Tugas Makalah II

Kriptanalisis terhadap Pembangkit Bilangan Acak Semu

Yudi Haribowo - 13504111

Program Studi Teknik Informatika
Institut Teknologi Bandung
Jl. Ganesha 10 Bandung 40132
E-mail : if14111@students.if.itb.ac.id

Abstraksi

Dalam algoritma-algoritma enkripsi, banyak diperlukan bilangan acak (random) sebagai inputnya. Sebagai contoh, algoritma One Time Pad (OTP), pembangkitan *initialization vector* (IV), pembangkitan parameter kunci di dalam sistem kriptografi kunci publik, dan sebagainya. Yang dimaksud dengan acak di sini adalah blangannya tidak mudah diprediksi oleh kriptanalis. Akan tetapi, sangat sulit memperoleh bilangan acak dalam praktek kriptografi. Tidak ada prosedur komputasi yang benar-benar menghasilkan deret bilangan acak secara sempurna. Bilangan acak yang dihasilkan dengan rumus-rumus matematika adalah bilangan acak semu, karena bilangan acak yang dibangkitkan dapat berulang kembali secara periodik. Pembangkit deret bilangan acak semacam itu disebut *pseudo-random number generator* (PRNG).

Dalam makalah ini akan dibahas mengenai beberapa model algoritma pembangkit bilangan acak, membahas kemungkinan serangan-serangan yang terjadi terhadapnya dan mengaplikasikan model serta serangannya dalam lima buah algoritma PRNG. Kelima algoritma itu antara lain fungsi acak rand ANSI C, PRNG LCG, PRNG Blum Blum Shut, PRNG DSA, dan PRNG ANSI X9.17. Pada bagian akhir makalah akan dibahas mengenai kesimpulan yang diperoleh tentang desain dan penggunaan PRNG.

***Kata kunci**: PRNG, kriptanalisis, kriptografi kunci publik*

1 Pendahuluan

Kebanyakan aplikasi kriptografi yang di desain dengan baik menggunakan bilangan acak sebagai elemennya. Kunci sesi, vector inisialisasi, *salts* yang digunakan dalam fungsi hash dengan password, dan parameter unik dalam operasi tanda tangan digital semuanya diasumsikan merupakan bilangan acak oleh desainer sistem. Akan tetapi, sangat sulit membuat bilangan acak yang benar-benar acak seperti yang diinginkan. Akhirnya, para desainer sistem menggunakan mekanisme yang digunakan dalam kriptografi, yaitu Pembangkit Bilangan Acak Semu (Pseudo-Random Number Generator) untuk membangkitkan bilangan-bilangan acak tersebut.

Dalam makalah ini, kita memandang PRNG dari sudut pandang kriptanalis. Makalah ini mendiskusikan kebutuhan PRNG, memberikan gambaran bagaimana sebuah PRNG bekerja, dan mengidentifikasi serangan-serangan yang mungkin terjadi terhadap PRNG. Makalah ini akan membahas secara lebih detail apa yang mungkin dilakukan oleh penyerang untuk membuat PRNG tidak menghasilkan bilangan random sebagaimana harusnya dan memperkirakan kemungkinan output dari sebuah PRNG menggunakan pengetahuan yang ada.

Makalah ini mengimplikasikan beberapa teori dan praktik yang penting, yaitu:

1. PRNG adalah sebuah fungsi kriptografi yang primitf dalam arti tidak ada kesepakatan umum yang menyatakan kemungkinan serangan yang mungkin dilakukan terhadapnya dan juga perbedaan yang tampak antara PRNG yang satu dengan yang lain. Pemahaman tentang hal-hal tersebut memudahkan kita untuk mendesain dan menggunakan sebuah PRNG yang aman.
2. PRNG memegang peranan penting dalam sistem kriptografi karena kesalahan desain atau pemilihan PRNG membuat algoritma kriptografi yang sudah dipilih dengan hati-hati menjadi percuma.

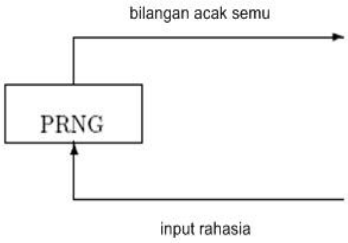
3. Banyak sistem kriptografi menggunakan PRNG yang tidak aman sehingga membuat memperbanyak kemungkinan serangan yang dapat dilakukan terhadapnya. Penulis menyadari hal ini dalam membuat makalah ini.
4. Makalah ini memberikan contoh hasil dari PRNG yang banyak digunakan dalam sistem kriptografi umum.

2 Dasar Teori

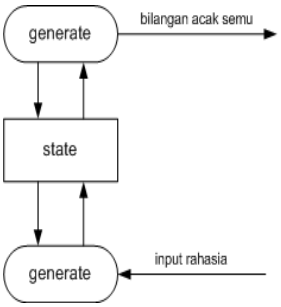
Dalam algoritma-algoritma enkripsi, banyak diperlukan bilangan acak (random) sebagai inputnya. Sebagai contoh, algoritma One Time Pad (OTP), pembangkitan *initialization vector* (IV), pembangkitan parameter kunci di dalam sistem kriptografi kunci publik, dan sebagainya. Yang dimaksud dengan acak di sini adalah blangannya tidak mudah diprediksi oleh kriptanalis. Akan tetapi, sangat sulit memperoleh bilangan acak dalam praktek kriptografi. Tidak ada prosedur komputasi yang benar-benar menghasilkan deret bilangan acak secara sempurna. Bilangan acak yang dihasilkan dengan rumus-rumus matematika adalah bilangan acak semu, karena bilangan acak yang dibangkitkan dapat berulang kembali secara periodik. Pembangkit deret bilangan acak semacam itu disebut *pseudo-random number generator* (PRNG).

Secara umum, sebuah PRNG didefinisikan sebagai algoritma kriptografi yang digunakan untuk menghasilkan bilangan secara acak. Pengertian acak sendiri adalah bilangan yang dihasilkan dalam setiap waktu tidaklah sama. Sebuah PRNG memiliki sebuah kondisi awal K yang rahasia. Saat digunakan, PRNG harus membangkitkan output acak yang tidak dapat diidentifikasi oleh kriptanalis yang tidak tahu dan tidak dapat menebak kondisi awal K. Dalam hal ini, PRNG memiliki kesamaan dengan cipher aliran. Akan tetapi, sebuah PRNG harus mampu mengubah kondisi awalnya dengan memproses input sehingga tidak dapat diprediksi oleh kriptanalis. Umumnya PRNG memiliki kondisi awal yang tidak sengaja dapat ditebak oleh kriptanalis dan harus mengalami banyak proses sebelum kondisinya rahasia dan aman.

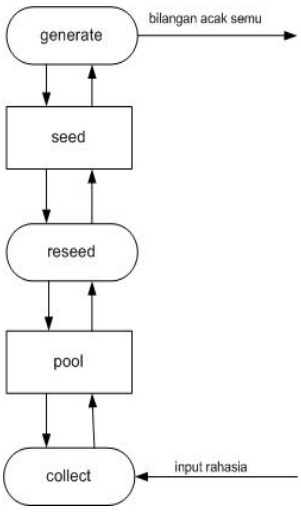
Patut dipahami bahwa sebuah input untuk PRNG memiliki informasi rahasia yang tidak diketahui oleh kriptanalis. Input-input ini umunya diperoleh dari proses-proses fisik, interksi user dengan mesin, atau proses eksternal lain yang sulit diprediksi. Dalam desain dan implementasi harus dapat dipastikan bahwa input-input ini memiliki cukup jaminan keamanan dan kerahasiaan.



