# BERBAGAI KASUS PENYERANGAN TERHADAP KRIPTOGRAFI

Muara P. Sipahutar – NIM : 13503064

*Program Studi Teknik Informatika, Institut Teknologi Bandung*
*Jl. Ganesha 10, Bandung*
E-mail : if13064@students.if.itb.ac.id

## ABSTRAK

Terdapat berbagai algoritma enkripsi di dunia ini. Seiring dengan berkembangnya algoritma tersebut, berkembang juga cara untuk memecahkannya. Orang yang memahami kriptografi terus menerus muncul dari waktu ke waktu. Beberapa dari mereka mengambil jalan untuk menjadi sang pembuat enkripsi data(cryptografer) sedangkan yang lain mengambil peran yang sesat seperti penyadap dan kriptanalis.

Serangan-serangan terhadap kriptografi mulai bermunculan satu persatu. Makin lama, serangan ini semakin kuat dan dibutuhkan algoritma yang lebih robust untuk menahannya. Walau pun begitu, tetap saja seiring dengan dipelajarinya algoritma enkripsi tersebut, banyak algoritma enkripsi yang terpecahkan.

Makalah ini membahas tentang berbagai serangan terhadap kriptografi serta kasus-kasus serangan yang pernah terjadi terhadap berbagai algoritma enkripsi tersebut, dan juga bagaimana cara kriptanalis mendekripsikan ciphertext menjadi plaintext yang telah melalui berbagai proses algoritma enkripsi tersebut.

***Kata kunci**: cryptografer,* kriptanalisis, serangan, algoritma, enkripsi, dekripsi.

## 1. Pendahuluan

Proses pengiriman informasi secara rahasia telah dilakukan dari masa ke masa. Sejak 4000 tahun yang lalu Bangsa Mesir telah membuat *hyroglyph* untuk mengkode informasi rahasia mereka.

Selain itu, 400 tahun yang lalu, tentara Sparta di Yunani juga menggunakan alat yang bernama *Scytale* untuk mengirimkan pesan selama perang. *Scytale* ini berupa pita panjang dari daun *papyrus* ditambah sebatang silinder. Pesan ditulis horizontal baris per baris. Bila pita dilepaskan maka huruf-huruf di dalamnya telah tersusun berbentuk pesan rahasia. Untuk membaca pesan, penerima melilitkan kembali silinder yang diameternya sama dengan diameter silinder pengirim. Tentunya diameternya harus tepat sehingga pesan terbaca dengan baik.

Pada abad 17, Queen Mary juga menggunakan pesan rahasia yang terenkripsi. Pesan tersebut berisi rencana pembunuhan Ratu Elizabeth I.

Bukti penggunaan kriptografi yang paling jelas adalah pada Perang Dunia II di mana pemerintah Nazi Jerman membuat mesin enkripsi yang dinamakan Enigma. Enigma cipher berhasil dipecahkan oleh pihak Sekutu dan keberhasilan memecahkan Enigma sering dikatakan sebagai faktor yang memperpendek perang dunia kedua.

Dari bukti-bukti yang dikemukakan di atas terdapat suatu perkembangan kriptografi. Dari yang semula hanya menggunakan kertas dan alat tulis saja, seiring dengan berkembangnya waktu, mesin untuk mengenkripsi pesan pun dijadikan.

Sampai sekarang kriptografi digunakan dalam kehidupan sehari-hari. Contohnya seperti sinyal yang ditransmisikan dalam percakapan dengan handphone, pengenkripsian data PIN kartu ATM dari mesin ATM ke komputer bank, atau pun penyimpanan data pada suatu disk storage (contohnya *hard disk*) disimpan dalam bentuk cipherteks.

Sejarah kriptografi pararel dengan sejarah kriptanalisis (*cryptanalysis*), yaitu bidang ilmu dan seni untuk memecahkan cipherteks.

Teknik kriptanalisis sudah ada sejak abad ke-9. Adalah seorang ilmuwan Arab pada abad IX bernama Abu Yusuf Yaqub Ibnu Ishaq Ibnu As-

Sabbah Ibnu 'Omran Ibnu Ismail Al-Kindi, atau yang lebih dikenal sebagai Al-Kindi yang menulus buku tentang seni memecahkan kode. Dalam buku yang berjudul '*Risalah fi Istikhraj al-Mu'amma* (*Manuscript for the Deciphering Cryptographic Messages*)'.

Al –Kindi menulis naskah yang berisi cara-cara untuk menguraikan kode-kode rahasia. Ia juga mengklasifikasikan sandi rahasia itu serta menjelaskan ilmu fonetik Arab dan sintaksisnya. Yang paling penting lagi, dalam bukunya ini ia mengenalkan penggunaan beberapa teknik statistika untuk memecahkan kode-kode rahasia.

Apa yang dilakukan oleh Al-Kindi di dalam kriptanalisis dikenal dengan nama teknik analisis frekuensi, yaitu teknik untuk memecahkan cipherteks berdasarkan frekuensi kemunculan karakter di dalam pesan dan kaitannya dengan frekuensi kemunculan karakter di dalam alfabet.

Keseluruhan point dari kriptografi adalah menjaga kerahasiaan plainteks dari penyadap (*eavesdropper*) atau kriptanalis(*cryptanalist*). Penyadap bisa juga merangkap sebagai seorang kriptanalis.. Penyadap berusaha mendapatkan data yang digunakan untuk kegiatan kriptanalis. Sedangkan kriptanalais berusaha mengungkap plainteks atau kunci dari data yang disadap. Kriptanalis juga dapat menemukan kelemahan dari sistem kriptografi yang pada akhirnya mengarah untuk menemukan kunci dan mengungkap plainteks.

Yang dimaksud dengan serangan(*attack*) adalah setiap usaha (*attempt*) atau percobaan yang dilakukan oleh kriptanalis untuk menemukan kunci atau menemukan plainteks dari cipherteksnya.

Dalam membahas setiap serangan terhadap kriptografi, kita selalu mengasumsikan kriptanalis mengetahui algoritma kriptografi yang digunakan, sehingga satu-satunya keamanan sistem kriptografi terletak sepenuhnya pada kunci. Hal ini didasarkan pada Prinsip Kerckhoff(1883) yang berbunyi "Semua algoritma kriptografi harus publik, hanya kunci yang rahasia."

