# Penanganan Kolisi pada Fungsi *hash* dengan Algoritma Pengembangan *Vigenere Cipher* (menggunakan Deret *Fibonacci*)

Jaisyalmatin Pribadi - 13510084[1]
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*[1]13510084@std.stei.itb.ac.id*

***Abstraksi*—Fungsi hash merupakan fungsi yang menerima suatu masukan string dengan berbagai ukuran dan menghasilkan string baru yang memiliki panjang tetap. Fungsi ini telah banyak digunakan untuk melakukan otentikasi terhadap dokumen yang ada, membuktikan bahwa dokumen tersebut tidak dimanipulasi. Kemampuan fungsi hash dalam melakukan otentikasi disebabkan karakteristik fungsi hash yang merupakan fungsi satu-arah. Fungsi ini tidak dapat mengembalikan string yang dihasilkan menjadi string pada masukan awal. Hal ini membuat fungsi hash hanya dapat digunakan satu kali. Namun, fungsi hash memiliki kekurangan pada sekelompok string masukan tertentu yang menghasilkan string yang sama, terjadinya kolisi.**

***Penulis akan melakukan penanganan permasalahan ini dengan mengkombinasikan fungsi hash dengan algoritma kriptografi klasik, yaitu algoritma pengembangan vigenere cipher. Algoritma tersebut merupakan algoritma klasik yang telah penulis modifikasi sehingga dapat menyelesaikan kekurangan pada algoritma vigenere cipher yang sebenarnya. Tujuan dilakukannya kombinasi ini adalah agar string yang dihasilkan oleh fungsi hash dapat merupakan string yang berbeda dari string masukan yang lain, tidak terjadi kolisi.***

***Index Terms*—kriptografi, fungsi *hash*, *vigenere cipher*, kolisi.**

## I. PENDAHULUAN

Kriptografi adalah seni dan ilmu dalam melakukan pengamanan pesan. Langkah yang dilakukan dalam melakukan penjagaan sebuah pesan adalah dengan mentransformasikan data atau pesan yang memiliki makna, dikenal sebagai *plainteks*, ke dalam bentuk data atau pesan yang tidak memiliki arti, dikenal sebagai *ciphertext*. Bentuk data atau pesan *ciphertext* yang akan dikirimkan oleh pengirim kepada penerima. Saat penerima mendapatkan *ciphertext*, maka penerima akan melakukan langkah sebaliknya, yaitu mentransformasikan kembali *ciphertext* menjadi *plaintext* sehingga data atau pesan dapat dipahami oleh pihak penerima. Transformasi ini umumnya dilakukan menggunakan kunci transfomrasi yang telah disepakati oleh kedua belah pihak di awal pengiriman data atau pesan.

Terdapat banyak algoritma kriptografi yang telah dikenal hingga saat ini. Bentuk yang dimiliiki juga memiliki berbagai keberagaman berdasarkan kategori waktu kemunculan algoritma kriptografi ini. Algoritma klasik merupakan algoritma yang masih menggunakan perpindahan setiap karakter sehingga kalimat yang terbentuk dalam *ciphertext* tidak dapat dipahami. Sedangkan algoritma kriptografi modern merupakan algoritma yang telah melakukan manipulasi dari karakter data atau pesan ke dalam bit-bit sehingga memiliki tingkat kesulitan dalam melakukan kriptanalis.

Saat sebuah data atau pesan sedang dikirim, tidak ada yang dapat memastikan bahwa pesan tersebut tidak diawasi atau dimanipulasi oleh orang lain. Terdapat kemungkinan bahwa pesan tersebut telah dibaca dan dimanipulasi oleh orang lain sehingga pesa yang sampai ke pihak penerima tidaklah pesan yang sesungguhnya. Karena itu digunakanlah sebuah fungsi hash, yaitu fungsi yang dapat menerima masukan berupa string dengan berbagai ukuran dan melakukan manipulasi sehingga menghasilkan string dengan panjang yang tetap. Pihak pengirim sebelumnya dapat mengirimkan nilai dari fungsi hash yang dimiliki oleh pesan yang dikirim, sehingga pihak penerima dapat memastikan bahwa pesan yang dikirim adalah pesan yang sebenarnya berdasarkan nilai fungsi hash yang dimiliki pesan yang sampai kepada pihak penerima. Jika nilai fungsi hashnya sama, pesan tersebut tidak mengalami perubahan. Sebaliknya, jika terdapat manipulasi terhadap pesan, maka nilai dari fungsi hash dapat berubah.