Gambar 1. Skema dasar PRNG



Gambar 2 Skema internal PRNG



Gambar 3 Skema PRNG dengan pembibitan ulang

PRNG umunya dibangun dari fungsi-fungsi primitive kriptografi yang lain, seperti cipher block, fungsi hash, dan cipher airan. Banyak anggapan bahwa keamanan sistem-sistem di atas

menggambarkan keamanan dari PRNG yang menggunakannya.

Pembangkit bilangan acak yang dapat menghasilkan bilangan yang tidak dapat diprediksi oleh penyerang cocok untuk kriptografi. Pembangkti ini dinamakan *cryptographically secure pseudorandom number generator* (CSPRNG). CSPRNG memiliki syarat:

1. Lulus uji keacakan statistik
2. Tahan terhadap serangan yang serius. Serangan ini bertujuan memprediksi bilangan acak yang dihasilkan.

Untuk persyaratan yang kedua ini, maka CSPRNG seharusnya memenuhi dua syarat sebagai berikut:

1. setiap CSPRNG seharusnya memenuhi "uji bit berikutnya" (*next bit test*) sebagai berikut: diberikan k buah bit barisan acak, maka tidak ada algoritma dalam waktu polynomial yang dapat memprediksi bit ke-(k+1) dengan peluang keberhasilan lebih dari ½.
2. setiap CSPRNG dapat menahan "perluasan status", yaitu jika sebagian atau semua statusnya dapat diungkap atau diterka dengan benar, maka tidak mungkin merekonstruksi aliran bilangan acak.

Dalam makalah ini, kita membahas beberapa serangan baru yang dapat dilakukan terhadap PRNG. Kebanyakan serangan ini hanya merupakan konsep atau teori. Namun, kita tidak bisa mengesampingkan hal tersebut meski kecil kemungkinan terjadinya. Hal ini sesuai dengan pernyataan dalam Murphy's Law:"*If something can go wrong it will*". Sebagai tambahan, dapat diyakini bahwa serangan terhadap PRNG dalam sistem yang tidak memungkinkan serangan semacam ini bisa saja terjadi dalam penggunaan yang lain.

3 Macam-macam Serangan

3.1 *Direct Cryptanalytic Attack*

Serangan semacam ini dapat terjadi jika kriptanalis dapat memebedakan secara langsung antara hasil PRNG dengan ouput yang benar-benar acak. Serangan ini dapat diterapkan hampir pada semua jenis PRNG. Sebagai contoh, PRNG yang digunakan dalam membangkitkan kunci triple-DES tidak diserang dengan cara ini karena hasil pembangkitan bilangan acaknya tidak dapat diketahui.

3.2 *Input-based Attack*

Serangan mungkin terjadi jika krptanalis dapat mengetahui atau mengendalikan penggunaan masukan untuk PRNG untuk mengkriptanalisnya, contoh untuk membedakan hasil PRNG dengan bilangan acak.

Serangan terhadap masukan lebih jauh dapat dibedakan menjadi *known-input, replayed-input,* dan *chosen-input attacks*. *Chosen-input attacks* dapat dilakukan terhadap smart-cards dan alat lain yang tahan terhadap intervensi dari serangan kriptanalis. Serangan ini juga dapat terjadi pada aplikasi yang memberi pesan masuk, password pilihan user, statistic jaringan dan sebagainya. *Replayed-input attack* sepertinya dapat dilakukan dalam situasi yang sama dengan *chosen-input attack* tetapi membutuhkan lebih sedikit memebutuhkan kendali dari kriptanalis. *Known input attack* dapa diterapkan dalam PRNG yang inputnya didesain agar sulit diprediksi, tetapi ternyata menjadi mudah dalam kasus-kasus tertentu. Sebagai contoh menggunakan latency hard-drive untuk inputnya tetapi penyerang menggunakan drive network yang timingnya sudah diatur.

3.3 *State Compromise Extension Attack*

Serangan ini berusaha memanfaatkan keberhasilan dari usaha sebelumnya untuk mencoba memperoleh informasi mengenai kondisi awal K semaksimal mungkin meski tidak utuh informasinya. Diasumsikan ada keberhasilan dengan penetrasi keamanan komputer atau kriptanalisis, kriptanalis kemudian mempelajari kondisi awal K pada suatu satuan waktu. Serangan ini disebut berhasil jika penyerang mampu memperoleh hasil PRNG sebelum kondisi awal K diubah atau memperoleh output setelah PRNG mengolah input-input yang tidak diketahui oleh penyerang.

State compromise extension attack hanya dapat dilakukan terhadap PRNG yang dimulai dalam kondisi tidak aman/dapat diterka. Serangan ini juga dapat beekrja ketika K sudah diubah oleh serangan-serangan jenis lain seperti yang sudah disebut di sini. Dalam kenyataannya, penting untuk menganggap bahwa perubahan kondisi K yang sangat jarang mungkin saja terjadi. Untuk menjamin kekokohan sistem, PRNG seharunya mampu menahan serangan semacam ini sebaik mungkin. Serangan jenis ini sendiri terbagi menjadi beberapa varian:

a) Backtracking Attacks.

Backtracking attack menggunakan peluasan status K dari sebuah PRNG pada suatu waktu t untuk memperoleh output PRNG sebelumnya.

b) Permanent Compromise Attacks.

Serangan ini muncul jika pada suatu saat penyerang berhasil memperoleh informasi mengenai kondisi internal K dari PRNG, semua nilai-nilai sebelum dan sesudah S menjadi rawan diserang.

c) Iterative Guessing Attacks.

Serangan ini menggunakan informasi nilai kondisi internal K pada suatu waktu t dan output-output yang muncul untuk mempelajari nilai K pada waktu t+ε saat input-input yang diperoleh pada interval waktu ini dapat ditebak (tapi tidak diketahui) oleh penyerang.

d) Meet-in-the-middle Attacks.

Serangan ini merupakan kombinasi dari i*terative guessing attack*s dengan *backtracking attack*. Informasi mengenai kondisi internal K pada waktu t dan t+2ε memungkinkan penyerang merekonstruksi K pada waktu t+ε.

4 Kriptanalisis PRNG

4.1 Fungsi Random ANSI C

Fungsi random ANSI C atau dalam penggunaannya dinotasikan dengan `rand()` berfungsi menghasilkan bilangan acak yang dapat digunakan untuk berbagai macam keperluan. Bilangan yang dihasilkan memiliki interval 0..32767.

Fungsi ini memiliki bentuk umum sebagai berikut:

```
unsigned long int next = 1;
/* rand: mengembalikan bilangan acak semu 0..32767 */
int rand(void)
{
   next = next*1103515245+12345;
   return   (unsigned int)
            (next/65536)%32768;
}

/*    srand:    menset    nilai seed(bibit) untuk fungsi rand() */
void srand(unsigned int seed)
{
   next = seed;
}
```

Macam-macam serangan yang mungkin:

1 Direct Cryptanalytic Attack.

Dengan melihat algoritma di atas, dapat diperhatikan bahwa dengan sebuah nilai bibit yang sama, bilangan-bilangan yang dihasilkan pasti akan memiliki urutan yang sama pula sehingga dapat disimpulkan PRNG ini dapat dibedakan antara output yang dihasilkan dengan bilangan yang benar-benar acak. Dengan menggunakan informasi ini, penyerang hanya perlu melakukan *exhaustive search* sebanyak 32768 kali untuk mendapatkan bilangan yang diinginkan.