Dengan berdasarkan prinsip ini, tentu saja para kriptanalis berusaha mempelajari setiap algoritma kriptografi guna menemukan plainteks dari cipherteksnya dengan menggunakan semua source yang ada. Dalam hal ini, kriptanalis mungkin berhasil memecahkannya. Masalahnya tentunya hanya pada waktu dan kemahiran kriptanalis dalam menganalisa algoritma kriptografi, seperti yang dilakukan pihak Sekutu saat memecahkan Enigma cipher.

## 2. Tipe Serangan Terhadap Kriptografi

Berdasarkan keterlibatan penyerangnya dalam komunikasi, serangan dapat dibagi atas dua macam, yaitu :

1. Serangan pasif (*passive attack*)
2. Serangan aktif (*active attack*)

Berdasarkan ketersediaan data yang ada serangan terhadap kriptografi diklasifikasikan menjadi :

1. *Ciphertext-only attack*
2. *Known-plaintext attack*
3. *Chosen-plaintext attack*
4. *Adaptive-chosen-plaintext attack*
5. *Chosen-ciphertext attack*
6. *Chosen-text attack*

Berdasarkan teknik yang digunakan dalam menemukan kunci, serangan dapat dibagi menjadi :

1. Exhaustive attack / brute force attack
2. Analytical attack

Selain itu ada beberapa jenis serangan lain yang tidak termasuk klasifikasi di atas, yaitu :

1. Related-key attack
2. Rubber-hose cryptanalysis

### *2.1. Serangan Pasif (Passive Attack)*

Pada jenis serangan ini, penyerang tidak terlibat dalam komunikasi antara pengirim dan penerima, namun penyerang menyadap semua pertukaran pesan antara kedua entitas tersebut.

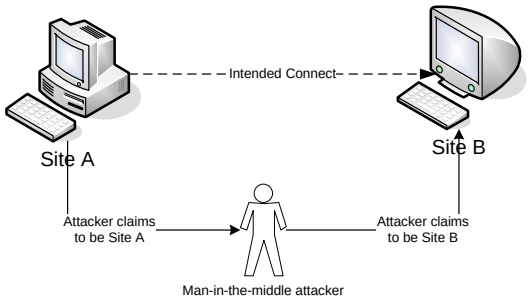
Tujuan dari serangan pasif adalah untuk mendapatkan sebanyak mungkin informasi yang digunakan utnuk kriptanalisis. Beberapa metode penyadapan data antara lain :

a. *Wiretapping* : penyadap mencegat data yang ditransmisikan pada saluran kabel komunikasi dengan menggunakan sambungan perangkat keras.
b. *Electromagnetic eavesdropping* : penyadap mencegat data yang ditransmisikan melalui saluran wireless, misalnya radio dan microwave.
c. *Accoustic eavesdropping* : menangkap gelombang suara yang dihasilkan oleh suara manusia.

### *2.2. Serangan Aktif (Active Attack)*

Pada jenis serangan ini, penyerang mengintervensi komunikasi dan ikut mempengaruhi sistem untuk keuntungan dirinya. Misalnya penyerang mengubah aliran pesan seperti menghapus sebagian cipherteks, mengubah cipherteks, menyisipkan potongan cipherteks palsu, me-*replay* pesan lama, mengubah informasi yang tersimpan, dan sebagainya.

Serangan yang termasuk jenis serangan aktif adalah man-in-the-middle attack.




**Gambar 1 : Man-in-the-middle attack**

Pada serangan ini, penyerang mengintersepsi komunikasi antara dua pihak yang berkomunikasi dan kemudian ”menyerupai” salah satu pihak dengan cara bersikap seolah-olah ia adalah pihak yang berkomunikasi (pihak yang lainnya tidak menyadari kalau ia berkomunikasi dengan pihak yang salah).

Tujuan dari serangan ini adalah untuk mendapatkan informasi berharga seperti kunci atau nilai rahasia lainnya. Caranya, penyerang memutus komunikasi antara dua pihak lalu menempatkan dirinya di antara keduanya. Seluruh informasi dari pengirim dan penerima akan diterima oleh penyerang. Penyerang mengubah pesan yang dikirim oleh pengirim sebenarnya dengan pesannya sendiri, lalu mengirimkannya ke penerima. Penerima mengira informasi tersebut berasal dari pengirim sebenarnya.

### *2.3. Chipertext-only Attack*

Kriptanalis memiliki beberapa cipherteks dari beberapa pesan, semuanya dienkripsi dengan algoritma yang sama.

Tugas kriptanalis adalah menemukan plainteks sebanyak mungkin atau menemukan kunci yang digunakan untuk mengenkripsi pesan. (C = ciphertext, P = plaintext, E = enkripsi, D = dekripsi, k = keys, kunci)

Diberikan :
$C_1 = E_k(P_1)$, $C_2 = E_k(P_2)$,..., $C_i = E_k(P_i)$

Deduksi :
$P_1, P_2$,..., $P_i$ atau $k$ untuk mendapatkan $P_{i+1}$ dari $C_{i+1} = E_k(P_{i+1})$.

### *2.4. Known-plaintext Attack*

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

Misalnya : *From* dan *To* di dalam *e-mail*, ”Dengan hormat” pada surat resmi, #*include*, *program*, di dalam *source code*

Diberikan :
$P_1, C_1 = E_k(P_1)$, $P_2, C_2 = E_k(P_2)$,... $P_i, C_i = E_k(P_i)$

Deduksi :
$k$ untuk mendapatkan $P_{i+1}$ dari $C_{i+1} = E_k(P_{i+1})$

### *2.5. 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.

Diberikan :
$P_1, C_1 = E_k(P_1)$, $P_2, C_2 = E_k(P_2)$,...
$P_i, C_i = E_k(P_i)$ di mana kriptanalis dapat memilih di antara $P_1$, $P_2$, ,... $P_i$

Deduksi :
$k$ untuk mendapatkan $P_{i+1}$ dari $C_{i+1} = E_k(P_{i+1})$

### *2.6. Adaptive-chosen-plaintext Attack*

Kasus khusus dari jenis serangan nomor 3 di atas. Misalnya, kriptanalis memilih blok plainteks yang besar, lalu dienkripsi, kemudian blok lainnya yang lebih kecil berdasarkan hasil serangan sebelumnya, begitu seterusnya.

### *2.7. Chosen-ciphertext Attack*

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

Diberikan :
$C_1, P_1 = D_k(C_1)$, $C_2, P_2 = D_k(C_2)$,..., $C_i, P_i = D_k(C_i)$

Deduksi :
*k* (yang mungkin diperlukan untuk mendekripsi pesan pada waktu yang akan datang).

Jenis serangan ini dipakai pada algoritma kunci-publik.

