# BAB 2
# LANDASAN TEORI

## 2.1 Pengertian Kriptografi

Kriptografi (*cryptography*) berasal dari bahasa Yunani yang terdiri atas kata "cryptos" yang artinya rahasia, dan "graphein" yang artinya tulisan. Berdasarkan terminologi, kriptografi adalah ilmu dan seni untuk menjaga kerahasiaan pesan dengan cara mengubahnya dari satu bentuk ke bentuk lainnya yang tidak dapat dimengerti lagi artinya. Kriptografi disebut ilmu, karena didalamnya menggunakan berbagai metode (rumusan), dan sebagai seni, karena didalamnya membutuhkan teknik khusus daalam mendesainnya. (Rinaldi Munir, 2006).

Kriptografi merupakan cabang ilmu dari kriptologi. Pelaku kriptografi ialah kriptografer (*cryptographer),* yang bertugas untuk mengubah plainteks menjadi cipherteks dengan algoritma dan kunci tertentu. Sedangkan lawan dari kriptografi adalah kriptanalisis (*cryptanalysis),* merupakan ilmu yang memecahkan cipherteks menjadi plainteks kembali tanpa mengatahui kunci, dan pelakunya ialah kriptanalis (*criptanalys*).

Tujuan mendasar dari kripotgrafi itu sendiri adalah sebagai berikut:

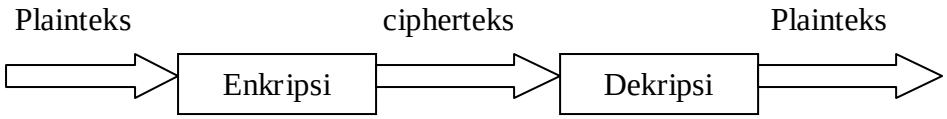
a. Kerahasiaan (*confindentiality*)
   Memastikan bahwa tidak ada yang membaca pesan selain orang yang dituju
b. Integritas data (*data integrity*)
   Suatu layanan yang menjamin bahwa pesan yang asli tidak mengalami perubahan.
c. Otentikasi (*authentication*)
   Mengidentifikasi pihak-pihak yang berkomunikasi maupun mengidentifikasi kebenaran pesan.

d. Nirpenyangkalan (*non-repudiation*)

Layanan yang mencegah terjadinya penyangkalan oleh pengirim pesan atau penyangkalan oleh penerima pesan sudah menerima pesan.

## 2.2 Proses Kriptografi

Urutan proses kriptografi secara umum ditunjukkan oleh Gambar 2.1:




**Gambar 2.1 Proses Kriptografi**

Sebuah plainteks (p) akan diproses dengan proses enkripsi untuk menghasilkan cipherteks (c). Setelah itu untuk memperoleh kembali plainteks (p), cipherteks (c) diproses dengan proses dekripsi. Secara matematis dinyatakan sebagai berikut:

$$E(p) = c \quad (2.1)$$

$$D(c) = p \quad (2.2)$$

$$D(E(p)) = p \quad (2.3)$$

Keamanan dari cipherteks tergantung dari algoritma yang digunakan. Jika algoritma yang digunakan semakin kompleks, maka cipherteks akan semakin aman atau sulit dipecahkan . sebaliknya, jika algoritma yang digunakan sederhana, maka cipherteks akan semakin mudah dipecahkan.

## 2.3 Algoritma Cipher Transposisi

Algoritma cipher transposisi adalah algoritma yang mempermutasikan karakter-karakter yang ada pada plainteks, yakni dengan menyusun ulang karakter. Pada algoritma ini. Contoh penggunaan algoritma ini sebagai berikut:

Plainteks : KRIPTOGRAFI
Cipherteks : IFARGOTPRIK

Cipher transposisi ini memiliki berbagai macam bentuk dan algoritma, diantara adalah cipher transposisi itu adalah *Rail Fence Cipher*, *Route Cipher* dan *Columnar Cipher*. Untuk membatasi masalah, penelitian ini hanya membahas *Columnar transpostion*.

### 2.3.1 *Columnar Transposition*

*Columnar transposition* merupakan salah satu bagian dari cipher transposisi dengan metode kriptografi dimana pesan dituliskan berderet dari suatu panjang yang ditetapkan, lalu dibaca kembali kolom per kolom dengan urutan pembacaan berdasarkan suatu kata kunci. Panjang deret ditentukan oleh panjang kata kunci. Urutan pembacaan kolom berdasarkan urutan kolom. Hal ini dijelaskan pada Tabel 2.1.