2 Input-Based Attack

Dengan *input-baesd attack*, PRNG ini sangat mudah untuk dikriptanalis. Hal ini disebabkan persamaan yang digunakan dalam PRNG ini hanya menghasilkan sebuah output yang tetap untuk sebuah bilangan bibit. Sebagai contoh jika kita menggunakan angka 100 sebagai seed maka 10 bilangan pertama yang dihasilkan secara berurutan adalah sebagai berikut:

- T[1] = 365
- T[2] = 1216
- T[3] = 5415
- T[4] = 16704
- T[5] = 24504
- T[6] = 11254
- T[7] = 24698
- T[8] = 1702
- T[9] = 23209
- T[10] = 5629

Salah satu cara untuk sedikit mempersulit kriptanalis melakukan penyerangan adalah dengan menggunakan waktu sebagai bibit untuk PRNG ini.

3 State Compromise Extension Attack

PRNG ANSI C tidak mampu merekonstruksi dengan baik jika kondisi internalnya diketahui informasinya. penyerang yang mengetahui informasi ini mampu melakukan kriptanalisis terhadap seluruh kondisi PRNG ANSI C tanpa perlu banyak usaha tambahan yang perlu dilakukan.

Kesimpulan yang diperoleh mengenai PRNG ANSI C:

- PRNG ini memilik banyak kelemahan dterhadap berbagai macam serangan
- Sedikit masukan mengenai bibit yang digunakan mungkin bisa digunakan fungsi terhadap waktu
- Bibit yang dimasukkan mungkin bisa berasal dari output sebelumnya

4.2 *Linear Congruential Generator* (LCG)

Pembangkit bilangan acak semu kongruen-lanjar (*linear congruential generator* atau LCG) adalah salah satu pembangkit bilangan acak tertua dan sangat terkenal. LCG didefinisikan dalam relasi rekurens:

$$x_n = (ax_{n-1} + b) \bmod m$$

yang dalam hal ini,

x_n = bilangan acak ke-n dalam deretnya

x_{n-1} = bilangan acak sebelumnya

a = factor pengali

b = *increment*

m = modulus

(a, b, dan m semuanya konstan)

Kunci pembangkti adalah x_0 yang disebut bibit (*seed*). Sebagai contoh kita memiliki fungsi pembangkit sebagai berikut:

$$x_n = (7x_{n-1} + 11) \bmod 17; X_0 = 0$$

sehingga rangkaian bilangan acak yang dihasilkan sebagai berikut:

n	x_n
0	0
1	11
2	3
3	15
4	14
5	7
6	9
7	6
8	2
9	8
10	16
11	4
12	5
13	12
14	10
15	13
16	**0**
17	**11**
18	**3**
19	**15**
20	**14**
21	**7**
22	**9**
23	**6**
24	**2**

Perhatikan bahwa setelah pembangkitan 16 buah bilangan, bilangan acak yang dihasilkan akan berulang kembali.

LCG mempunyai periode yang tidak lebih besar dari m, dan pada kebanyakan kasus periode bisa lebih pendek lagi. LCG mempunyai periode penuh (m-1) jika memenuhi syarat sebagai berikut:

1. b relatif prima terhadap m.
2. a-1 dapat dibagi dengan semua faktor prima dari m.
3. a-1 adalah kelipatan 4 jika m adalah kelipatan 4.
4. $m > \text{maks}(a, b, x_0)$.
5. $a > 0, b > 0$.

Meskipun LCG secara teoritis mampu menghasilkan bilangan acak yang tidak terlalu buruk, ia sangat sensitive terhadap pemilihan nilai-nilai a, b. dan m. Pemilihan nilai-nilai yang buruk dapat mengarah pada implementasi LCG yang tidak bagus. Tabel nilai konstanta yang bagus untuk LCG berdasarkan analisis dapat dilihat pada **tabel 1**.

a	b	m
106	1283	6075
211	1663	7875
421	1663	7875
430	2351	11979
936	1399	6655
1366	1283	6075
171	11213	53125
859	2531	11979
419	6173	29282
967	3041	14406
141	28411	134456
625	6571	31104
1541	2957	14000
1741	2731	12960
1291	4621	21870
205	29573	139968
421	17117	81000
1255	6173	29282
281	28411	134456

Tabel 1 Konstanta yang bagus untuk LCG

Keunggulan LCG terletak pada kecepatannya dan hanya membutuhkan sedikit operasi bit. LCG banyak digunakan untuk aplikasi-aplikasi simulasi karena kemangkusannya dan LCG memperlihatkan sifat statistik yang bagus dan sangat tepat untuk uji-uji empirik. Akan tetapi, LCG bisa diserang dengan macam-macam serangan sebagai berikut:

1. *Direct cryptanalytic attack*

   Dengan terjadinya pengulangan bilagnan acak yang dihasilkan, dapat disimpulkan bahwa PRNG ini menghasilkan output yang bisa dibedakan dengan bilangan yang benar-benar acak. Hal ini menyatakan bahwa PRNG LCG sangat mudah diserang dengan serangan semacam ini.

2. *Input-based attack*

   Jika penyerang mengetahui nilai bibit yang dijadikan input LCG, sangat mudahlah baginya untuk melakukan kriptanalisis terhadap PRNG ini. Hal ini disebabkan dengan mengetahui input, dia bisa memiliki persamaan matematika yang linear sehingga untuk mencari konstanta-konstanta yang tidak diketahui cukup melakukan exhaustive search dengan kompeksitas n saja. Bahkan, jika dia bisa melakukan *chosen-inpput attack*, hal ini semakin mempermudah saja karena dengan memilih input-input tertentu, penyerang bisa mengendalikan output-output yang memudahkannya. Sebagai contoh, jika kita memasukkan nilai sebagai input, dapat dipastikan bahwa nilai pertama yang muncul merupakan mc + b. Kemudian dia bisa melanjutkan dengan menggunakan nilai 1 sebagai bibit.

3. *State compromise attack*

   Penyerang yang berhasil melakukan perluasan status memiliki informasi mengenai nilai-nilai konstanta (semua atau sebagian) a, b, dan m untuk kemudian mengubahnya sesuai keinginannya. Seperti kita diketahui dalam persamaan aritmatika linear, nilai 1 dan 0 merupakan nilai identitas masing-masing untuk perkalian dan penjumlahan. Oleh karena, perluasan status yang dilakukan penyerang menggunakan nilai-nilai tersebut sangat memudahkannya untuk melakukan kriptanalisis terhadap PRNG LCG ini.

   a. *Permanent compromise attack*

      Dengan serangan ini, ktia bisa menurunkan kondisi internal menggunakan dua output. Asumsikan seorang kriptanalis berhasil mengetahui nilai m. Lebih jauh, setelah kondisi internal menjadi jauh berbeda dengan kondisi saat itu, dia memperoleh tiga output x_{i-1}, x_i dan x_{i+1}. Tujuan penyerang adalah mengetahui nilai a dan b. Dengan mengetahui informasi-informasi tersebut, penyerang mempunyai persamaan:

      $$\mathbf{x_i} = (a\mathbf{x_{i-1}} + b) \bmod \mathbf{m} \quad \text{(i)}$$

      $$\mathbf{x_{i+1}} = (a\mathbf{x_i} + b) \bmod \mathbf{m} \quad \text{(ii)}$$

      atau

      $$\mathbf{m}c + \mathbf{x_i} = a\mathbf{x_{i-1}} + b \quad \text{(i)}$$

      $$\mathbf{m}d + \mathbf{x_{i+1}} = a\mathbf{x_i} + b \quad \text{(ii)}$$

      (dengan yang dicetak tebal diketahui nilainya).