Pada makalah ini, penulis ingin meningkatkan kualitas keamanan dari fungsi hash yang sebelumnya memiliki kekurangan pada terdapatnya kolisis dari beberapa masukan string berbeda yang menghasilkan nilai fungsi hash yang sama. Penulis akan melakukan kombinasi dua algoritma sehingga keamanan yang dimiliki mengalami peningkatan. Ide yang penulis rancang adalah melakukan manipulasi string masukan sehingga string masukan akan memiliki nilai yang lebih berbeda. Nilai yang berbeda ini kemudiakan akan dimasukkan ke dalam fungsi hash sehingga kolisi yang sebelumnya terjadi dapat diatasi.

## II. DASAR TEORI

### A.Vigenere Cipher

Vigenere Cipher merupakan salah satu algoritma kriptografi klasik yang merupakan pengembangan lebih lanjut dari Caesar cipher. Algoritma ini termasuk ke dalam algoritma yang menggunakan cipher substitution, yaitu suatu langkah enkripsi dengan cara melakukan pergeseran setiap karakter yang ada pada plain text. Jumlah pergeseran karakter yang dilakukan ditentukan oleh key yang dimiliki.

Berdasarkan jenisnya, vigenere cipher merupakan polyalphabetic substitution cipher (cipher abjad-majemuk). Ciri algoritma kriptografi jenis ini adalah menggunakan kunci yang berbeda sehingga setiap karakter yang sama pada plain text tidak akan disubstitusi menjadi karakter yang sama pada cipher text.

Pada saat pertama kali ditemukan, langkah dalam melakukan kriptografi vigenere cipher harus menggunakan bujursangkar vigenere cipher untuk menentukan karakter yang akan dienkripsi atau didekripsi.Namun terdapat cara lain dalam melakukan kriptografi viginere cipher, yaitu dengan rumus matematika.

Berikut rumus untuk melakukan:

$$C(p) = (p + ki) \bmod 26$$

Keterangan,
C(p) : karakter cipher
p : karakter plain
ki : karakter ke-i pada key

Sedangkan rumus untuk melakukan dekripsi sebagai berikut:

$$p = (C(p) - ki) \bmod 26$$

Jika panjang key lebih pendek daripada panjang plaintext, maka key akan digunakan secara periodik.

### B. Pengembangan Vigenere Cipher

Pengembangan vigenere cipher merupakan algoritma pengembangan dari bentuk yang telah ada oleh penulis, yang digunakan dalam membuat makalah sebelumnya. Ide dari pengembangan algoritma vigenere cipher ini adalah melakukan pengembangan terhadap key yang digunakan dalam melakukan pergeseran karakter saat melakukan enkripsi dan dekripsi. Jika melihat pada pola metode Kasiski, algoritma enkripsi ini dapat dipecahkan karena mengetahui adanya suatu pola secara periodik sehingga dapat ditebak panjang karakter key. Dibutuhkan suatu cara yang membuat key yang dilakukan untuk melakukan pergeseran tidak dapat ditebak, bahkan tidak dapat diketahui panjang key tersebut.

Maka penulis memiliki ide untuk menggunakan deret matematika Fibonacci sehingga key yang digunakan dalam melakukan pergeseran tidak dapat diketahui panjangnya. Aturan deret Fibonacci kedua menjadi ide dari pengembangan ini, yaitu dengan menjumlahkan dua bilangan sehingga menghasilkan key yang digunakan dalam melakukan pergeseran. Namun tidak seperti aturan deret Fibonacci, penjumlahan yang dilakukan adalah jumlah bilangan pada urutan tersebut dan bilangan setelahnya. Jika telah sampai pada karakter terakhir, penjumlahan akan dilakukan antara bilangan tersebut dengan bilangan yang pertama. Bilangan pertama akan menjadi key dalam proses vigenere cipher dengan pengembangan deret Fibonacci.

Rumus dalam menentukan deret *Fibonacci* sebagai berikut,

$$F(n) = \begin{cases} 0, & \text{jika } n = 0; \\ 1, & \text{jika } n = 1; \\ F(n-1) + F(n-2) & \text{jika tidak.} \end{cases}$$

Deret *fibonacci* yang dihasilkan dari aturan di atas adalah:

0, 1, 1, 2, 3, 5, 8, 13, ..

Namun, pada implementasi deret *Fibonacci* yang akan digunakan, hanya akan menggunakan aturan yang ada pada saat n tidak sama dengan 0 dan 1.