### *2.8. Chosen-text Attack*

Gabungan *chosen-plaintext attack* dan *chosen-ciphertext attack*.

### *2.9. Exhaustive Attack (Brute Force Attack)*

Percobaan yang dibuat untuk mengungkap plainteks atau kunci dengan mencoba semua kemungkinan kunci (*trial and error*).
Asumsi yang digunakan :
a. Kriptanalis mengetahui algoritma kriptografi.
b. Kriptanalis memiliki sebagian plainteks dan cipherteks yang bersesuaian.

Caranya : plainteks yang diketahui dienkripsikan dengan setiap kemungkinan kunci, dan hasilnya dibandingkan dengan cipherteks yang bersesuaian. Jika hanya cipherteks yang tersedia, cipherteks tersebut didekripsi dengan setiap kemungkinan kunci dan plainteks hasilnya diperiksa apakah mengandung makna.

Misalkan sebuah system kriptografi membutuhkan kunci yang panjangnya 8 karakter, karakter dapat berupa angka (10 buah), huruf (26 huruf besar dan 26 huruf kecil), maka jumlah kunci yang harus dicoba adalah sebanyak :
$62 \times 62 \times 62 \times 62 \times 62 \times 62 \times 62 \times 62 = 62^8$ buah.

Secara teori, serangan secara *exhaustive* ini dipastikan berhasil mengungkap plainteks tetapi dalam waktu yang sangat lama.

| **Ukuran kunci** | **Jumlah kemungkinan kunci** | **Lama waktu untuk $10^6$ percobaan per detik** | **Lama waktu untuk $10^{12}$ percobaan per detik** |
|---|---|---|---|
| 16 bit | $2^{16} = 65536$ | 32.7 milidetik | 0.0327 mikrodetik |
| 32 bit | $2^{32} = 4.3 \times 10^9$ | 35.8 menit | 2.15 milidetik |
| 56 bit | $2^{56} = 7.2 \times 10^{16}$ | 1142 tahun | 10.01 jam |
| 128 bit | $2^{128} = 4.3 \times 10^{38}$ | $5.4 \times 10^{24}$ tahun | $5.4 \times 10^{18}$ tahun |

**Tabel 1 : Waktu yang diperlukan untuk *exhaustive key search***

Untuk menghadapi serangan ini, perancang kriptosistem (kriptografer) harus membuat kunci yang panjang dan tidak mudah ditebak.

### *2.10. Analytical Attack*

Pada jenis serangan ini, kriptanalis tidak mencoba-coba semua kemungkinan kunci tetapi menganalisis kelemahan algoritma kriptografi untuk mengurangi kemungkinan kunci yang tidak mungkin ada.

Analisis dilakukan dengan memecahkan persamaan-persamaan matematika (yang diperoleh dari definisi suatu algoritma kriptografi) yang mengandung peubah-peubah yang merepresentasikan plainteks atau kunci.

Asumsi yang digunakan :
Kriptanalis mengetahui algoritma kriptografi.
Untuk menghadapi serangan ini, kriptografer harus membuat algoritma kriptografi yang kompleks sedemikian sehingga plainteks merupakan fungsi matematika dari cipherteks dan kunci yang cukup kompleks, dan tiap kunci merupakan fungsi matematika dari cipherteks dan plainteks yang kompleks juga.

### *2.11. Related-key Attack*

Kriptanalis memiliki cipherteks yang dienkripsi dengan dua kunci yang berbeda. Kriptanalis tidak mengetahui kedua kunci tersebut namun ia mengetahui hubungan antara kedua kunci, misalnya mengetahui kedua kunci hanya berbeda 1 bit.

### *2.12. Rubber-hose Cryptanalysis*

Ini mungkin jenis serangan yang paling ekstrim dan paling efektif. Penyerang mengancam,

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

## 3. Kompleksitas Serangan

Kompleksitas serangan dapat diukur dengan beberapa cara :
a. Kompleksitas data (*data complexity*)
Jumlah data yang dibutuhkan sebagai masukan untuk serangan. Semakin banyak data yang dibutuhkan untuk melakukan serangan, berarti semakin bagus algoritma kriptografi tersebut.

b. Kompleksitas waktu (*time complexity*)
Waktu yang dibutuhkan untuk melakukan serangan. Ini disebut juga factor kerja (*work factor*). Semakin lama waktu yang dibutuhkan untuk melakukan serangan, berarti semakin bagus algoritma kriptografi tersebut.

c. Kompleksitas ruang memori (*space/storage complexity*)
Jumlah memori yang dibutuhkan untuk melakukan serangan. Semakin banyak memori yang dibutuhkan untuk melakukan serangan, berarti semakin bagus algoritma kriptografi tersebut.

## 4. Aspek Keamanan yang Disediakan oleh Kriptografi

### *4.1. Kerahasiaan (confidentiality)*

Kerahasiaan adalah layananyang digunakan untuk menjaga isi pesan dari siapa pun yang tidak berhak untuk membacanya. Di dalam kriptografi, layanan ini umumnya direalisasiakn dengan cara menyandikan pesan menjadi bentuk yang tidak dapat dimengerti. Misalnya pesan ”Harap datang pukul 8” disandikan menjadi ”TrxC#45motyptre!%”. Istilah lain yang serupa dengan *confidentiality* adalah *secrecy* dan *privacy*.

### *4.2. Integritas data (data integrity)*

Integritas data adalah layanan yang menjamin bahwa pesan masih asli/utuh atau belum pernah dimanipulasi selama pengiriman. Dengan kata lain, aspek keamanan ini dapat diungkapkan dengan pertanyaan : ”Apakah pesan yang diterima masih asli atau tidak mengalami perubahan (modifikasi)?”. Istilah lain yang serupa degnan *data integrity* adalah otentifikasi pesan *(message authentification).*

Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi pesan oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubtitusian data lain ke dalam pesan yang sebenarnya. Di dalam kriptografi, layanan ini direalisasikan dengan menggunakn tanda-tangan digital (*digital signature*). Pesan yang telah ditandatangani menyiratkan bahwa pesan yang dikirim adalah asli.

### *4.3. Otentikasi (Authentification)*

Otentikasi adalah layanan yang berhubungan dengan identifikasi, baik mengidentifikasi kebenaran pihak-pihak yang berkomunikasi (*user authentification* atau *entity authentification)* maupun mengidentifikasi kebenaran sumber pesan (*data origin autentification*).