      Sehingga dengan melakukan grafik linear terhadap a, b, c, dan d penyerang bisa mendapatkan nilai-nilai tersebut dengan melakukan beberapa spekulasi jika tidak ada informasi tambahan mengenai nilai-nilai tersebut.

b. Iterative guessing attack

Jika penyerang mengetahui nilai x_i dan sebagian informasi mengenai fungsi yang digunakan dalam LCG, dia bisa menemukan nilai untuk x_{i+1} dengan melakukan beberapa spekulasi tekait nilai-nilai konstanta yang tidak dia ketahui. Hal ini didukung dengan keterbatasan nilai hasil yang tidak lebih besar dari nilai modulusnya.

c. Backtracking attack

Penyerang dapat merunut balik dengan sangat mudah seperti halnya pada *iterative guessing attack* dengan menggunakan serangan ini dengan mengasumsikan dia dapat menemukan fungsi untuk memperoleh output PRNG. Atau dengan cara lain, dia bisa mencari pasangan nilai-nilai yang diperoleh secara berurutan yang paling dekat dengan nilai output yang tidak dia ketahui yang ingin ia pelajari dan melakukan *permanent compromise attack*.

Kesimpulan yang bisa ditarik mengenai PRNG LCG ini antara lain:

- PRNG ini tidak cukup aman menghadapi berbagai macam serangan.
- PRNG LCG tidak bisa digunakan dalam aplikasi kriptografi yang *real-world* dikarenakan bilangan acaknya dapat diprediksi kemunculannya.
- Penggunaan pencacah waktu sebagai bibit juga tidak banyak membantu karena PRNG menangani inputnya tidak dengan baik sehingga serangan yang melibatkan input sangat rawan terhadapnya.

4.3 PRNG Blum Blum Shut

PRNG Blum Blum Shut adalah CSPRNG yang paling sederhana dan paling mangkus (secara kompleksitas teoritis). BBS dibuat pada tahun 1986oleh Lenore Blum, Manuel Blum, dan Michael Shub. Untuk membangkitkan bilangan acak dengan BBS, algoritmanya adalah sebagai berikut:

1. Pilih dua buah bilanga nprima rahasia p dan q yang masing-masing kongruen dengan 3 modulo 4 (dalam kenyataannya digunakan bilangan yang sangat besar untuk mempersulit kriptanalisis).
2. Kalikan kedua bilangan menjadi n = pq. Bilangan n ini disebut **bilangan bulat Blum**.
3. Pilih bilangan bulat acak lain, s, sebagai bibit sedemikian hingga:
   a. $2 \leq s < n$
   b. S dan n relatif prima

   kemudian hitung $x_0 = s^2 \bmod n$
4. Barisan bit acak dihasilkan dengan melakukan iterasi berikut sepanjang yang diinginkan:
   a. Hitung $x_i = x_{i-1}^2 \bmod n$
   b. z_i = bit LSB (*Least Significant Bit*) dari x_i

   Barisan bit acak adalah $z_1, z_2, z_3, \ldots$

Agar lebih jelasnya perhatikan contoh berikut ini. Misalkan bilangan-bilangan yang dipilih adalah p = 11 dan q = 7 sehingga n = pq = 77. Sedangkan s yang dipilih adalah s = 3 dan kita hitung $x_0 = 3^2 \bmod 77 = 9$. Barisan bit acak yang dihasilkan adalah sebagai tertulis secara berurutan di bawah ini:

$x_1 = x_0^2 \bmod n = 9^2 \bmod 77 = 4 \rightarrow z_1 = 0$

$x_2 = x_1^2 \bmod n = 4^2 \bmod 77 = 16 \rightarrow z_1 = 0$

$x_3 = x_2^2 \bmod n = 16^2 \bmod 77 = 25 \rightarrow z_1 = 1$

$x_4 = x_3^2 \bmod n = 25^2 \bmod 77 = 9 \rightarrow z_1 = 1$

$x_5 = x_4^2 \bmod n = 9^2 \bmod 77 = 4 \rightarrow z_1 = 0$

$x_6 = x_5^2 \bmod n = 4^2 \bmod 77 = 16 \rightarrow z_1 = 0$

$x_7 = x_6^2 \bmod n = 16^2 \bmod 77 = 25 \rightarrow z_1 = 1$

$x_8 = x_7^2 \bmod n = 25^2 \bmod 77 = 9 \rightarrow z_1 = 1$

Terlihat bahwa terjadi pengulangan setiap empat kali iterasi sehingga barisan bita aak yang dihasilkan adalah 001100110011…

Satu hal yang perlu diperhatikan dalam penggunaan PRNG Blum Blum Shut ini adalah bahwa kita tidak perlu melakukan iterasi untuk mendapatkan bilangan acak jika p dan q diketahui, sebab x_i dapat dihitung secara langsung menggunakan persamaan:

$$x_i = x_0^{2^i \bmod ((p-1)(q-1))} \bmod n$$

Bilangan acak yang dihasilkan tidak harus 1 bit LSB tetapi bisa juga k buah bit (k adalah bilangan bulat positif yang tidak melebihi $\log_2(\log_2 n)$). Agar lebih jelas perhatikan contoh berikut ini. Misalkan kita memilih p = 11351 dan q = 11987 sehingga n = pq = 136064437. kita pilih s = 80331757 dan k = 4 (k tidak melebihi $\log_2(\log_2 136064437) = 4.75594$). Kita hitung $x_0 = 80331757^2 \bmod 136064437 = 1312737111$. Barisan bit acak yang dihasilkan adalah sebagai berikut:

- $x_1 = x_0^2 \bmod n$
  $= 131273718^2 \bmod 136064437$
  $= 47497112$
  $z_i = 47497112 \equiv 8 \bmod 2^4$
  $= 1000_{\text{basis 2}}$
  (4 bit LSB dari 47497112)
- $x_2 = x_1^2 \bmod n$
  $= 47497112^2 \bmod 136064437$
  $= 69993144$
  $z_i = 69993144 \equiv 8 \bmod 2^4$
  $= 1000_{\text{basis 2}}$
  (4 bit LSB dari 69993144)
- $x_3 = x_2^2 \bmod n$
  $= 69993144^2 \bmod 136064437$
  $= 13801821$
  $z_i = 13801821 \equiv 5 \bmod 2^4$
  $= 0101_{\text{basis 2}}$
  (4 bit LSB dari 69993144)
  ……

Barisan bit acak yang dihasilkan:

1000 1000 0101 …

atau dalam basis 10:

8 8 5 …

Sepintas terlihat bahwa PRNG Blum Blum Shut ini sempurna untuk digunakan dalam aplikasi yang memerlukan bilangan acak sebagai elemennya. Namun, ternyata PRNG ini bisa juga bisa diserang. Serangan-serangan yang mungkin dilakukan terhadap pembangkit bilangan acak semu Blum Blum Shut ini antara lain:

1. *Direct cryptanalytic attack*

   Jika nilai k yang digunakan untuk menentukan seberapa panjang bit yang dihasilkan tidak terlalu panjang, PRNG ini memiliki kesamaan sifat dengan LCG pada pengulangannya. Dengan terjadinya pengulangan bilangan acak yang dihasilkan, dapat disimpulkan bahwa PRNG ini menghasilkan output yang bisa dibedakan dengan bilangan yang benar-benar acak. Hal ini menyatakan bahwa PRNG LCG sangat mudah diserang dengan serangan semacam ini.

