Pembangkit Kunci untuk *Cipher One-time Pad*

Arief Pratama [1)]

1) Jurusan Teknik Informatika Sekolah Teknik Elektro dan Informatika ITB, Bandung, email: if14070@students.if.itb.ac.id

Abstract – *One-time Pad (OTP) ditemukan oleh Major Joseph Mauborgne pada tahun 1917 dan termasuk ke dalam kelompok algoritma kriptografi simetri. Keunggulan algoritma ini adalah kesempurnaan cipher-nya sehingga tidak bisa dipecahkan. OTP menggunakan prinsip Vigenere dalam melakukan enkripsi dan dekripsi pesan. Perbedaannya, pada OTP tidak terjadi pengulangan kunci, namun panjang kunci harus sama dengan panjang plaintext. Ini merupakan salah satu kelemahan OTP karena mensyaratkan kunci sepanjang plaintext.*

Kata Kunci*: One-time pad, Vigenere, pembangkit kunci, kasiski.*

1. PENDAHULUAN

One-time pad adalah satu-satunya algoritma *unbreakable cipher* yang tidak bisa dipecahkan. Satu pad (kertas bloknot) biasanya digunakan untuk menyimpan deretan karakter kunci yang acak dan hanya dapat dipakai sekali untuk mengenkripsi pesan. Setelah pesan diterima, pad tersebut dihancurkan dan tidak akan dipakai lagi untuk enkripsi pesan berikutnya.

Masalah yang timbul pada algritma ini adalah ketika pendistribusian kunci. Pendistribusian ini harus dilakukan karena kunci dibangkitkan secara acak, sehingga mustahil bagi pihak penerima mendapatkan kunci yang sama dengan kunci pihak pengirim. Pihak penerima juga tidak mungkin untuk mengingat kunci, karena panjangnya sama dengan panjang pesan. Oleh karena itu, pengiriman pesan harus diikuti pula dengan pengiriman kunci. Hal ini sangat berisiko menimbang saluran komunikasi yang sangat padat dan tidak aman.

Salah satu cara untuk mengatasi masalah ini adalah dengan menghindari proses pengiriman kunci. Si penerima dan pengirim dapat menyepakati kunci (asli) yang akan digunakan tanpa mengirimnya dengan syarat kunci tersebut tidak harus sepanjang pesan dan dapat diingat dengan mudah. Hal ini dapat dilakukan dengan membangkitkan karakter tambahan dari kunci asli, sehingga panjangnya sama dengan panjang pesan.

2. PEMBANGKITAN KUNCI

Pada makalah ini penulis mengusulkan dua metode untuk membangkitkan kunci. Pertama, dengan memanfaatkan karakter-karakter pada kunci; dan kedua, dengan memanfaatkan karakter-karakter pada kunci dengan cipher.

Metode-metode pembangkit kunci pada makalah ini mengasumsikan bahwa karakter-karakter pesan dan kunci hanya terdiri dari karakter ‘A’ sampai ‘Z’.

1.1. Pembangkitan kunci berdasarkan karakter kunci sebelumnya

Pada metode ini, kunci dibangkitkan dengan menggunakan karakter-karakter sebelumnya yang ada pada kunci.

Langkah-langkah pembangkitan kunci sebagai berikut:

1. karakter terakhir dari kunci dijumlahkan dengan *n-1* karakter sebelumnya (*n* adalah panjang kunci asli), kemudian jumlah tersebut dimodulo 26.
2. Hasil modulo merupakan karakter baru yang kemudian digabungkan dengan kunci sebelumnya menjadi kunci baru.
3. Kembali ke langkah pertama sampai kunci tersebut sepanjang plaintext.

Dari langkah-langkah di atas, dapat dirumuskan persaman berikut untuk menghasilkan karakter kunci ke-*i*:

$$k_i = (k_{i-n} + k_{i-1}) \bmod 26 \qquad (1)$$

i = n+1, n+2, …
k_i = karakter kunci ke-*i* (A = 0, B = 1, …)
n = panjang kunci asli

Dari persamaan (1), ciphertext dapat diperoleh dengan:

$$c_i = (p_i + k_i) \bmod 26, \quad \text{untuk } i <= n \qquad (2)$$

$$c_i = (p_i + ((k_{i-n} + k_{i-1}) \bmod 26)) \bmod 26, \quad \text{untuk } i > n \qquad (3)$$

c_i = karakter ciphertext ke *i*
p_i = karakter plaintext ke *i*
k_j = karakter kunci ke *j*
i = 1, 2, 3, …
n = panjang kunci asli

Sebagai contoh, misalkan kunci “KEY” akan

digunakan untuk mengenkripsi plaintext sepanjang 20 karakter. Kunci asli tersebut (n = 3) harus diperpanjang menjadi 20 karakter dengan membangkitkan 17 karakter tambahan.

