih memiliki nilai jual, apabila kelemahannya dapat ditanggulangi.

5. KESIMPULAN

MD5 sebagai suatu fungsi yang sangat ekstensif penggunaannya hingga saat ini, bukan tidak ada kelemahannya. Malah, bisa dibilang benar apabila seseorang mengambil kesimpulan bahwa jumlah besar usaha yang dilakukan berbagai pihak untuk mencari kelemahannya justru muncul dari kenyataan akan luasnya penggunaan fungsi ini.

Adalah hal yang sangat baik bahwa pihak-pihak yang berhasil memecahkan fungsi MD5 ini tidak memiliki maksud untuk menghancurkan, namun justru untuk membangun karena mereka telah bersedia untuk membagikan penemuan mereka ke dunia.

Lantas hal yang masih menjadi pertanyaan adalah mengapa MD5, setelah berbagai kelemahan yang telah diekspos dan usia yang hampir mencapai 20 tahun, masih digunakan dengan sangat luas?

REFERENSI

[1] en.wikipedia.org, diakses 17-12-2010
[2] http://www.win.tue.nl/hashclash/rogue-ca/, diakses 17-12-2010
[3] http://www.ilmuhacking.com/cryptography/md5-itu-berbahaya/, diakses 17-12-2010
[4] Rivest, MD5 the Message Digest Algorithm, Protocol In *RFC 1321*
[5] Xiayoun Wang, Dengguo Feng, Xuejia Lai, Hongbo Yu, Collisions for Hash Functions, Shandong University
[6] http://www.mscs.dal.ca/~selinger/md5collision/, diakses 17-12-2010