   Akan tetapi, pada prakteknya bilangan yang digunakan baik p, q, ataupun k selalu bilangan yang sangat besar sehingga menyulitkan penyerang jika melakukan exhaustive search. Penggunaan bilangan besar ini juga berarti semakin mempersulit penyerang untuk membedakan output yang dihsilkan dengan bilangan yang benar-benar acak. Hanya saja, perlu diperhatikan masalah performa ketika proses berjaklan jika digunakan bilangan yang besar. Hal ini yang sedikit mengurangi kelebihan algoritma ini dibanding algoritma lain yang cukup menggunakan bilangan yang tidak terlalu besar untuk menghailkan outputnya.

2. *Input-based attack*

   Jika penyerang mengetahui nilai p dan q yang dijadikan input BBS, sangat mudahlah baginya untuk melakukan kriptanalisis terhadap PRNG ini. Hal ini disebabkan dengan mengetahui nilai-nilai tersebut, dia bisa cukup melakukan pencarian sebanyak n (pq) kali untuk mendapatkan nilai s. Dari sekian banyak kandidat s yang ada, nilai yang memenuhi persamaan untuk mendapatkan x_0 adalah nilai s yang digunakan dalam pembangkitan tersebut. Jika penyerang mampu melakukan *chosen-input attack*, tidak ada kesulitan berarti dalam menemukan nilai s karena dia cukup menggunakan nilai p dan q yang kecil saja untuk mempermudah pencarian secara *brute force*. Sebagai contoh dengan menggunakan p = 2 dan q = 3 dia hanya memiliki *range* pencarian sebanyak 6 saja [2...5].

3. *State compromise attack*

   Penyerang yang berhasil melakukan perluasan status memiliki informasi mengenai nilai-nilai konstanta (semua atau sebagian) p, q, dan s untuk kemudian

mengubahnya sesuai keinginannya. Kita misalkan seorang kriptanalis mampu memperluas seluruh status internal dari PRNG tetapi kehilangan jejak dari input dan outputnya dalam waktu yang lama. Jika interval kondisi yang dimasukkan cukup besar, maka PRNG BBS ini memiliki ketahan yang cukup besar terhadap serangan jenis ini. Perhatikan bahwa interval kondisi sesuai dengan seberapa besar nilai p dan q yang digunakan.

a. Permanent compromise attack

Dengan serangan ini, kita bisa menurunkan kondisi internal menggunakan dua output. Asumsikan seorang kriptanalis berhasil mengetahui nilai p dan q. Lebih jauh, setelah kondisi internal menjadi jauh berbeda dengan kondisi saat itu, dia memperoleh tiga output x_{i-1}, x_i dan x_{i+1}. Tujuan penyerang adalah mengetahui nilai s. Akan tetapi, nilai-nilai yang diperoleh oleh kriptanalis hanya LSB dari hasil perhitungan yang dilakukan sehingga sangat sulilah baginya untuk mencari nilai s. Hanya saja akan lain ceritanya jika yang diperoleh oleh penyerang adalah fungsi untuk mendapatkan LSB tersebut. Tanpa perlu mengetahui nilai p dan q dia sudah dapat mencari barisan bilangan-bilangan tersebut karena dalam perhitungannya p dan q tidak dilibatkan setelah nilai n didapatkan.

b. Iterative guessing attack

Jika penyerang mengetahui nilai x_i dan sebagian informasi mengenai fungsi yang digunakan dalam BBS, dia bisa menemukan nilai untuk x_{i+1} dengan melakukan beberapa spekulasi tekait nilai-nilai konstanta yang tidak dia ketahui. Hal ini didukung dengan sedikitnya parameter yang digunakan untuk menghasilkan barisan bilangan acak tersebut.

c. Backtracking attack

Penyerang dapat merunut balik dengan sangat mudah seperti halnya pada *iterative guessing attack* dengan menggunakan serangan ini dengan mengasumsikan dia dapat menemukan fungsi untuk memperoleh output PRNG. Sekali lagi perlu ditegaskan bahwa nilai output yang diperoleh haruslah hasil perhitungan sebelum diambil LSB-nya. Jika tidak, maka sangat sulit untuk melakukan pencarian balik.

Kesimpulan yang bisa ditarik mengenai PRNG Blum Blum Shut ini:

- Perlu diperhatikan penggunaan nilai p dan q yang besar untuk mempersulit kriptanalisis. Akan tetapi, jangan dilupakan fakta bahwa penggunaan nilai yang besar mempengaruhi performa proses PRNG ini.
- PRNG ini cukup kuat menerima serangan yang melibatkan perluasan status selama fungsi untuk mendapatkan nilai yang akan diambil LSB-nya belum diketahui oleh penyerang. Kebocoran fungsi untuk mendapatkan nilai tersebut akan membuat PRNG hancur karena tidak ada elemen proteksi lain untuk menghadapinya.
- Untuk serangan yang melibatkan input, PRNG ini sangat lemah karena dengan pengaturan nilai input dapat diperoleh informasi untuk mendapatkan output yang sesuai point sebelumnya, sangat rawan terhadap PRNG ini.

4.4 PRNG DSA

Spesifikasi Digital Signature Standard mendeskripsikan sebuah PRNG sederhana yang didasarkan pada SHA (atau konstruksi DES) yang ditujukan untuk membangkitkan parameter bilangan acak semu untuk algoritma tanda tangan DSA. Karena pembangkit bilangan acak semu ini muncul berdasarkan pengakuan dari NSA, banyak yang menggunakan dan mengusulkan pembangkit ini secara berbeda dengan apa yang awalnya didesain.

PRNG DSA memungkinkan pilihan input sesuai keinginan user ketika membangkitkan bilangan, tetapi tidak ketika membangkitkan parameter tanda tangan digital DSA. Akan tetapi, kita mengasumsikan input dari user ini bisa dilakukan kapan pun seperti halnya PRNG lain yang dibahas dalam malakah ini. Setiap kali pembangkit bilangan acak semu DSA membangkitkan output, PRNG ini bisa disisipkan input opsional M_i. Perlu diketahui bahwa kegagalan melakukan input opsional dalam desain sebuah PRNG memastikan bahwa

PRNG tersebut tidak mampu pulih dari perluasan status.

Semua fungsi aritmatika dalam PRNG DSA ini memungkinkan dilakukan dalam modulo 2^N, dimana $160 \leq N \leq 512$. Sebagai pengingat dalam makalah ini, kita asumsikan bahwa modulus dilakukan dengan factor sama dengan 160, karena ini adalah nilai paling lemah yang dimungkinkan oleh desain.