Karakter ke 4 = `(K+Y)` mod 26 = `I` (kunci=`KEYI`)

Karakter ke 5 = `(E+I)` mod 26 = `M` (kunci=`KEYIM`)

Karakter ke 6 = `(Y+M)` mod 26 = `K` (kunci=`KEYIMK`)

dan seterusnya.

Setelah diproses sampai karakter ke-20, kunci menjadi: `KEYIMKSEOGKYEOMQEQGK`. Kunci tersebut siap digunakan untuk mengenkripsi pesan.

Pembangkitan kunci dengan metode ini mensyaratkan kunci asli tidak boleh hanya terdiri dari karakter 'A'. Jika kunci asli hanya terdiri dari karakter-karakter 'A', kunci hasil pembangkitan juga akan terdiri dari karakter-karakter 'A' dan hasil enkripsinya akan sama dengan ciphertext.

1.2. Pembangkitan kunci berdasarkan karakter kunci dan karakter cipher sebelumnya

Pada metode ini, kunci dibangkitkan dengan didapatkan dengan mengenkripsi karakter cipher n-1 dengan karakter n-1, di mana n adalah panjang kunci saat ini (*current*).

Langkah-langkah pembangkitan kunci:

1. Karakter ke *i* dihasilkan dengan mengenkripsi karakter cipher ke *i-n* dengan karakter kunci ke *i-n*. Hasilnya kemudian dimodulo 26.
2. Hasil modulo tersebut merupakan karakter baru yang kemudian digabungkan dengan kunci sebelumnya menjadi kunci baru.
3. Kembali ke langkah pertama sampai kunci tersebut sepanjang plaintext.

Dari langkah-langkah di atas, dapat dirumuskan persamaan berikut untuk menghasilkan karakter kunci ke-*i*:

$$k_i = E(c_{i-n},k_{i-n}) = (c_{i-n} + k_{i-n}) \bmod 26 \qquad (4)$$

k_i = karakter kunci ke-*i* (A = 0, B = 1, …)
i = n+1, n+2, …
E = enkripsi dengan Vigenere
n = panjang kunci asli
c_i = karakter cipher ke-*i* (A = 0, B = 1, …)

Dengan menggunakan rumus di atas, ciphertext dapat doperoleh dengan:

$$c_i = (p_i + k_i) \bmod 26, \qquad (5)$$
untuk i <= n

$$c_i = (p_i + (c_{i-n} + k_{i-n}) \bmod 26) \bmod 26, \qquad (6)$$
untuk i > n

c_i = karakter ciphertext ke *i*
p_i = karakter plaintext ke *i*
k_j = karakter kunci ke *j*
i = 1, 2, 3, …
n = panjang kunci asli

Sebagai contoh, kunci '`KEY`' akan digunakan untuk mengenkripsi pesan '`ALLRIGHTSRESERVED`'.

```
Plain  : ALL RIG HTS RES ERV ED
Kunci  : KEY UTH FUU RHG ZSE CB
Cipher : KPJ LBN MNM ILY DJZ GE
```

Dengan menggunakan persamaan (6):

- Enkripsi karakter 'R' (i=4, n=3)
  $c = (p_4 + (c_1 + k_1) \bmod 26) \bmod 26 = L$
- Enkripsi karakter 'I' (i=5, n=3)
  $c = (p_5 + (c_2 + k_2) \bmod 26) \bmod 26 = B$
- Enkripsi karakter 'G' (i=6, n=3)
  $c = (p_6 + (c_3 + k_3) \bmod 26) \bmod 26 = N$
- Enkripsi karakter 'H' (i=7, n=3)
  $c = (p_7 + (c_4 + k_4) \bmod 26) \bmod 26 = M$
- Enkripsi karakter 'T' (i=8, n=3)
  $c = (p_8 + (c_5 + k_5) \bmod 26) \bmod 26 = N$

dan seterusya.

Kunci asli dapat terdiri dari karakter apapun dengan panjang minimal 1 karakter.

3. ANALISIS TERHADAP PEMBANGKIT KUNCI

1. Analisis metode pertama

a. Kunci hasil pembangkitan
Misalkan kita ingin memperpanjang kunci "KE" menjadi 100 karakter. Kunci yang dihasilkan adalah:

```
KEOSGYECGIOWKGQWMIUCWYUSMEQUKEOSGYECGIOWKGQ
WMIUCWYUSMEQUKEOSGYECGIOWKGQWMIUCWYUSMEQUKE
OSGYECGIOWKGQW
```

Pada kunci di atas, terjadi perulangan setelah 28 karakter.

Jika kunci yang ingin diperpanjang adalah "AD", maka kunci yang dihasilkan adalah:

```
ADDGJPYNLYJHQXNKXHELPAPPETXQNDQTJCLNYLJUDXA
XXURLCNPCRTKDNQDTWPLALLWHDKNXKHRYPNCPRGXDAD
DGJPYNLYJHQXNK
```

Pada kunci di atas, terjadi perulangan setelah 84 karakter.