Contoh:
Plainteks : PROGRAM STUDI S1 ILMU KOMPUTER USU

**Tabel 2.1 Enkripsi *Columnar Transposition***

| | | | | | |
|---|---|---|---|---|---|
| P | R | O | G | R | A |
| M | S | T | U | D | I |
| S | 1 | I | L | M | U |
| K | O | M | P | U | T |
| E | R | P | Q | R | S |

Cipherteks : PMSKE RS1OR OTIMP GULPQ RDMUR AIUTS

Untuk memperoleh plainteks kembali, penerima pesan harus mencari jumlah kolom dengan membagi panjang pesan dengan panjang kunci. Kemudian dia akan dapat menulis kembali pesan dalam kolom-kolom. Selanjutnya mengurutkan kembali kolom tersebut dengan melihat kata kunci.

## 2.4 Secure Hash Algorithm (SHA)

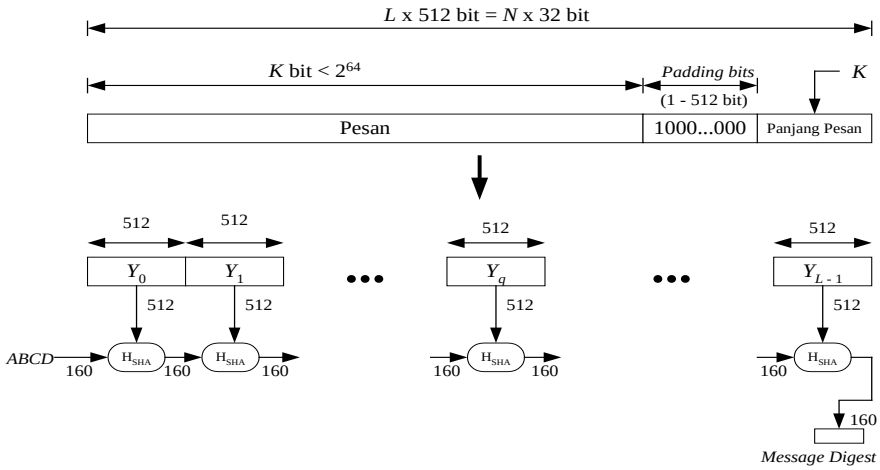
*Secure Hash Algorithm* adalah fungsi satu arah yang dirancang oleh NIST (*The National Institute of Standard and Technology*) bersama NSA (*National Security Agen*). SHA dibuat berdasarkan rancangan yang sama dengan MD4 yang dibuat oleh Profesor Ronald L. Rivest dari MIT. SHA dikatakan aman karena dirancang supaya secara matematis tidak memungkinkan untuk memperoleh pesan asli dari tanda tangan yang diberikan.

Fungsi satu arah sering dinamakan fungsi *hash* atau *message digest.* Disebut sebagai *message digest*, karena seolah-olah merupakan inti sari pesan. Selain itu, fungsi ini juga menghasilkan nilai yang lebih kecil dari pesan aslinya, sehingga sering disebut juga fungsi kompresi. Namun, hasil kompresi dari fungsi ini tidak dapat digunakan untuk memperoleh pesan aslinya kembali sehingga disebut fungsi satu arah.

Fungsi *hash* (H) beroperasi pada pesan (M) yang nilainya sembarang dan selalu menghasilkan nilai hash (h) yang selalu sama panjangnya, maka H(M) = h. Pada SHA, masukan pesan yang masuk sembarang panjangnya, tapi keluarannya selalu 160 bit. Sifat-sifat yang harus dimiliki fungsi hash adalah:

1. Diberikan M, mudah menghitung H(M) = h.
2. Diberikan h, tidak mudah untuk mendapatkan M sehingga H(M) = h.
3. Diberikan M, Sulit untuk bisa mendapatkan M' sehingga H(M) = H(M'). Jika diperoleh pesan M', maka hal ini disebut *collision* (tabrakan).
4. Sulit untuk mendapatkan dua pesan M dan M', sehingga H(M) = H(M').

SHA dapat menerima pesan masukan dengan ukuran maksimum $2^{64}$ bit (2.147.483.648 *gigabyte*) dan selalu menghasilkan *message digest* dengan ukuran tetap 160 bit. Proses pembuatan *message digest* dengan SHA digambarkan pada Gambar 2.2.