Dua pihak yang saling berkomunikasi harus dapat mengotentikasi satu sama lain sehingga ia dapat memastikan sumber pesan. Pesan yang dikirim melalui saluran komunikasi juga harus diotentikasi asalnya. Dengan kata lain, aspek keamanan ini dapat diungkapkan dengan pertanyaan: ”Apakah pesan yang diterima benar-benar berasal dari pengirim yang benar?”.

Otentikasi sumber pesan secara implisit juga memberikan kepastian integritas data, sebab jika pesan telah dimodifikasi berarti sumber pesan sudah tidak benar. Oleh karena itu, layanan integritas data selalu dikombinasikan dengan layanan otentikasi sumber pesan. Di dalam kriptografi, layanan ini direalisasikan dengan menggunakn tanda tangan digital (*digital signature*). Tanda tangan digital menyatakan sumber pesan.

### *4.4. Nirpenyangkalan (non-repudiation)*

Nirpenyangkalan adalah layanan untuk mencegah entitas yang berkomunijkasi melakukan penyangkalan, yaitu pengirim pesan menyangkal melakukan pengiriman atau penerima pesan menyangkal telah menerima pesan. Sebagai contoh, pengirim pesan memberi

otoritas kepada penerima pesan untuk melakukan pembelian, namun kemudian ia menyangkal telah memberikan otoritas tersebut. Contoh lainnya, seorang pemilik emas mengajukan tawaran kepada toko mas bahwa ia akan menjual emasnya. Tetapi, tiba-tiba harga emas turun drastis, lalu ia membantah telah mengajukan tawaran menjual emas. Dalam hal ini, pihak toko emas perlu prosedur nirpenyangkalan untuk membuktikan bahwa pemilik emas telah melakukan kebohongan.

## 5. Keamanan Algoritma Kriptografi

Sebuah algoritma kriptografi dikatakan aman (computationally secure) bila ia memenuhi tiga kriteria berikut :
a. Persamaan matematis yang menggambarkan operasi algoritma kriptografi sangat kompleks sehingga algoritma tidak mungkin dipecahkan secara analitik.

b. Biaya untuk memecahkan cipherteks melampaui nilai informasi yang terkandung di dalam cipherteks tersebut.

c. Waktu yang diperlukan untuk memecahkan cipherteks melampaui lamanya waktu informasi tersebut harus dijaga kerahasiaannya.

## 6. Serangan dan Pemecahan Terhadap Berbagai Algoritma Kriptografi

### *6.1. Kriptanalisis Caesar Cipher*

*Caesar Cipher* mudah dipecahkan dengan metode *exhaustive key search* karena jumlah kuncinya sangat sedikit (hanya ada 26 kunci).

Misalkan kriptanalis menemukan potongan cipherteks (disebut juga cryptogram) XMZVH. Diandaikan kriptanalis mengetahui bahwa plainteks disusun dalam bahasa Inggris dan algoritma kriptografi yang digunakan adalah *Caesar Cipher*. Untuk memperoleh plainteks, lakukan dekripsi mulai dari kunci yang terbesar, 25, sampai kunci yang terkecil, 1. Periksa apakah dekripsi menghasilkan pesan yang mempunyai makna. Ketika diperiksa, ada satu kunci yang menghasilkan kata yang benar-benar bermakna.

| Kunci(k) Ciphering | 'Pesan' hasil dekripsi | Kunci(k) Ciphering | 'Pesan' hasil dekripsi |
|---|---|---|---|
| 0 | XMZVH | 13 | KZMIU |
| 25 | YNAWI | 12 | LANJV |
| 24 | ZOBXJ | 11 | MBOKW |
| 23 | APCYK | 10 | NCPLX |
| 22 | BQDZL | 9 | ODQMY |
| 21 | CREAM | 8 | PERNZ |
| 20 | DSFBN | 7 | QFSOA |
| 19 | ETGCO | 6 | RGTPB |
| 18 | FUHDP | 5 | SHUQC |
| 17 | GVIEQ | 4 | TIVRD |
| 16 | HWJFR | 3 | UJWSE |
| 15 | IXKGS | 2 | VKXTF |
| 14 | JYLHT | 1 | WLYUG |

**Tabel 2 : Contoh *exhaustive key search* terhadap cipherteks XMZVH**

Kadang-kadang satu kunci yang potensial menghasilkan pesan yang bermakna tidak selalu satu buah. Untuk itu, kita membutuhkan informasi lainnya, misalnya konteks pesan tersebut atau mencoba mendekripsi potongan cipherteks lain untuk memperoleh kunci yang benar.

### *6.2. Kriptanalisis Vigènere Cipher*

Untuk memecahkan *Vigènere Cipher*, cukup menentukan panjang kuncinya. Jika periode kunci diketahui dan tidak terlalu panjang, maka kunci dapat ditentukan dengan menulis program komputer untuk melakukan *exhaustive key search*.

Sebagai contoh, jika diberikan cipherteks yang dihasilkan dengan *Vigènere Cipher* :

TGCSZ GEUAA EFWGQ AHQMC

dan diperoleh informasi bahwa panjang kunci adalah 3 huruf dan plainteks ditulis dalam Bahasa Inggris, maka running program dengan mencoba semua kemungkinan kunci yang panjangnya tiga huruf, lalu periksa apakah hasil dekripsi dengan kunci tersebut menyatakan kata yang berarti. Cara ini membutuhkan usaha percobaan sebanyak $26^6$ kali. Semakin panjang kunci (semakin besar periode), semakin banyak usaha percobaan yang harus dilakukan.

Metode Kasiski membantu menemukan panjang kunci. Friedrich Kasiski adalah orang yang pertama kali memecahkan *Vigènere Cipher* pada tahun 1863, tetapi Charles Babbage telah

mengembangkan cara yang serupa pada tahun 1854. Metode Kasiski memanfaatkan keuntungan bahwa bahasa Inggris tidak hanya mengandung perulangan huruf tetapi juga perulangan pasangan huruf atau tripel huruf, seperti TH, THE, dan sebagainya. Perulangan kelompok huruf ini ada kemungkinan menghasilkan kriptogram yang berulang. Perhatikan contoh di bawah ini :

Plainteks
CRYPTO IS SHORT FOR CRYPTOGRAPHY
Kunci
abcdab cd abcda bcd abcdabcdabcd
Cipherteks
**CSASTP** KV SIQUT GQU **CSASTP**IUAQJB

Pada contoh ini, CRYPTO dienkripsi menjadi kriptogram yang sama, yaitu CSATP. Tetapi kasus seperti ini tidak selalu demikian, misalnya pada contoh berikut ini :

Plainteks
CRYPTO IS SHORT FOR CRYPTOGRAPHY
Kunci
abcdef ab cdefa bcd efabcdefabcd
Cipherteks
**CSASXT** IT UKWST GQU **CWYQVR**KWAQJB

Pada contoh di atas, CRYPTO tidak dienkripsi menjadi kriptogram yang sama. Secara intuitif kita dapat membuat argumentasi bahwa jika jarak duab buah *string* yang berulang didalam plainteks merupakan kelipatan dari panjang kunci, maka *string* yang sama tersebut akan muncul menjadi kriptogram yang sama pula di dalam cipherteks. Hal ini ditunjukkan pada contoh pertama, yang mana kuncinya adalah abcd dengan panjang 4, sementara *string* CRYPTO muncul berulang di dalam plainteks dan jarak antara dua buah kemunculan ini adalah kelipatan 4. Sedangkan pada contoh kedua kuncinya abcdef dengan panjang 6 tetapi 6 bukan kelipatan 16.