Alasan tidak menggunakan rumus pada deret Fibonacci sercara keseluruhan karena karakter yang akan dienkripsi atau dekripsi dapat berjumlah sangat banyak. Jika key yang digunakan untuk melakukan pergeseran karakter mengikut jumlah pada deret Fibonacci maka dibutuhkan proses

komputasi yang tinggi. Sehingga dibutuhkan jumlah pergeseran yang tidak bernilai besar, lebih dari seribu pergeseran. Terdapat sedikit perbedaan dalam melakukan enkripsi dan dekripsi pada metode pengembangan vigener cipher dengan deret Fibonacci.

Terdapat sedikit perbedaan pada langkah yang diperlukan untuk melakukan enkripsi dengan metode pengembangan vigenere cipher dengan deret Fibonacci. Perbedaan tersebut dapat dilihat bahwa langkah enkripsi tidak membutuhkan key dari luar. Sedangkan pada langkah dekripsi, metode pengembangan ini tetap membutuhkan masukan key untuk melakukan prosesnya.

### C. Fungsi Hash

Fungsi hash merupakan sebuah fungsi yang menerima suatu masukan string dengan ukuran yang beragam, kemudian akan melakukan transformasi sehingga menghasilkan suatu string baru dengan bentuk yang sama (*fixed*). Fungsi ini merupakan fungsi satu arah, yaitu hanya dapat melakukan satu kali transformasi dan tidak dapat dikembalikan menjadi bentuk string seperti di awal masukan. Karena sifat inilah, fungsi hash merupakan

fungsi yang dapat dipercaya. Fungsi ini memiliki tingkat manipulasi yang lebih kecil karena pihak lain tidak dapat mengetahui bentuk awal pesan sebelum dilakukan fungsi hash.

Persamaan fungsi hash adalah sebagai berikut:

$$h = H(M)$$

Keterangan,
M = pesan kuran sembarang
h = nilai hash atau pesan-ringkas (message-digest)

Beberapa sifat fungsi hash lainnya dijelaskan sebagai berikut:

1. Fungsi *H* dapat diterapkan pada blok data berukuran berapa saja.
2. *H* menghasilkan nilai (*h*) dengan panjang tetap (*fixed-length output*).
3. *H*(*x*) mudah dihitung untuk setiap nilai *x* yang diberikan.
4. Untuk setiap *h* yang dihasilkan, tidak mungkin dikembalikan nilai *x* sedemikian sehingga *H*(*x*) = *h*. Itulah sebabnya fungsi *H* dikatakan fungsi *hash* satu-arah (*one-way hash function*).
5. Untuk setiap *x* yang diberikan, tidak mungkin mencari $y \neq x$ sedemikian sehingga *H*(*y*) = *H*(*x*).
6. Tidak mungkin mencari pasangan *x* dan *y* sedemikian sehingga *H*(*x*) = *H*(*y*).

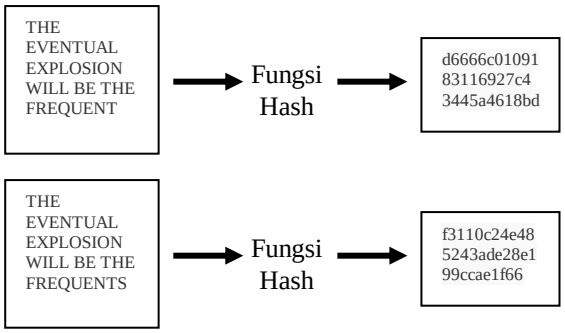
Ada beberapa fungsi hash satu-arah yang terdapat di dalam kriptografi:

1. MD2, MD4, MD5,
2. Secure Hash Function (SHA),
3. Snefru,
4. N-hash,
5. RIPE-MD, dan lain-lain

## III. PERANCANGAN DAN ANALISIS

### A. Skema Awal Fungsi Hash

Fungsi hash memiliki skema umum seperti yang dijelaskan pada sub bab berikut ini:




Skema umum di atas menunjukkan bahwa fungsi hash dapat menerima masukan string yang memiliki panjang dinamis, tidak ditentukan. Output yang dihasilkan oleh fungsi tersebut juga merupakan string yang memiliki panjang yang sama, fixed length.

Jika diamati, kedua contoh plainteks yang digunakan merupakan plainteks yang memiliki tingkat kesamaan yang tinggi. Hanya terdapat perbedaan pada kedua plainteks ini, hanya pada kata terakhir dari plainteks tersebut. Namun hasil dari fungsi hash yang dihasilkan menunjukkan hal yang sangat berbeda. Hal ini membuktikan bahwa fungsi hash merupakan fungsi yang sangat sensitif karena sedikit perubahan pada string masukan, dapat memberikan dampak perubahan yang berbeda pada string yang dihasilkan.