**Gambar 2.2 Pembuatan *message digest* dengan SHA**

Penjelasan langkah-langkah dari Gambar 2.2 adalah:

1. Penambahan Bit-bit Pengganjal

Pesan ditambah dengan bit pengganjal sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 512. Dengan demikian, panjang pesan setelah ditambahkan bit-bit pengganjal adalah 64 bit kurang dari kelipatan 512. SHA memperoses pesan dalam blok-blok yang berukuran 512.

Pesan dengan panjang 448 bit pun tetap ditambah dengan bit-bit pengganjal. Jika panjang pesan 448 bit, maka pesan tersebut ditambah dengan 512 bit menjadi 960 bit. Jadi, panjang bit-bit pengganjal adalah antara 1 sampai 512. Bit-bit pengganjal terdiri dari sebuah bit 1 diikuti dengan sisanya bit 0.

2. Penambahan Nilai Panjang Pesan Semula

Pesan yang telah diberi bit-bit pengganjal selanjutnya ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula. Setelah ditambah dengan 64 bit, panjang pesan sekarang menjadi 512 bit.

3. Inisialisai Penyangga MD

SHA membutuhkan 5 buah penyangga (*buffer*) yang masing-masing panjangnya 32 bit (MD5 hanya mempunyai 4 buah penyangga). Total panjang penyangga adalah 5 × 32 = 160 bit. Kelima penyangga menampung hasil antara nilai awal dan hasil akhir.

Kelima penyangga ini diberi nama *A*, *B*, *C*, *D*, dan *E*. Setiap penyangga diinisialisasi dengan nilai-nilai (dalam notasi HEX):

*A* = 67452301

*B* = EFCDAB89

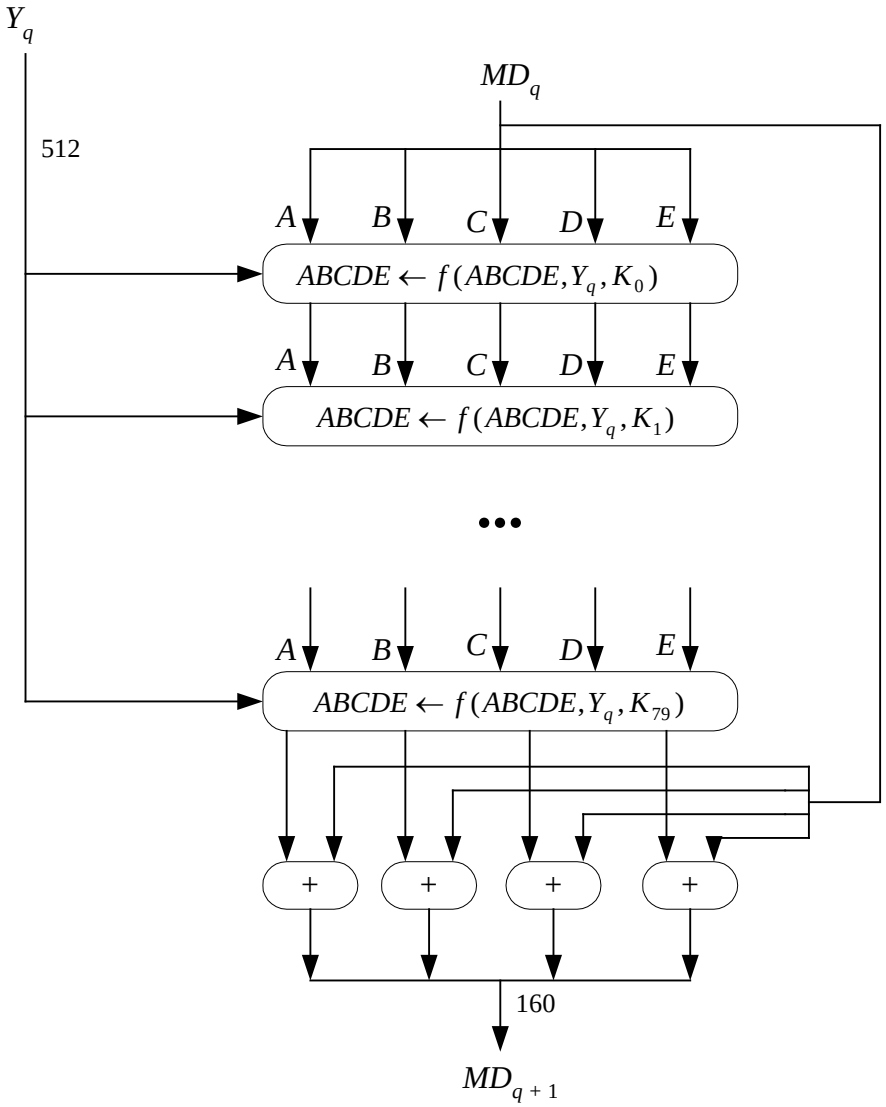
*C* = 98BADCFE

*D* = 10325476

*E* = C3D2E1F0.