Untuk menentukan panjang kunci, langkah-langkahnya adalah sebagai berikut :

i. Kriptanalis menghitung semua kriptogram yang berulang di dalam cipherteks (pesan yang panjang biasanya mengandung kriptogram yang berulang). Kemudian, jarak antara kriptogram yang berulang dihitung.

ii. Kriptanalis menghitung semua faktor (pembagi) dari jarak tersebut. Faktor pembagi menyatakan panjang kunci yang mungkin. Tentukan irisan dari himpunan faktor pembagi tersebut. Nilai yang muncul di dalam irisan menyatakan angka yang muncul pada semua faktor pembagi dari jarak-jarak tersebut. Nilai tersebut mungkin adalah panjang kunci. Hal ini karena *string* yang berulang dapat muncul bertindihan (*coincidence*), tetapi sangat mungkin terjadi huruf yang sama dienkripsi dengan huruf kunci yang sama. Huruf-huruf kunci diulang pada kelipatan panjang kunci, sehingga jarak yang ditemukan pada langkah 1 sangat mungkin merupakan kelipatan panjang kunci.

Misalkan kriptanalis memperoleh cipherteks :

**DYDUXRM**HTVDV**NQD**QNW**DYDUXRM**HARTJGW**NQD**

Kriptogram yang berulang adalah DYUDUXRM dan NQD. Jarak antara dua buah perulangan adalah 18. Semua faktor pembagi 18 adalah {18,9,6,3,2} (1 tidak dimasukkan) yang menyiratkan bahwa panjang kunci kemungkinan adalah faktor-faktor tersebut. Jarak antara dua buah perulangan NQD adalah 20. Semua faktor pembagi 20 adalah {20,9,6,3,2}. Irisan dari kedua buah himpunan tersebut adalah 2 sehingga kita menyimpulkan panjang kunci kemungkinan terbesar adalah 2.

### *6.3. Serangan Terhadap Cipher Aliran*

Serangan yang dapat dilakukan oleh kriptanalis terhadap cipher aliran adalah :

a. *Known-plaintext attack*
Misalkan kriptanalis memiliki potongan plainteks (P) dan cipherteks (C) yang berkoresponden, maka ia dapat menemukan bagian bit aliran-kunci (K) yang berkoresponden dengan meng-XOR-kan bit-bit plainteks dan cipherteks (Θ = XOR) :

$$\begin{aligned} P \Theta C &= P \Theta (P \Theta K) \\ &= (P \Theta P) \Theta K \\ &= 0 \Theta K \\ &= K \end{aligned}$$

b. *Ciphertext-only attack*
Serangan ini terjadi jika keystream yang sama digunakan dua kali terhadap potongan plainteks yang berbeda.

Serangan semacam ini disebut juga *keystream reuse attack*.

Misalkan kriptanalis memiliki dua potongan cipherteks berbeda ($C_1$ dan $C_2$) yang dienkripsi dengan bit-bit kunci yang sama. Ia meng-XOR-kan kedua cipherteks tersebut dan memperoleh dua buah plainteks yang ter-XOR satu sama lain :

$$C_1 \Theta C_2 = (P_1 \Theta K) \Theta (P_2 \Theta K)$$
$$= (P_1 \Theta P_2) \Theta (K \Theta K)$$
$$= (P_1 \Theta P_2) \Theta 0$$
$$= (P_1 \Theta P_2)$$

Jika salah satu dari $P_1$ atau $P_2$ diketahui atau dapat diterka, maka XOR-kan salah satu plainteks tersebut dengan cipherteksnya untuk memperoleh bit-bit kunci K yang berkoresponden:

$$P_1 \Theta C_1 = P_1 \Theta (P_1 \Theta K) = K$$

Selanjutnya $P_2$ dapat diungkap dengan kunci K ini.

Jika $P_1$ atau $P_2$ tidak diketahui, dua buah plainteks yang ter-XOR satu sama lain ini dapat diketahui dengan menggunakan nilai statistic dari pesan. Misalnya dalam teks Bahasa Inggris, dua buah spasi ter-XOR, atau satu spasi dengan huruf 'e' yang paling sering muncul, dsb. Kriptanalis cukup cerdas untuk mendeduksi kedua plainteks tersebut.

Seharusnya pengguna cipher aliran harus mempunyai bit-bit kunci yang tidak dapat diprediksi sehingga mengetahui sebagian dari bit-bit kunci tidak memungkinkan kriptanalis dapat mendeduksi bagian sisanya.

c. *Flip-bit attack*
Serangan ini tidak bertujuan menemukan kunci atau mengungkap plainteks dari cipherteks, tetapi mengubah bit cipherteks tertentu sehingga hasil dekripsinya berubah. Pengubahan dilakukan dengan membalikkan (flip) bit tertentu (0 menjadi 1, atau 1 menjadi 0).

Pengubah pesan tidak perlu mengetahui kunci, ia hanya perlu mengetahui posisi pesan yang diminati saja.

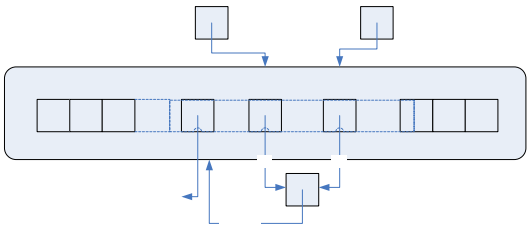
Serangan semacam ini memanfaatkan karakteristik cipher aliran yang sudah disebutkan di atas, bahwa kesalahan 1-bit pada cipherteks hanya menghasilkan kesalahan 1-bit pada plainteks hasil dekripsi.