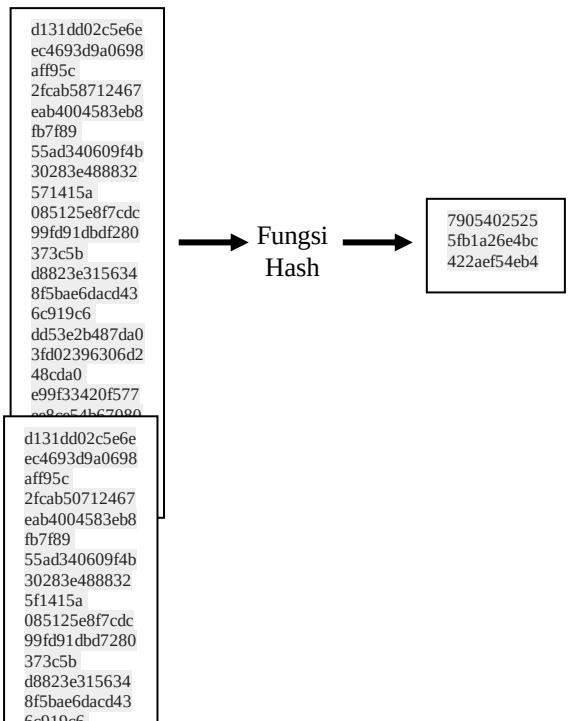
### B. Kolisi pada Fungsi Hash

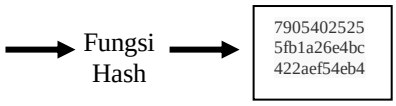
Fungsi hash merupakan fungsi satu arah. Sifat inilah yang membuat fungsi hash menjadi fungsi yang pada awalnya diperkirakan tidak dapat dimanipulasi. Kriptanalisis yang tidak dapat memahami bentuk sebenarnya dari string keluaran dari fungsi hash pasti tidak dapat menganalisis fungsi hash yang telah dilakukan. Panjang message digest yang dimiliki fungsi hash pada MD5 adalah 128 bit. Jika dilakukan brute force, maka dibutuhkan percobaan sebagai $2^{128}$ kali untuk menemukan dua buah pesan atau lebih yang memiliki message digest yang sama. Waktu brute force yang lama ini membuat algoritma MD5 sebagai fungsi hash diyakini telah sempurna.

Namun terdapat permasalahan yang muncul pada keberjalanan fungsi hash. Fungsi hash yang seharusnya menghasilkan string keluaran dalam bentuk yang unik, penuh keragaman, ternyata dapat menghasilkan string yang sama dari dua masukan string yang berbeda.

Pada tahun 1996, Dobbertin menemukan kolisi pada algoritma MD5. Tetapi perlu diingat bahwa kecacatan ini bukan merupakan kelemahan yang fatal karena kolisi hanya terjadi pada kasus tertentu. Pada tahun 2005, Arjen Lenstra, Xiaoyun Wang, dan Benne de Weger melakukan demonstrasi pembentukan dua buah string yang dapat menghasilkan hasil dari fungsi hash dengan nilai yang sama.

Salah satu contoh kolisi yang terjadi adalah sebagai berikut:



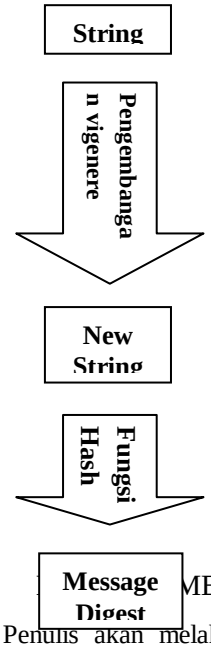



Kedua string masukan tersebut memiliki tingkat kemiripan yang tinggi namun bukan merupakan string yang sama. Namun kedua string masukan ini menghasilkan message digest yang sama.

C. Kombinasi Fungsi Hash dan Pengembangan Vigenere Cipher

Ide yang penulis ajukan dalam menangani kolisi yang terjadi dalam fungsi hash, dalam hal ini algoritma MD5, adalah dengan melakukan manipulasi string yang sebelum dimasukkan ke dalam fungsi hash. Hal ini ditujukan agar dua string masukan yang sebelumnya menyebabkan kolisi dapat teratasi dengan menghasilkan string keluaran yang berbeda. Tujuan kedua adalah dengan menangani kasus reverse analysis, yaitu pembuatan string dengan menganilisis dari keluaran string yang sama yang dapat berasal dari dua string masukan yang berbeda. Manipulasi bentuk string di awal akan menambah kompleksitas dalam terjadinya kolisi dalam kasus serupa.