4. Pengolahan Pesan dalam Blok Berukuran 512 bit.

Pesan dibagi menjadi *L* buah blok yang masing-masing panjangnya 512 bit ($Y_0$ sampai $Y_{L-1}$). Setiap blok 512-bit diproses bersama dengan penyangga MD menjadi keluaran 128-bit, dan ini disebut proses $H_{\text{SHA}}$. Gambaran proses $H_{\text{SHA}}$ diperlihatkan pada Gambar 2.3.




**Gambar 2.3 Pengolahan blok 512 bit**

Proses $H_{SHA}$ yang diperlihatkan pada Gambar 2.3 memiliki 80 buah putaran, dimana masing-masing putaran menggunakan bilangan penambah $K_t$, yaitu:

$K_t$ = 5A827999, untuk t = 0 sampai 19

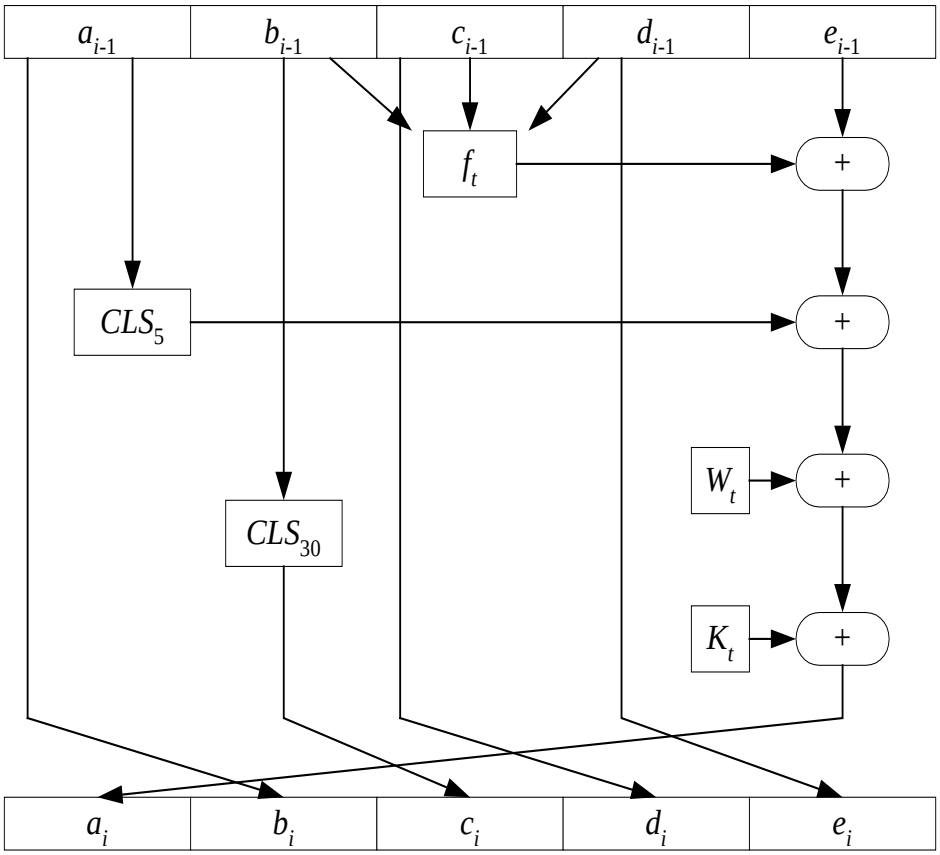
$K_t$ = 6ED9EBA1, untuk t = 20 sampai 39

$K_t$ = 8F1BBCDC, untuk t = 40 sampai 59

$K_t$ = CA62C1D6, untuk t = 60 sampai 79

Sedangkan $Y_q$ pada Gambar 2.3 menyatakan blok 512-bit ke-*q* dari pesan yang telah ditambah bit-bit pengganjal dan tambahan 64 bit nilai panjang pesan semula. $MD_q$ merupakan nilai *message digest* 160-bit dari proses $H_{SHA}$ ke-*q*. Pada awal proses, $MD_q$ berisi nilai inisialisasi penyangga MD.