### *6.4. Serangan Terhadap RC4*

RC4 (atau ARCFOUR) adalah cipher aliran yang digunakan secara luas pada sistem keamanan seperti protocol SSL (Secure Socket Layer). Algoritma kriptografi ini sederhana dan mudah diimplementasikan. RC4 dibuat oleh Ron Rivest dari Laboratorium RSA (RC adalah singkatan dari Ron's Code).

RC4 membangkitkan aliran kunci (keystream) yang kemudian di-XOR-kan dengan plainteks pada waktu enkripsi (atau di-XOR-kan dengan bit-bit cipherteks pada waktu dekripsi). Tidak seperti cipher aliran yang memproses data dalam bit, RC4 memproses data dalam ukuran *byte*(1 *byte* = 8 bit). Untuk membangkitkan aliran kunci, cipher menggunakan status internal yang terdiri dari dua bagian:

a. Permutasi angka 0 sampai 255 di dalam larik $S_0$, $S_1$, … , $S_{255}$ = 255. Permutasi merupakan fungsi dari kunci U dengan panjang variabel.
b. Dua buah pencacah indeks, i dan j.



**Gambar 2 : Diagram pembangkitan kunci-aliran *K*.**

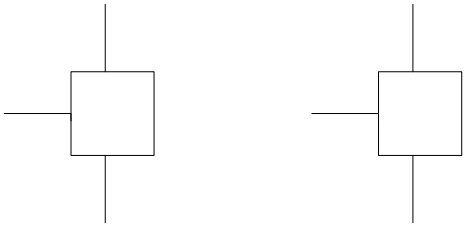
Karena karakter-karakter kunci di-*copy* berulang-ulang (untuk mengisi kekurangan 256 *byte*) maka ada kemungkinan nilai-nilai di dalam larik S ada yang sama. RC4 juga mudah diserang dengan *known-plaintext attack* jika kriptanalis

mengetahui beberapa buah plainteks dan cipherteks yang berkoresponden.

### *6.5. Serangan Terhadap ECB*

Kata "code book" di dalam ECB(Electronic Code Book) muncul dari fakta bahwa karena blok plainteks yang sama selalu dienkripsi menjadi blok cipherteks yang sama, maka secara teoritis dimungkinkan membuat buku kode plainteks dan cipherteks yang berkoresponden.

Namun, semakin besar ukuran blok, semakin besar pula ukuran buku kodenya. Misalkan jika blok berukuran 64 bit, maka buku kode terdiri dari $2^{64}$-1 buah kode(entry), yang berarti terlalu besar untuk disimpan. Lagipula, setiap kunci mempunyai buku kode yang berbeda.




**Gambar 3 Skema enkripsi dan dekripsi dengan mode *ECB***

Karena bagian plainteks sering berulang (sehingga terdapat blok-blok plainteks yang sama), maka hasil enkripsinya menghasilkan blok cipherteks yang sama.

Bagian plainteks yang sering berulang misalnya kata-kata seperti *dan, yang , ini , itu*, dan sebagainya.

Di dalam *e-mail*, pesan sering mengandung bagian yang redundan seperti *string* 0 atau spasi yang panjang, yang bila dienkripsi maka akan menghasilkan pola-pola cipherteks yang mudah dipecahkan dengan serangan yang berbasis statistik (menggunakan frekuensi kemunculan blok cipherteks). Selain itu, *e-mail* mempunyai struktur yang teratur yang menimbulkan pola-pola yang khas dlam cipherteksnya.

Misalnya kriptanalis mempelajari bahwa blok plainteks 5EB82F (dalam notasi HEX) dienkripsi menjadi blok AC209D, maka setiap kali ia menemukan cipherteks AC209D, ia dapat langsung mendekripsinya menjadi 5EB82F.

Satu cara untuk mengurangi kelemahan ini adalah menggunakan ukuran blok yang besar, misalnya 64 bit, sebab ukuran blok yang besar dapat menghilangkan kemungkinan menghasilkan blok-blok yang identik.

Pihak lawan dapat memanipulasi cipherteks untuk "membodohi" atau mengelabui penerima pesan.

Misalkan seseorang mengirim pesan

Uang ditransfer lima satu juta rupiah

Andaikan bahwa kriptanalis mengetahui bahwa blok plainteks terdiri dari dua huruf (spasi diabaikan sehingga menjadi 16 blok plainteks) dan blok-blok cipherteksnya adalah

$C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16}$

Penerima pesan mendekripsi keseluruhan blok cipherteks menjadi plainteks semula, sehingga ia dapat mendekripsi $C_1$ menjadi Ua, $C_2$ menjadi ng, $C_3$ menjadi di dan seterusnya. Kriptanalis memanipulasi cipherteks dengan membuang blok cipherteks ke-8 dan 9 sehingga menjadi

$C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_{10}, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16}$

Penerima pesan mendekripsi cipherteks yang sudah dimanipulasi dengan kunci yang benar menjadi

Uang ditransfer satu juta rupiah

Karena dekripsi menghasilkan pesan yang bermakna, maka penerima menyimpulkan bahwa uang yang dikirim kepadanya sebesar satu juta rupiah. Kelemahan ini dapat diatasi dengan mengatur enkripsi tiap blok individual bergantung pada semua blok-blok sebelumnya.

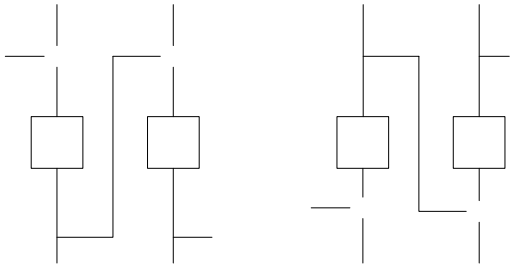
### *6.6. Serangan Terhadap CBC*

Mode CBC (*Cipher Block Chaining*) ini menerapkan mekanisme umpan-balik (*feedback*) pada sebuah blok, yang dalam hal ini hasil enkripsi blok sebelumnya di-umpan-balikkan ke dalam enkripsi blok yang *current*.

Caranya, blok plainteks yang current di-XOR-kan terlebih dahulu dengan blok cipherteks hasil enkripsi sebelumnya, selanjutnya hasil peng-XOR-an ini masuk ke dalam fungsi enkripsi.

Dengan mode CBC, setiap blok cipherteks bergantung tidak hanya pada blok plainteksnya tetapi juga pada seluruh blok plainteks sebelumnya.