Perulangan ini umumnya terjadi pada setiap kelipatan 28. Jika kunci asli sebanyak 3 karakter, maka perulangan dapat terjadi setelah 168 karakter. Oleh karena itu, tingkat keacakan kunci bergantung

kepada panjang dan keacakan kunci asli. Semakin panjang kunci asli maka kunci yang dibangkitkan akan semakin acak dan perulangan semakin sedikit.

b. Hasil enkripsi

Misalkan kita ingin mengenkripsi pesan berikut:
`PEMBANGKITKUNCICIPHERONETIMEPAD`
dengan menggunakan kunci : `KEY`

Plain : `PEMBANGKITKUNCICIPHERONETIMEPAD`
Kunci : `KEYIMKSEOGKYEOMQEQGKAGQQWMCYKMK`
Cipher : `ZIKJMXYOWZUSRQUSMFNORUDUPUOCZMN`

Pada contoh di atas, ciphertext yang dihasilkan tidak dapat dipecahkan dengan mencari pola-pola statistic, karena pola-pola pada ciphertext sangat tidak beraturan. Metode kasiski juga tidak dapat digunakan karena keacakan pola ini membuat kriptanalis sulit untuk menentukan panjang kunci. Kalaupun seandainya terjadi pengulangan pada karakter kunci yang dibangkitkan, pengulangan tersebut tidak berpengaruh besar pada pola ciphertext karena ukurannya besar (kelipatan 28).

Jika diasumsikan bahwa kriptanalis mengetahui panjang kunci asli, pemecahan terhadap cipher dapat dilakukan dengan *brute force* terhadap n (panjang kunci asli) karakter pertama karena n karakter pertama pada kunci adalah kunci asli itu sendiri

Jika serangan yang dilakukan adalah dengan *known-plaintext attack*, serangan tersebut hanya akan bekerja jika, (1) plaintext yang diketahui adalah bagian awal dari pesan; (2) panjang kunci diketahui. Jika panjang kunci tidak diketahui, kriptanalis akan kesulitan mendekripsi ciphertext berikutnya.

2. Analisis metode kedua

Dengan menggunakan contoh di atas, enkripsi dilakukan sebagai berikut:

Plain : `PEMBANGKITKUNCICIPHERONETIMEPAD`
Kunci : `KEYJMITYDSGODWWTUAOWPJWVGFUFSAO`
Cipher : `ZIKKMVZILLQIQYEVCPVAGXJZZNGJHAR`

Berbeda dengan metode pertama, pada metode ini pengulangan terhadap kunci tidak akan terjadi. Keacakan kunci hasil pembangkitan bergantung pada keacakan kunci asli dan keacakan plaintext.

Sama seperti metode pertama, ciphertext sangat sulit dipecahkan dengan mencari pola-pola statistik, karena hasil yang begitu acak. Metode kasiski juga tidak bisa dilakukan karena panjang kunci tidak bisa ditentukan dari ciphertext.

Jika kriptanalis mengetahui panjang kunci asli, pencarian kunci dapat dilakukan dengan *brute force* terhadap bagian awal ciphertext. Jika kriptanalis menggunakan *known-plaintext attack*, serangan ini bekerja dengan syarat, (1) plaintext yang diketahui adalah bagian awal dari pesan; (2) panjang kunci diketahui.

3. PENGEMBANGAN LEBIH LANJUT

Berdasarkan hasil analisis, kedua metode tersebut dapat dikembangkan lebih lanjut. Beberapa pengembangan yang dapat dilakukan adalah:

1. Memperluas cakupan plaintext yang dapat dienkripsi.
2. Untuk metode pertama, pembangkitan kunci dapat dilakukan dalam beberapa tahap, yaitu:
   - Bangkitkan kunci sampai terjadi perulangan
   - Kunci hasil pembangkitan dianggap kunci asli sementara yang dari kunci tersebut dibangkitkan lagi kunci baru.
   - Proses diatas dilakukan secara berulang-ulang sampai kunci sepanjang plaintext dan tidak ada perulangan pada kunci.
3. Metode ini dapat diterapkan dalam kriptografi modern untuk membangkitkan kunci internal.

4. KESIMPULAN

Dengan menggunakan pembangkit kunci, aplikasi *one-time pad* pada dunia nyata dapat dilakukan dengan lebih mudah. Kunci yang dikirimkan tidak perlu sepanjang plantext. Namun kekuatan cipher tetap bergantung pada keacakan kunci. Semakin acak kunci yang dihasilkan, maka akan semakin sulit cipher tersebut dipecahkan.

Metode yang penulis usulkan pada makalah ini hanya metode sebagian dari banyak metode yang dapat ditemukan untuk membangkitkan kunci.

DAFTAR REFERENSI

[1] Munir, Rinaldi, “Diktat Kuliah IF5054 Kriptografi”, Institut Teknologi Bandung, 2006.