Setiap putaran menggunakan operasi dasar yang sama (dinyatakan sebagai fungsi *f*). Operasi dasar SHA diperlihatkan pada Gambar 2.4.




**Gambar 2.4 Operasi dasar SHA dalam satu putaran (fungsi *f*)**

Operasi dasar SHA yang diperlihatkan pada Gambar 2.4 dapat ditulis dengan persamaan 2.4.

$$a, b, c, d, e \leftarrow (CLS_5(a) + f_t(b, c, d) + e + W_t + K_t), a, CLS_{30}(b), c, d \qquad (2.4)$$

dimana:

a, b, c, d, e = lima buah peubah penyangga 32-bit (berisi nilai penyangga *A*, *B*, *C*, *D*, *E*)

t = putaran, $0 \le t \le 79$

$f_t$ = fungsi logika

$CLS_s$ = *circular left shift* sebanyak *s* bit

$W_t$ = word 32-bit yang diturunkan dari blok 512 bit yang diproses

$K_t$ = konstanta penambah

$+$ = operasi penjumlahan modulo $2^{32}$

Fungsi $f$ pada Gambar 2.4 merupakan fungsi logika yang melakukan operasi logika bitwise yang berbeda setelah 20 putaran. Operasi logika yang dilakukan setiap putaran dapat dilihat pada Tabel 2.2.

**Tabel 2.2 Fungsi logika $f_t$ pada setiap putaran**

| **Putaran** | $f_t(b, c, d)$ |
|---|---|
| 0 .. 19 | $(b \wedge c) \vee (\sim b \wedge d)$ |
| 20 .. 39 | $b \oplus c \oplus d$ |
| 40 .. 59 | $(b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$ |
| 60 .. 79 | $b \oplus c \oplus d$ |

Catatan: operator logika AND, OR, NOT, XOR masing-masing dilambangkan dengan $\wedge, \vee, \sim, \oplus$

Nilai $W_1$ sampai $W_{16}$ berasal dari 16 word pada blok yang sedang diproses, sedangkan nilai $W_t$ berikutnya didapatkan persamaan:

$$W_t = W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3} \qquad (2.5)$$

Setelah putaran ke-79, *a*, *b*, *c*, *d*, dan *e* ditambahkan ke *A*, *B*, *C*, *D*, dan *E* dan selanjutnya algoritma memproses untuk blok data berikutnya ($Y_{q+1}$). Keluaran akhir dari algoritma SHA adalah hasil penyambungan bit-bit di *A*, *B*, *C*, *D*, dan *E*.

## 2.5 Metode Serangan Terhadap Kriptografi

Serangan (*attack*) adalah setiap usaha (*attempt*) atau percobaan yang dilakukan untuk menemukan kunci atau menemukan plainteks dari cipherteksnya. Orang yang melakukan serangan ini disebut kriptanalis.

Berdasarkan ketersediaan data yang ada, serangan terhadap kriptografi dapat diklasifikasikan menjadi 8 bagian (Rinaldi Munir, 2006), yaitu:

1. *Chipertext-only attack*

Kriptanalis memiliki beberapa cipherteks dari beberapa pesan, semuanya dienkripsi dengan algoritma yang sama. Pada serangan ini, kriptanalisis berusaha menemukan plainteks sebanyak mungkin atau menemukan kunci yang digunakan untuk mengenkripsi pesan.

2. *Known-plaintext attack*

Beberapa pesan yang formatnya terstruktur membuka peluang kepada kriptanalis untuk menemukan plainteks dari cipherteks yang bersesuaian.

3. *Chosen-plaintext attack*

Serangan jenis ini lebih hebat daripada *known-plaintext attack*, karena kriptanalis dapat memilih plainteks tertentu untuk dienkripsikan, yaitu plainteks-plainteks yang lebih mengarahkan penemuan kunci.

4. *Adaptive-chosen-plaintext attack*

Kasus khusus dari jenis serangan nomor *Chosen-plaintext attack* . Misalnya, kriptanalis memilih blok plainteks yang besar, lalu dienkripsi, kemudian memilih blok lainnya yang lebih kecil berdasarkan hasil serangan sebelumnya.