Secara umum algoritma PRNG DSA adalah sebagai berikut:

1 PRNG mempertahankan kondisi X_i yang bisa berubah sewaktu-waktu.

2 PRNG menerima input opsional W_i yang diasumsikan nol jika tidak dimasukkan.

3 PRNG membangkitkan setiap input sebagai berikut:

a) $T[i] = \text{hash}(W_i + X_i \bmod 2^{160})$

b) $X_{i+1} = X_i + T[i] + 1 \pmod{2^{160}}$

Serangan-serangan yang mungkin dilakukan terhadap PRNG ini:

1. Direct cryptanalytic attack

Jika fungsi hash yang digunakan dalam PRNG ini cukup baik, maka urutan output yang dihasilkan tampaknya sulit untuk dibedakan dengan urutan bilangan acak. Akan sempurna jika desainer PRNG ini memiliki bukti kualitas output yang dihasilkan PRNG ini berdasarkan ketahan terhadap terjadinya kolisi dalam fungsi hash satu arah.

2. Input-based attack

Dalam serangan ini, kita asumsikan penyerang mmapu mengendalikan input yang masuk ke PRNG melalui status W. Jika hal ini mampu direalisasikan, maka ada cara sederhana yang bisa memaksa output yang dihasilkan akan berulang terus menerus. Hal ini sangat terasa jika dilakukan terhadap PRNG yang memungkinkan penyerang mengatur interval input yang dimasukkan ke dalam PRNG. Untuk itu, penyerang cukup mengatur agar status W ditentukan sebagai berikut:

$$W_i = W_{i-1} - T[i-1] - 1 \pmod{2^{160}}$$

Persamaan ini memaksa nilai bibit yang digunakan untuk membangkitkan bilangan berikutnya terus berulang yang otomatis membuat output yang dihasilkan berulang pula. Namun, serangan ini akan menemui kegagalan jika dalam mekanisme PRNG tersebut, bibit yang dimasukkan terlebih dulu diubah menjadi nilai hashnya sebelum dimasukkan ke dalam kondisi W. Berhubung kebanyakan sistem melakukan hal tersebut, maka hanya ada sedikit sistem yang rawan terhadap serangan semacam ini.

3. State Compromise Extension Attack

PRNG DSA tidak menangani perluasan status sebagaimana kita inginkan tetapi PRNG ini lebih baik daripada PRNG ANSI X9.17. Kita misalkan seorang kriptanalis mampu memperluas seluruh status internal dari PRNG tetapi kehilangan jejak dari input dan outputnya dalam waktu yang lama. Jika interval kondisi yang dimasukkan cukup besar, maka PRNG DSA ini memiliki ketahan yang cukup besar terhadap serangan jenis ini.

a. Kelemahan input

PRNG DSA memiliki kesamaan dengan ANSI X9.17 dalam hal kekurangtahanan terhadap input yang tidak dapat diterka dalam outputnya. Perhatikan seorang kriptanalis yang mampu melakukan perluasan status PRNG. Aplikasi memasukkan bibit yang tidak dapat diterka oleh kriptanalis tersebut. Jika kriptanalis memperhatikan output selanjutnya, dia tidak menerka sample yang digunakan karena output selanjutnya yang data dihasilkan oleh sample ini adalah output tersebut. Pehatikan bahwa jika X_{i+1} yang baru tergantung penuh terhadap W_i dan X_i, kelemahan ini tidak akan muncul. Penyerang yang mengetahui kondisi tersebut tetap dapat menerka sample, tetapi jika dia tidak menerka dengan benar dia akan kehilangan jejak dari status itu.

b. Iterative guessing attack

PRNG ini rawan terhadap i*terative guessing attack* setelah status berhsil diperluas, yaitu jika penyerang mengetahui X_i dan mengetahui juga bahwa W_i hanya memiliki 20 bit entropi. Dia hanya perlu melakukan pencarian sebanyak 2^{20} kali dan memilik daftar output 160 bit sebanyak 2^{20}, satu di antaranya T[i]. Perhatikan bahwa penyerang hanya membutuhkan fungsi

keluaran yang bisa dia cek, seperti tanda tangan digital DSA dengan menggunakan T[i] sebagai nilai parameter rahasianya. Perlu diperhatikan juga bahwa informasi mengenai sebuah output tunggal T[i] menentukan secara unik nilai T[i+1].

c. Backtracking attack

Jika penyerang mengetahui nilai X_i dan T[i-1], maka dia bisa merunut balik informasi mengenai T[i-1]. Hal ini tidak secara langsung memberikan banyak hasil terhadapnya karena dia harus terlebih dahulu mengetahui nilai T[i-1] untuk melakukan ini. Akan tetapi, dalam beberapa kondisi hal ini akan sangat berguna.

d. Menutup celah

Perhatikan situasi dimana kriptanalis mengetahui X_i, X_{i+2} dan T[i+1] tetapi tetap memerlukan T[i]. Dalam situasi ini, dia dapat menemukan nilai T[i] dengan persamaan:

$$T[i] = X_{i+2} - X_i - 2 - T[i+1]$$

Kesimpulan yang bisa diambil dari PRNG DSA ini sebagai berikut:

- PRNG DSA standar tampaknya cukup aman jika digunakan dalam sistem yang memang tujuan pendesainan PRNG ini, yaitu tanda tangan digital DSA
- PRNG DSA tidak cukup baik seperti halnya PRNG kriptografi yang ditujukan untuk penggunaan universal karena PRNG ini tidak cukup baik dalam hal penanganan input yang digunakannya.
- Untuk mengadaptasi PRNG DSA agar bisa digunakan untuk tujuan universal, tidak hanya algoritma tanda tangan digital DSA saja, perubahan di bawah ini bisa dilakukan untuk mencegah serangan-serangan seperti yang dibahas di atas:
  - Melakukan hash terhadap semua input PRNG sebelum dimasukkan sebagai sample
  - Modifikasi X dengan menggunakan persamaan:

$$X_{i+1} = X_i + \text{hash}(T[i] + W_i) \bmod 2^{160}$$

4.5 PRNG ANSI X9.17

PRNG ANSI X9.17 ditujukan sebagai mekanisme untuk menghasilkan kunci DES dan vector insialisasi menggunakan fungsi primitif triple-DES. Dalam kesempatan lain triple-DES juga bisa diganti dengan algoritma cipher block yang lain. algoritma ini sudah digunakan sebagai PRNG yang bisa digunakan untuk aplikasi dengan berbagai tujuan.

Secara umum, hal yang perlu diperhatikan mengenai PRNG ini adalah seperti yang tertera di bawah ini:

1. K adalah kunci rahasia triple-DES yang dibangkitkan dengan suatu cara tertentu pada saat inisialisasi. Kunci ini harus acak dan hanya digunakan oleh pembangkit ini. Kunci ini merupakan bagian dari kondisi rahasia PRNG yang tidak pernah diubah oleh input PRNG manapun.
2. Setiap pembangkitan output dilakukan hal sebagai berikut:
   a. $S_i = E_K(t)$
   b. $T[i] = E_K(S_i \oplus \text{seed}[i])$
   c. $\text{seed}[i+1] = E_K(S_i \oplus T[i])$

PRNG ini digunakan dalam berbagai macam aplikasi.

