Bahan Kuliah ke-2

IF5054 Kriptografi

Serangan (*Attack*) Terhadap Kriptografi

Disusun oleh:

Ir. Rinaldi Munir, M.T.

Departemen Teknik Informatika
Institut Teknologi Bandung
2004

2. Serangan (*Attack*) Terhadap Kriptografi

2.1 Pendahuluan

- Keseluruhan *point* dari kriptografi adalah menjaga kerahasiaan plainteks (atau kunci, atau keduanya) dari penyadap (*eavesdropper*) atau kriptanalis (*cryptanalyst*).

- Penyadap bisa juga merangkap sebagai seorang kriptanalis.
  Nama lain penyadap: - penyusup (*intruder*)
  - penyerang (*attacker*)
  - musuh (*enemy, adversaries*)
  - pencegat (*interceptor*)
  - lawan (opponent)

- Penyadap berusaha mendapatkan data yang digunakan untuk kegiatan kriptanalisis (*cryptanalysis*).

- Kriptanalis 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.

2.2 Metode Penyadapan

Beberapa metode penyadapan data:

1. *Wiretapping*
   Penyadap mencegat data yang ditransmisikan pada saluran kabel komunikasi dengan menggunakan sambungan perangkat keras.

2. *Electromagnetic eavesdropping*
   Penyadap mencegat data yang ditransmisikan melalui saluran *wireless*, misalnya radio dan *microwive*.

3. *Acoustic Eavesdropping*.
   Menangkap gelombang suara yang dihasilkan oleh suara manusia.

2.3 Serangan (*attack*)

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

- Secara umum, ada dua macam serangan:

  1. *Exhaustive attack* atau *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 dengan setiap kemungkinan kunci dan plainteks hasilnya diperiksa apakah mengandung makna.

Misalkan sebuah sistem 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 (lihat Tabel 1).

Tabel 1 Waktu yang diperlukan untuk *exhaustive key search* (Sumber: William Stallings, *Data and Computer Communication Fourth Edition*)

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	$2^{128} = 4.3 \times 10^{38}$	5.4×10^{24} tahun	5.4×10^{18} tahun

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

2. 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 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 plianteks merupakan fungsi matematika dari cipherteks dan kunci yang cukup kompleks, dan tiap kunci merupakan fungsi matematika dari cipherteks dan plainteks yang cukup kompleks.

Metode *analytical attack* biasanya lebih cepat menemukan kunci dibandingkan dengan *exhaustive attack*.

- Data yang digunakan untuk menyerang sistem kriptografi dapat dikategorikan sebagai berikut:
  1. *Chipertext only*.
  2. *Known plaintext* dan *corresponding chipertext.*
  3. *Chosen plaintext* dan *corresponding chipertext.*
  4. *Chosen chipertext* dan *corresponding plaintext.*

- Berdasarkan ketersediaan data yang ada, serangan terhadap kriptografi dapat diklasifikasikan menjadi (asumsi yang digunakan: kriptanalis mengetahui algoritma kriptografi yang digunakan):

1. *Chipertext-only attack*

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

   Tugas kriptanalisis adalah menemukan plainteks sebanyak mungkin atau menemukan kunci yang digunakan untuk mengenkripsi pesan.

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

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

2. *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", *wassalam*, pada surat resmi.
   #include, *program*, di dalam *source code*

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

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

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.

   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 diantara P_1, P_2, …, P_i

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

4. *Adaptive-chosen-plaintext attack*

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

5. *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, $P_2 = D_k(P_2)$, …,
   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.

6. *Chosen-text attack*

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

- Jenis-jenis serangan lainnya:

  7. *Chosen-key attack*

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

  8. *Rubber-hose cryptanalysis*
     Kriptanalis mengancam, mengirim surat gelap, atau melakukan penyiksaan sampai orang yang memegang kunci memberinya kunci untuk mendekripsi pesan. Mungkin ini cara yang terbaik untuk memecahkan kriptografi.

- Kompleksitas serangan dapat diukur dengan beberapa cara:
  1. 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.

  2. Kompleksitas waktu (*time complexity*)

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

  3. Kompleksitas ruang memori (*space/storage complexity*)

     Jumlah memori yang dibutuhkan untuk melakukan serangan. Semakin banyak memori yang yang dibutuhkan untuk untuk melakukan serangan, berarti semakin bagus algoritma kriptografi tersebut.

- Dalam pembahasan tentang serangan terhadap kriptografi, kita selalu mengasumsikan kriptanalis mengetahui algoritma kriptografi, sehingga keamanan algoritma terletak sepenuhnya pada kunci. Hal ini didasarkan pada *Prinsip Kerckhoff* (1883) yang berbunyi:

  Prinsip Kerckhoff: *Semua algoritma kriptografi harus publik; hanya kunci yang rahasia.*

- Jika keamanan algoritma kriptografi ditentukan dengan menjaga kerahasiaan algoritmanya, maka algoritma tersebut dinamakan algoritma **terbatas** (*restricted*). Algoritma terbatas tidak cocok lagi saat ini.

  Misalkan algoritma terbatas digunakan oleh orang-orang di dalam sebuah grup. Setiap orang mengenkripsi dan mendekripsi pesan dengan algoritma yang hanya diketahui oleh mereka saja. Tetapi, jika seorang anggota keluar dari grup, maka algoritma tersebut harus diganti (anggota yang keluar belum tentu bisa dipercaya akan tetap merahasiakan algoritma grupnya dulu).

- Dengan mempublikasikan algoritma kriptografi, kriptografer memperoleh konsultasi gratis dari sejumlah kriptologis akademisi yang ingin sekali memcahkan algoritma sehingga mereka dapat mempubliksikan paper yang memperlihatkan kecerdasan mereka.

- Jika banyak pakar telah mencoba memecahkan algoritma selama 5 tahun setelah dipublikasikan dan tidak seorangpun berhasil, maka mungkin algoritma tersebut tangguh.

2.4 Keamanan Algoritma Kriptografi

- Sebuah algoritma kriptografi dikatakan aman (*computationally secure*) bila ia memenuhi tiga kriteria berikut:

    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.

       Lihat Tabel 1 untuk panjang kunci 128 bit (7 karakter). Untuk menemukan kunci, setidaknya setengah dari semua kemungkinan kunci yang ada harus dicoba, dan akan menghabiskan waktu 5.4×10^{24} tahun untuk satu juta percobaan per detik. Hal ini tidak mungkin karena umur alam ini saja baru pada orde 10^{11} tahun.