5. *Chosen-ciphertext attack*

Kriptanalis memiliki akses terhadap cipherteks yang didekripsi (misalnya terhadap mesin elektronik yang melakukan dekripsi secara otomatis).

6. *Chosen-text attact*

Jenis serangan gabungan dari *Chosen-plaintext attack* dan *Chosen-ciphertext attack.*

7. *Chosen-key attack*

Kriptanalis memiliki pengetahuan mengenai hubungan antara kunci-kunci yang berbeda, dan memilih kunci yang tepat untuk mendekripsi plainteks.

8. *Rubber-hose cryptanalysis*

Kriptanalis mengancam, mengirim surat gelap, atau melakukan penyiksaan sampai orang yang memegang kunci memberinya kunci untuk mendekripsi plainteks.

## 2.6 Keamanan Algoritma Kriptografi

Sebuah algoritma kriptografi dikatakan aman (*computationally secure*) bila ia memenuhi tiga kriteria berikut (Rinaldi Munir, 2006) :

1. Persamaan matematis yang menggambarkan operasi algoritma kriptografi sangat kompleks sehingga algoritma tidak mungkin dipecahkan secara analitik.
2. Biaya untuk memecahkan cipherteks melampaui nilai informasi yang terkandung di dalam cipherteks tersebut.
3. Waktu yang diperlukan untuk memecahkan cipherteks melampaui lamanya waktu informasi tersebut harus dijaga kerahasiaannya

## 2.7 Serangan terhadap Sistem Pengamanan Data

Serangan terhadap sistem pengamanan gabungan kedua algoritma yang akan dibangun ini memilki 2 bagian, yaitu serangan mencari *collision* nilai *hash* dengan cipherteks yang berbeda, dan yang kedua, serangan dengan mengubah cipherteks yang asli menjadi plainteks.

1. Serangan mencari *collision* dengan cipherteks yang berbeda

*Collision* adalah suatu kondisi dimana 2 pesan yang berbeda memilki nilai *hash* yang sama dengan menggunakan fungsi *hash* yang sama juga. Pada umumnya, serangan dengan pencarian *collision* dapat menggunakan serangan secara acak (*brute force*). Akan tetapi, dengan teknologi yang ada saat ini serangan ini membutuhkan komputasi waktu yang bertahun-tahun.

Pada bulan Februari 2005, tiga orang peneliti dari Cina, Xiaoyun Wang, Yiqun Lisa Yin, dan Hongbo Yu, mempublikasikan bahwa mereka telah berhasil menemukan cara untuk melakukan pencarian *collision* fungsi SHA-1 dengan kompleksitas $2^{69}$. Ketiga orang tersebut adalah sebuah tim riset yang sudah sangat bereputasi karena sebelumnya dapat mencari *collision* pada SHA-0 dengan kompleksitas $2^{39}$. Walaupun demikian kompleksitas operasi yang dibutuhkan masih sangat besar juga (dengan menggunakan komputer yang sama dengan contoh sebelumnya, masih akan dibutuhkan waktu sekitar 170.000 tahun, waktu yang tidak mungkin bagi manusia).

Jika *collision* ditemukan, maka kemungkinan pesan yang diterima oleh si penerima bukan pesan yang asli, karena 2 pesan yang berbeda memiliki nilai *hash* yang sama.

2. Serangan dengan mengubah cipherteks asli menjadi plainteks

Serangan ini merupakan serangan dengan mengembalikan cipherteks yang asli menjadi plainteks yang berisi informasi, dengan syarat cipherteks yang asli diubah kembali dengan SHA menghasilkan nilai *hash* 2 yang sama dengan nilai *hash* 1.

Jika syarat tersebut terpenuhi, maka cipherteks asli dapat dipecahkan dengan melakukan analisis frekuensi, dimana penyerang dengan mudah menyadari bahwa cipher transposisilah yang digunakan, karena frekuensi karakter pada cipherteks menunjukkan pola yang sama dengan frekuensi karakter pada plainteks. Dengan

melakukan penyusunan ulang tertentu atau yang sering disebut anagram, maka penyerang dapat mengetahu plainteks.

Penyerangan kedua ini lebih sulit, dimana penyerang harus menemukan metode yang sama untuk menghasilkan nilai *hash* yang sama untuk meyakinkan cipherteks merupakan cipherteks asli. Kemudian menemukan metode kedua untuk mengubah cipherteks menjadi plainteks.