Dekripsi dilakukan dengan memasukkan blok cipherteks yang *current* ke fungsi dekripsi, kemudian meng-XOR-kan hasilnya dengan blok cipherteks sebelumnya. Dalam hal ini, blok cipherteks sebelumnya berfungsi sebagai umpan-maju(*feedforward*) pada akhir proses dekripsi.



**Gambar 4 : Skema enkripsi dan dekripsi dengan mode *CBC***

Karena blok cipherteks yang dihasilkan selama proses enkripsi bergantung pada blok-blok cipherteks sebelumnya, maka kesalahan satu bit pada sebuah blok plainteks akan merambat pada blok cipherteks yang berkoresponden dan semua blok cipherteks berikutnya.

Tetapi, hal ini berkebalikan pada proses dekripsi. Kesalahan pada satu bit pada blok cipherteks hanya mempengaruhi blok plainteks yang berkoresponden dan satu bit pada blok plainteks berikutnya (pada posisi bit yang berkoresponden pula).

Kesalahan bit cipherteks biasanya terjadi karena adanya gangguan (*noise*) saluran komunikasi data selama transmisi atau *malfunction* pada media penyimpanan.

Karena blok cipherteks mempengaruhi blok-blok berikutnya, pihak lawan dapat menambahkan blok cipherteks tambahan pada akhir pesan terenkripsi tanpa terdektesi. Ini akan menghasilkan blok plainteks tambahan pada waktu dekripsi.

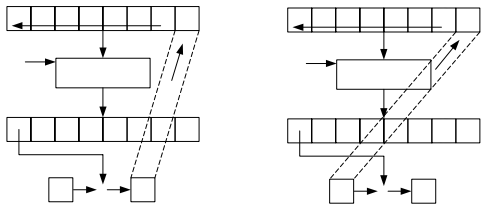
Pengirim pesan seharusnya menstrukturkan plainteksnya sehingga ia mengetahui di mana ujung pesan dan dapat mendeteksi adanya blok tambahan.

Pihak lawan juga dapat mengubah cipherteks, misalnya mengubah sebuah bit pada suatu blok cipherteks. Tetapi hal ini hanya mempengaruhi blok plainteks hasil dekripsinya dan satu bit kesalahan pada posisi plainteks berikutnya.

### *6.7. Serangan Terhadap CFB*

Pada mode CFB (*Cipher Feed-Back*), data dienkripsikan dalam unit yang lebih kecil daripada ukuran blok. Unit yang dienkripsikan dapat berupa bit per bit (jadi seperti cipher aliran), 2-bit, 3-bit, dan seterusnya. Bila unit yang dienkripsikan satu karakter setiap kalinya, maka mode CFB-nya disebut CFB 8-bit.

Secara umum, CFB *n*-bit mengenkripsi plainteks sebanyak *n* bit setiap kalinya, yang mana $n \leq m$ (m = ukuran blok). Dengan kata lain, CFB mengenkripsikan cipher blok seperti pada *cipher* aliran.



**Gambar 5 : Mode *CFB n*-bit**

$P_i$ $C_{i-1}$ $C_i$

Kesalahan 1-bit pada blok plainteks akan merambat pada blok-blok cipherteks yang berkoresponden dan blok-blok cipherteks selanjutnya pada proses enkripsi. Hal yang kebalikan juga terjadi pada proses dekripsi.

### *6.8. Serangan Terhadap DES*

$C_{i-2}$ $\oplus$ $\oplus$

$E_k$ $D_k$ $D_k$

Penggunaan DES (*Data Encryption Standard*) menimbulkan isu-isu yang menjadi perdebatan kontroversial menyangkut keamanan *DES* :

$C_{i-1}$ $C_i$ $P_{i-1}$ $P_i$

a. *Panjang kunci*
Panjang kunci DES pendek. Panjang kunci eksternal DES hanya 64 bit atau 8 karakter, itupun yang dipakai hanya 56 bit. Pada rancangan awal, panjang kunci yang diusulkan IBM adalah 128 bit, tetapi atas permintaan NSA, panjang kunci diperkecil menjadi 56 bit. Alasan pengurangan tidak diumumkan. Serangan paling praktis terhadap DES adalah *exhaustive key search*. Denagn panjang kunci 56 bit akan terdapat $2^{56}$ kemungkinan kunci. Jika diasumsikan serangan *exhaustive key search* dengan menggunakan prosesor paralel mencoba setengah dari jumlah kemungkinan kunci itu, maka dalam satu detik dapat dikerjakan satu juta serangan. Jadi seluruhnya diperlukan 1142 tahun untuk menemukan kunci yang benar.

Namun, pada tahun 1998 *Electronic Frontier Foundation* (EFE) merancang dan membuat perangkat keras khusus untuk menemukan kunci DES secara *exhaustive key search* dengan biaya $250.000 dan diharapkan dapat menemukan kunci selama 5 hari. Tahun 1999, kombinasi perangkat keras EFE dengan kolaborasi internet yang melibatkan lebih dari 100.000 komputer dapat menemukan kunci DES kurang dari 1 hari.

b. *Jumlah putaran*
Sebenarnya, delapan putaran sudah cukup untuk membuat cipherteks sebagai fungsi acak dari setiap bit plainteks dan setiap bit cipherteks. Dari penelitian, DES dengan jumlah putaran yang kurang dari 16 ternyta dapat dipecahkan dengan *known-plaintext attack* lebih mangkus daripada dengan *brute force attack*.

c. *Kotak-S*
Pengisian kotak-S DES masih menjadi misteri tanpa ada alasan mengapa memilih konstanta-konstanta di dalam kotak itu.

### *6.9. Kriptanalis terhadap Double DES*

Double DES mempunyai kelemahan yaitu ia dapat diserang dengan algoritma yang dikenal sebagai man-in-the-middle attack, yang pertama kali ditemukan oleh Diffie dan Hellman. Serangan ini didasarkan pada pengamatan bahwa jika kita memiliki :

$$C= E_{K2}(E_{K1}(P))$$

Maka

$$X= E_{K1}(P) = D_{K2}(C)$$

Jika kriptanalis memiliki potongan cipherteks C dan plainteks P yang berkoresponden, maka kriptanalis melakukan serangan sebagai berikut : Mula-mula, enkripsi P untuk semua kemungkinan nilai K1 (yaitu sebanyak $2^{56}$ kemungkinan kunci). Bandingkan semua hasil dekripsi ini dengan elemen di dalam tabel tadi. Jika ada yang sama, maka dua buah kunci, K1 dan K2, telah ditemukan. Tes kedua kunci ini dengan pasangan plainteks-cipherteks lain yang diketahui. Jika kedua kunci tersebut menghasilkan cipherteks atau plainteks yang benar, maka K1 dan K2 tersebut merupakan kunci yang benar.