Serangan-serangan yang mungkin dilakukan terhadap PRNG ini:

1. Direct cryptanalytic attack

Kriptanalisis secara langsung terhadap PRNG ini membutuhkan kriptanalisis terhadap triple-DES atau algoritma cipher blok lain yang menggunakan PRNG ini. Sejauh ini, belum ada penyataan dari pakar kriptanalisis yang menyatakan kemungkinan jenis serangan ini terhadap PRNG X9.17

2. Input-based attack

Dengan mengasumsikan ukuran blok adalah 64 bit, PRNG ini memilik kelemahan terhadap serangan semacam ini. Seorang kriptanalis yang bisa memaksa nilai S sedemikian hingga nilai tersebut tetap untuk setiap pembangkitan, dapat membedakan output PRNG dengan bilangan acak setelah memperhatikan 64 bit output sebanyak 2^{32}

buah. Dalam keterurutan bilangan acak 64 bit, kita mengharapkan terjadinya kolisi setelah dibangkitkan 2^{63} bilangan. Namun, dengan pengondisian S sedemikian hingga nilainya tetap, kolisi diharapkan terjadi setelah 2^{32} pembangkitan.

3. State Compromise Extension Attack

Seperti halnya PRNG DSA, PRNG X9.17 tidak mampu pulih secara penuh dari perluasan status. Kondisi ini bisa terjadi jika penyerang mampu melakukan perluasan status kunci triple-DES, K, sehingga secara otomatis dia bisa melakukan perluasan terhadap seluruh status internal PRNG tanpa melakukan usaha tambahan.

Dalam PRNG ANSI X9.17 terhadap dua cela yang hanya muncul jika PRNG ini dianalisis menggunakan serangan ini:

1) Hanya 64 bit status PRNG, seed[i] yang bergantung pada nilai input. Hal ini berarti jika pada suatu saat penyerang berhasil melakukan perluasan terhadap K, PRNG tidak dapat merekonstruksi secara penuh meski setelah melakukan pengolahan beberapa input penyerang tidak mampu menerkanya.

2) Nilai see[i+1] adalah fungsi yang menggunakan output, T_i, dan K sebelumnya sebagai parameter. Penyerang yang mengetahui K dari perluasan status sebelumnya dan properti-properti dasar satuan waktu yang digunakan untuk menghasilkan T_i, seed[i+1] tidaklah sulit untuk diterka.

a. Permanent compromise attack

Dengan serangan ini, ktia bisa menurunkan kondisi internal menggunakan dua output. Asumsikan seorang kriptanalis berhasil mengetahui nilai K. Lebih jauh, setelah kondisi internal menjadi jauh berbeda dengan kondisi saat itu, dia memperoleh dua output T[i] dan T[i+1]. Tujuan penyerang adalah mengetahui nilai seed[i+1] yang bisa saja dilakukan dengan melakukan 64 bit *exhaustive search* dan memperoleh nilai seed tersebut.

Akan tetapi, ada cara yang jauh lebih efektif untuk menerapkan serangan ini terhadap PRNG. Asumsikan setiap nilai satuan waktu terdiri dari 10 bit yang tidak diketahui oleh penyerang. Asumsi ini adalah asumsi yang umum digunakan dalam banyak sistem. Sebagai contoh, perhatikan pencatat waktu dalam satuan milidetik dan seorang penyerang yang mengetahui waktu terdekat ketika bilangan dibangkitkan. Seorang penyerang yang memiliki dua output PRNG secara berurutan dapat melakukan *meet-in-the-middle attack* untuk mengetahui nilai status internal yang membutuhkan sekitar 2^{11} percobaan enkripsi dengan menggunakan kunci K. hal ini dapat terjadi karena kita memliki persamaan sebagai berikut:

$$seed[i+1] = D_K(T[i+1]) \oplus T_{i+1}$$

$$seed[i+1] = E_K(T[i] \oplus T_i)$$

Penyerang mencoba semua nilai yang mungkin untuk T_i, dan mendaftarkan semua kemungkinan nilai seed[i+1]. Kemudian dia mencoba semua kemungkinan nilai T_{i+1} dan membuat daftar lain untuk kemungkinan nilai seed[i+1]. Nilai seed[i+1] yang benar adalah yang muncul di kedua daftar kemungkinan tersebut.

b. Iterative guessing attack

Jika penyerang mengetahui nilai seed[i] dan melihat beberapa fungsi T[i+1], dia dapat mengetahui nilai seed[i+1] pada hampir semua kasus. Hal ini dapat dibuktikan karena pencacah waktu umunya memiliki entropi yang kecil. Menggunakan asumsi awal kita bahwa setiap sample pencacah waktu terdiri dari 10 bit, berarti penyerang hanya perlu menebak 10 bit bilangan. Perlu diperhatikan bahwa penyerang hanya perlu melihat fungsi keluaran, bukan keluaran itu sendiri. Hal ini berarti sebuah pesan yang dienkripsi menggunakan kunci yang dihasilkan dari nilai output cukup untuk menjalankan serangan ini. Perhatikan bahwa perbedaan dengan *permanent compromise attack* adalah bahwa *permanent compromise attack* membutuhkan nilai output sedangkan *iterative guessing attack* hanya memerlukan fungsinya.

c. Backtracking attack

Penyerang dapat merunut balik semudah melihat ke depan dengan serangan ini dengan mengasumsikan dia dapat menemukan fungsi untuk memperoleh output PRNG. Atau dengan cara lain, dia bisa mencari pasangan nilai-nilai yang diperoleh secara berurutan yang paling dekat dengan nilai output yang tidak dia ketahui yang ingin ia pelajari dan melakukan *permanent compromise attack*.

d. Meet-in-the-middle attack

Kadang-kadang, sebuah PRNG bisa membangkitkan nilai rahasia yang sangat besar dan tidak menghasilkan output yang terkait langsung dengan nilai tersebut. Misalnya penyerang mengetahui nilai seed[i] dan seed[i+8] tapi tidak mengetahui nilai-nilai di antaranya. Karena hal ini menyebabkan dia memiliki misalnya 80 bit entropi, sangat naïf mengasumsikan dia tidak bisa merekonstruksi nilai output itu. Namun, hal itu tidak masalah dengan *meet-in-the-middle attack*. Berikut adalah langkah-langkahnya:

1) Penyerang melakukan serangan sebagaimana dijelaskan di atas untuk mempelajari status PRNG sebelum dan sesudah eksekusi kedua nilai yang digunakan bersama.

2) Penyerang melakukan *meet-in-the-middle attack*, menghasilkan satu set nilai-nilai yang mungkin untuk seed[i+4] dengan menebak $T_{i+1..i+4}$ dan menurunkan daftar kedua dengan menebak $T_{i+5..i+8}$. Jika setiap sekuens 4 satuan waktu terdiri dari 40 bit entropi, maka dibutuhkan 2^{41} percobaan. Nilai yang benar dari seed[i+4] akan muncul di kedua daftar.

3) Penyerang mencoba semua kemungkinan sekuens output sampai dia menemukan output yang benar. Sebagai contoh, jika delapan blok output digunakan sebagai kunci enkripsi, sebanyak 2^{16} percobaan perlu dilakukan untuk menghasilkan nilai yang benar itu.

Dalam diskusi di atas, kita mengasumsikan sebuah nilai input PRNG tunggal memenuhi kebutuhan entropi tertentu dan sehingga membutuhkan jumlah tertentu pula untuk melakukan terkaan. Pada prakteknya, hal ini tidak selalu berlaku. Sebuah pembangkitan pasangan kunci RSA wajarnya menggunakan dua buah bilangan acak semu berukuran 512 bit sebagai nilai awalnya sehingga membutuhkan 16 output dari PRNG. Namun, pemanggilan ini hampir dapat dipastikan dilakukan secara cepat. Kecuali satuan waktu dimana nilai T_i didasarkan pada presisi yang tinggi, kebanyakan nilai T_i ini akan didasarkan pada nilai satuan waktu yang sama atau sangat dekat. Hal ini membuat serangan *meet-in-the-middle attack* bisa diterapkan meskipun umunya masuk akal untuk memperkirakan paling tidak 3 bit yang tidak dapat diprediksi setiap satuan waktu.