Ide penulis dapat digambarkan dalam skema umum sebagai berikut.




## IMENTASI DAN PENGUJIAN

Penulis akan melakukan pengujian terhadap ide dari skema yang penulis ajukan untuk menangani kolisi pada fungsi hash, dalam hal ini algoritma MD5. Pengujian ini akan menggunakan contoh kasus dua masukan string yang mengalami kolisi, yang telah ditunjukkan pada bab sebelumnya.

A. Pengembangan Vigenere Cipher

Langkah pertama dalam ide ini adalah melakukan manipulasi string yang akan dimasukkan ke dalam fungsi hash dengan algoritma pengembangan vigenere cipher. Hasil yang diperoleh dari langkah ini adalah sebagai berikut:

Plain text 1:

d131dd02c5e6eec4693d9a0698aff95c
2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e488832571415a
085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6
dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1e
c69821bcb6a8839396f9652b6ff72a70

Cipher text 1:

y131ij02y5w6zji4693z9s0698vkl95y
2xxfh58712467asw4004583jh8bt7a89
55fj340609b4t30283z488832571415f
085125k8b7uyh99lz91vwil280373y5t
y8823j3156348l5xsz6igyv436x919h6
jz53w2w487ig03bv02396306y248hjw0
w99a33420k577ka8uz54g67080g80z1w
x69821gix6s8839396a9652g6lb72s70

Plain text 2:

d131dd02c5e6eec4693d9a0698aff95c
2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a
085125e8f7cdc99fd91dbd7280373c5b
d8823e3156348f5bae6dacd436c919c6
dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080280d1e
c69821bcb6a8839396f965ab6ff72a70

Cipher text 2:

y131ij02y5w6zji4693z9s0698vkl95y
2xxfh50712467asw4004583jh8bt7a89
55fj340609b4t30283z4888325k1415g
085125a8x7xii99bv91ygj7280373y5t
y8823j3156348l5xsz6igyv436x919h6
jz53w23487yf03lz02396306v248xig0
a99x33420a577jk8yw54w67080280i1k
y69821txg6g8839396b965sw6kl72w70

Setelah melalui tahap ini, terbentuk manipulasi string dari bentuk awal string yang sebelumnya dimiliki.

B. Fungsi Hash

Langkah selanjutnya adalah melakukan fungsi hash. String yang digunakan adalah string yang dihasilkan pada langkah sebelumnya.

Blok 1,

y131ij02y5w6zji4693z9s0698vkl95y
2xxfh58712467asw4004583jh8bt7a89
55fj340609b4t30283z488832571415f
085125k8b7uyh99lz91vwil280373y5t
y8823j3156348l5xsz6igyv436x919h6
jz53w2w487ig03bv02396306y248hjw0
w99a33420k577ka8uz54g67080g80z1w
x69821gix6s8839396a9652g6lb72s70

b1789beb02419290e6e8080224f47d32

Blok 2,

y131ij02y5w6zji4693z9s0698vkl95y
2xxfh50712467asw4004583jh8bt7a89
55fj340609b4t30283z4888325k1415g
085125a8x7xii99bv91ygj7280373y5t
y8823j3156348l5xsz6igyv436x919h6
jz53w23487yf03lz02396306v248xig0
a99x33420a577jk8yw54w67080280i1k
y69821txg6g8839396b965sw6kl72w70

0b732be4ebdf11f3a76c3d8f7365b249

## V. KESEIMPULAN

Kesimpulan yang dapat diambil setelah melakukan riset sederhana adalah penanganan kolisi pada fungsi hash dengan melakukan kombinasi dengan algoritma klasik yang telah dimodifikasi, pengembangan vigenere cipher, berhasil dilakukan dengan baik. Dapat dilihat pada bab sebelumnya bahwa message digest yang dihasilkan oleh kedua blok yang sebelumnya menghasilkan kolisi dapat menghasilkan message digest yang berbeda. Analisis yang menyatakan penanganan kolisi terbukti dengan baik.

Tujuan dari makalah ini untuk menangani kolisi pada fungsi hash dapat diselesaikan dengan baik, yang dapat dilihat pada bab sebelumnya. Tujuan kedua yang berupa meningkatkan keamanan fungsi hash dengan menambah kerumitan dalam menghasilkan fungsi hash juga dapat dilakukan dengan baik. Hal tersebut dapat dilihat pada bab analisis.

## REFERENCES

[1] Rinaldi Munir, Slide Materi Kuliah IF4020 Kriptografi.

## 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, 19 Mei 2014

Jaisyalmatin Pribadi - 13510084