## 7. Berbagai Kasus Serangan Kriptografi

Dalam kurun waktu terakhir ini telah banyak kasus serangan-serangan kriptanalis terhadap beberapa algoritma kriptografi tertentu. Berikut adalah pembahasan kasus-kasus tersebut.

### *7.1. Microsoft RC4 dipecahkan*

Salah satu aturan yang paling penting dari cipher aliran adalah jangan menggunakan *keystream* yang sama untuk mengenkripsi dua dokumen yang berbeda. Jika seseorang melakukannya, kita dapat memecahkan enkripsi dengan men-XOR-kan dua *stream* cipherteks bersamaan. *Keystream*-nya keluar, lalu kita men-XOR-kan plainteks dengan plainteks, dan kita dapat dengan mudah mendapatkan dua plainteks dengan menggunakan analisis frekuensi huruf atau pun dengan teknik yang lain.

Itu adalah kesalahan oleh amatir dalam kriptografi. Cara mudah untuk mencegah serangan ini adalah menggunakan vektor inisialisasi yang unik sebagai tambahan pada kunci ketika kita mengenkripsi dokumen.

*Microsoft* menggunakan cipher aliran RC4 di *Word* dan *Excel.* Cipher aliran RC4 dengan panjang kunci 128 bit digunakan di *Microsoft Word* dan *Excel* untuk menjaga dokumen. Tetapi ketika dokumen yang telah terenkripsi dimodifikasi dan disimpan, vektor inisialisasi tetap sama dan oleh karena itu *keystream* sama yang dibangkitkan dari RC4 digunakan juga untuk mengenkripsi dokumen yang sama dengan versi yang berbeda. Konsekwensinya sangat besar karena banyak informasi dari dokumen dapat didapatkan dengan mudah.

Hal ini bukan merupakan masalah yang baru. *Microsoft* juga membuat kesalahan yang sama di tahun 1999 dengan RC4 dalam aplikasi WinNT Syskey. Lima tahun kemudian, *Microsoft* membuat lagi kesalahan yang sama di berbagai produk.

### *7.2. CFB pada OpenPGP dipecahkan*

*OpenPGP Message Format* sangat populer untuk mengenkripsi data file, khususnya untuk menanda dan mengenkripsi *e-mail*. Format yang dijelaskan oleh *OpenPGP* telah diimplementasikan di banyak *freeware* dan produk enkrisi komersil. Enkripsi simetris di OpenPGP dibuat dengan mode CFB.

*Adaptive-chosen-ciphertext attack* digunakan pada protokol kriptografi yang mengizinkan penyerang untuk mendekripsikan cipherteks C, mendapatkan plainteks P, dengan mengirimkan sejumlah *chosen-ciphertext* C = C ke sebuah *oracle* yang mengirimkan informasi dekripsi. Cipherteks dapat dipilih sesuai kondisi, jadi informasi dari dekripsi sebelumnya didapatkan sebelum cipherteks terpilih berikutnya dikirim. Serangan pada *OpenGP CFB Mode* adalah didapatkannya seluruh plainteks dengan menggunakan satu *oracle query* yang dikirimkan ke penyerang seluruh dekripsi dari C.

Serangan ini membutuhkan oracle yang mengirimkan informasi dalam sebuah *ad-hoc integrity check* di *OpenPGP CFB Mod.* Dengan $2^{15}$ *oracle queries* untuk *initial setup* dan $2^{15}$ *queries* untuk setiap blok, penyerang dapat menentukan dua *byte* pertama dari plainteks di setiap blok. Serangan juga membutuhkan pengetahuan penyerang dua *byte* pertama dari plainteks satu blok apa aja untuk mem-*bootstrap* proses.

### *7.3. SHA-1 dipecahkan*

Tim riset yang terdiri dari Xiaoyun Wang, Yiqun Lisa Yin, dan Hongbo Yu (kebanyakan dari Shandong University di China) telah menyebarkan hasil riset mereka yaitu :

a. pecahnya SHA-1 di $2^{69}$ operasi *hash*, lebih sedikit daripada $2^{80}$ operasi *brute force* berdasarkan panjang *hash.*
b. Pecahnya SHA-0 di $2^{39}$ operasi
c. Pecahnya 58-round SHA-1 di $2^{33}$ operasi

Serangan ini membangun serangan-serangan sebelumnya di SHA-0 dan SHA-1 dan serangan ini telah dihasilkan dengan kriptanalisis. Ini merupakan ancaman bagi algoritma SHA-1 sebagai fungsi *hash* untuk *digital signature.*

## 8. Kesimpulan

Kesimpulan yang dapat dimbil dari studi berbagai serangan terhadap kriptografi adalah:

1. Algoritma kriptografi selalu berkembang dari masa ke masa.
2. Seiring dengan berkembangnya algoritma kriptografi, pengetahuan kriptanalis juga semakin berkembang sehingga serangan terhadap kriptografi semakin besar.
   Semakin rumit algoritma kriptografi atau semakin lama proses kriptanalisis dilakukan maka algoritma kriptografi tersebut semakin baik.
3. Jika perkembangan algoritma kriptografi terhenti maka ada waktunya seluruh algoritma kriptografi dapat dipecahkan. Oleh sebab itu harus ditemukan algoritma kriptografi yang baru yang lebih baik dan lebih sulit untuk dipecahkan oleh kriptanalis.
4. Pada setiap produk *software* yang dipakai oleh konsumen yang banyak haruslah memiliki prinsip dasar kriptografi yang benar sehingga data pribadi yang dipakai di software tersebut dapat diamankan dengan baik.

## DAFTAR PUSTAKA


[1] Munir, Rinaldi. (2004). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[2] Schneier, Bruce. (2005). Microsoft RC4 Flaw. *http://ironyuppie.com/2005/01/microsoft-rc4-flaw.html*. Tanggal akses: 5 Oktober 2006 pukul 15:00.

[3] Schneier, Bruce. (2005). SHA-1 Broken. *http://www.schneier.com/blog/archives/2005/02/sha1_broken.htm*. Tanggal akses: 5 Oktober 2006 pukul 15:30.

[4] Mister, Serge, Robert Zuccherato. (2005). An Attack on CFB Mode Encryption as Used by OpenPGP. *http://eprint.iacr.org/2005/033.pdf*. Tanggal akses: 5 Oktober 2006 pukul 15:45.