Kesimpulan yang diperoleh dari pembahasan di atas antara lain:

- PRNG ANSI X9.17 kelihatannya aman dari segala macam serangan kriptanalis yang tidak melibatkan penghentian pencacah waktu atau perluasan status internal kunci triple-DES.
- Melakukan usaha pengulangan input pencacah waktu sebanyak 2^{32} mengakibatkan munculnya kelemahan: cara membedakan banyaknya output PRNG X9.17 dengan bilangan yang benar-benar acak.
- Melakukan perluasan kunci triple-DES dapat menghancurkan PRNG X9.17 secara total. PRNG ini tidak mampu merekonstruksi meskipun setelah mendapatkan ribuan bit entropi dalam sample input satuan waktu.
- Untuk sistem yang menggunakan PRNG ini cara paling jelas untuk mencegah serangan semacam ini adalah menggunakan status X9.17 saat ini untuk membangkitkan semua status X9.17 yang baru termasuk K yang baru dan bibit awal seed[0].

5 Kesimpulan dan Saran

Dalam makalah ini telah dijelaskan bagaimana menentukan kebutuhan yang diperlukan untuk merancang sebuah PRNG, mengembangkan abstraksi serangan-serangan yang mungkin

dilakukan terhadap PRNG yang ideal dan mendeskripsikan rincian serangan-serangan tersebut terhadap sistem PRNG di dunia kriptografi saat ini. Oleh karena itu, dapat disimpulkan beberapa cara yang bisa dijadikan panduan untuk menggunakan PRNG:

1. Gunakan fungsi hash untuk menjamin keamanan output PRNG.

   Menggunakan fungsi hash berarti menghilangkan jejak output yang bisa diterka oleh penyerang. Selain itu, fungsi hash yang melakukan pemetaan satu arah(irresversible) membuat penyerang tidak mungkin melakukan analisa dari output yan dihasilkan. Akan tetapi, tidak semua celah yang terdapat dalam PRNG itu bisa tertutupi dengan penggunaan fungsi hash untuk melakukan *preprocessing* terhadap output.

2. Gunakan fungsi hash terhadap input yang akan dijadikan bibit.

   Melakukan hashing terhadap input membuat penyerang tidak bisa melakukan serangan *input-based attack.* Hal ini dikarenakan penggunaan fungsi hash terhadap input membuat penyerang tidak mengetahui nilai input yang sebenarnya yang masuk ke dalam proses. Melakukan hashin dapat dilakukan dengan menggunakan pencacah waktu karena nilai pencacah waktu selalu berubah-ubah. Jika hal ini tidak mungkin dilakukan mengingat usaha yang dibutuhkan untuk melakukan ini tidak mudah, dapat dilakukan dengan cara lain yang intinya mencoba menipu penyerang dengan fungsi hashing ini.

3. Membangkitkan ulang status PRNG

   Jika status PRNG selalu tetap setiap dilakukan pembangkitan, penyerang yang bisa melakukan perluasan status memiliki kemudahan untuk menyerang PRNG tersebut. Dengan mengubah status PRNG sewaktu-waktu, hal ini bisa dicegah. Untuk sebuah PRNG yang statusnya hampi selalu tidak dapat diubah jika sudah diinisialisasi, melakukan perubahan status menjamin PRNG itu memiliki waktu tambahan untuk melakukan rekonstruksi ulang jika penyerang sudah bisa melakukan perlusan status.

4. Mencegah terjadinya perluasan status

   Cara terbaik untuk mencegah penyerang bisa melakukan perluasan status adalah dengan tidak menyediakan kesempatan baginya untuk melakukan hal tersebut. Dengan membuat input yang tidak dapat diterka, hampir dapat dipastikan penyerang tidak dapat melakukan hal itu. Hal ini bisa dilakukan dengan membuat input atau bibit sedemikian hingga sangat jarang sistem yang menggunakan input tersebut atau memiliki pola tertentu.

Selain itu, dapat juga ditarik beberapa masukan yang dapa digunakan panduan untuk mendesain sebuah PRNG.

1. Pastikan status PRNG selalu berubah.

   Dengan mengubah status internal PRNG hampir dapat dipastikan serangan dengan jenis *state compromise attack* dan variannya tidak dapat menyerang PRNG. Hal ini disebabkan dengan perubahan status setiap kali akan dibangkitkan output baru, perluasan status yang dilakukan oleh penyerang tidak akan berlaku untuk *backward* atau *forward* sehinga sistem bisa merekonstruksi ulang.

2. Belajar dari kesalahan

   Dengan memanfaatkan informasi yang diperoleh dari sistem PRNG yang bsa ditembus oleh penyerang, PRNG baru yang akan dirancang paling tidak sudah melakukan pencegahan agar hal yang sama tidak berulang terhadapnya.

3. Melakukan pembibitan ulang

   Bagian status internal yang diperlukan untuk melakukan pembangkitan output baru hendaknya berbeda dengan yang akan digunakan untuk melakukan pembibitan ulang. Hal ini diperlukan agar status internal yang dkumpulkan memiliki entropi yang cukup untuk mencegah terjadinya serangan *iterative guessing attack.*

4. Mencegah *backtracking*.

   Mencegah terjadinya serangan dengan melakukan runut balik artinya mencegah penyerang untuk bisa melakukan terkaan terhadap suatu status internal t jika dia berhasil melakukan perluasan status (yang bisa dicegah dengan langkah nomor 1 sebelumnya) ketika status internal t+1. hal ini juga bisa dicegah dengan mendesain PRNG yang melakukan pembangkitan

dengan menggunakan fungsi satu arah, seperti halnya fungsi hash.

5. Cepat merekonstruksi ulang jika terjadi perluasan status.

   PRNG seharusnya memanfaatkan setiap bit input untuk prosesnya. Hal ini diperlukan agar penyerang yang melakukan perluasan status harus mendapatkan keseluruhan bit input agar dia bisa melakukannya. Dengan demikian, jika terjadi perluasan status PRNG bisa dengan cepat melakukan rekonstruksi.

6. Membuat sekuens input yang tidak dapat diterka.

   Dengan melakukan kombinasi kumpulan input dan status internal yang dimiliki oleh PRNG, diusahakan sekuens input baru yang dihasilkan tidak dapat diterka oleh penyerang. Hal ini mengakibatkan seorang penyerang harus mengetahui kedua bagian PRNG yaitu status internal dan inputnya/bibit untuk melakukan terkaan sekuens input. Hal ini sedikit membantu dalam mencegah terjadinya *input-based* dan *state compromise extension attack*.

Referensi

[1] Bishop,David, *Introduction to Cryptography with Java Applet*, Jones and Bartlet Computer Science, 2003

[2] Munir, Rinaldi, *Diktat Kuliah IF5054 Kriptografi*, Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, 2006

[3] **http://www.cacr.math.uwaterloo.ca/hac/**

[4] **http://eprint.iacr.org/**

[5] **http://cryptographyperlinkcz/2004/**

[6] http://random.mat.sbg.ac.at/ftp/pub/software/gen/prng-3.0/doc/prng.html