Implementasi Teorema Coppersmith dalam Serangan Terhadap *Broadcast* Kriptosistem RSA

Naufal Dean Anugrah 13518123[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13518123@std.stei.itb.ac.id

***Abstrak*—Kriptografi adalah suatu bidang yang mempelajari cara untuk mengamankan proses pertukaran informasi. Salah satu algoritma kriptografi yang banyak digunakan saat ini adalah RSA (Rivest–Shamir–Adleman). Algoritma RSA memanfaatkan fakta bahwa proses faktorisasi suatu bilangan besar memiliki tingkat kesulitan yang tinggi. Secara umum jika diterapkan dengan baik, *ciphertext* yang dihasilkan oleh proses enkripsi RSA tidak bisa dipecahkan secara cepat tanpa kunci yang tepat, setidaknya untuk kemampuan komputasi yang ada saat ini. Akan tetapi, dalam beberapa implementasi yang kurang baik, mungkin terdapat celah yang bisa diekploitasi. Salah satunya adalah dalam proses *broadcast* (penyiaran) suatu pesan ke banyak penerima. Kesalahan tersebut bisa dieksploitasi memanfaatkan Teorema Coppersmith untuk melakukan proses pemulihan *plaintext*.**

***Kata kunci—Broadcast,* Kriptografi, RSA, Teorema Coppersmith.**

I. PENDAHULUAN

Keamanan informasi merupakan salah satu hal yang sangat krusial di dunia digital saat ini. Dengan informasi, suatu pihak bisa menyebarkan perdamaian, maupun menciptakan perang besar. Oleh karena itu, proses penjagaan informasi harus dilakukan dengan cermat sehingga tidak jatuh ke tangan orang-orang yang tidak bertanggung jawab.

Kriptografi adalah salah satu cabang ilmu yang mempelajari mengenai cara untuk mengamankan informasi. Terdapat berbagai cara untuk melakukan proses tersebut, baik menggunakan kunci maupun tanpa kunci, *reversible* maupun *nonreversible*. Semua itu dengan tujuan agar pesan yang dikirimkan sampai pada tujuan yang tepat. Kriptografi merupakan ilmu yang berkembang dengan pesat. Dapat dilihat dalam sejarah, beberapa contoh algoritma enkripsi klasik yang cukup terkenal adalah Caesar cipher, Vigenere cipher, Morse, dll. Dengan berkembangnya teknologi, algoritma klasik menjadi tidak relevan. Saat ini, terdapat beberapa algoritma enkripsi yang banyak dipakai, yaitu RSA, AES, dan RC4. Berbagai algoritma ini pun kelak akan tergantikan, misalkan dengan merebaknya komputasi kuantum.

RSA adalah salah satu algoritma enkripsi yang kuat dan masih banyak digunakan saat ini. Algoritma yang diciptakan oleh Ron Rivest, Adi Shamir, dan Leonard Adleman pada tahun 1977 ini termasuk ke dalam algoritma enkripsi asimetris, yaitu kunci yang digunakan dalam proses enkripsi berbeda dengan kunci yang digunakan untuk melakukan proses dekripsi. Secara umum, hampir tidak ada celah dalam algoritma RSA yang bisa dieksploitasi menggunakan teknologi saat ini. Hal ini disebabkan karena, RSA berdasar pada fakta bahwa pemfaktoran bilangan integer besar merupakan suatu proses yang sulit dan memakan banyak waktu dalam komputer klasik.

Permasalahan utama umumnya terdapat pada kesalahan implementasi RSA yang membuka celah. Sehingga pesan yang dikirimkan bisa didapatkan kembali, bahkan terkadang tanpa perlu mengetahui kunci untuk melakukan proses dekripsi. Beberapa kesalahan implementasi yang cukup fatal adalah penggunaan nilai modulus N yang terlalu kecil, nilai *private exponent* d yang terlalu kecil, maupun nilai *public exponent* e yang terlalu kecil. Teorema Coppersmith adalah salah satu teorema yang banyak digunakan untuk menyerang implementasi RSA dengan nilai *public exponent* yang terlalu kecil. Teorema ini banyak menjadi dasar dari berbagai serangan lain, salah satunya adalah *broadcast attack* yang diciptakan oleh Hastad. Dengan diberikan jumlah pasangan modulus N dan *ciphertext* C untuk suatu *plaintext* M yang sama, seorang pihak luar dapat melakukan proses dekripsi terhadap pesan yang dikirimkan.

II. DASAR TEORI

A. Kriptografi

Selama sejarah, kriptografi sudah berkembang dengan pesat. Setiap masa, memiliki sistem kriptografi tersendiri. Sistem tersebut pun punya waktu hidup, sebelum diambil alih oleh perkembangan teknologi. Saat ini, kriptografi modern umumnya didasarkan pada beberapa hal. Pertama, aritmatika modulo, misalkan saja di algoritma RSA. Kedua, aljabar boolean, misalkan saja di berbagai mode enkripsi dalam algoritma AES maupun RC4 yang berdasarkan pada operasi XOR. Ketiga, lattice, penggunaan lattice dalam kriptografi memang belum terlalu luas, akan tetapi menjanjikan mengingat resistansinya terhadap serangan, baik oleh komputer klasik maupun modern.

Terdapat beberapa istilah penting yang sering digunakan dalam bidang kriptografi, yaitu:

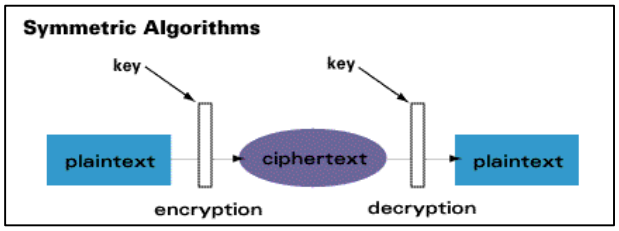
1. *Plaintext* adalah pesan asli yang akan dikirimkan oleh pihak pengirim ke pihak penerima. *Plaintext* biasa dilambangkan dengan karakter M.
2. *Ciphertext* adalah pesan asli yang telah dienkripsi menggunakan suatu algoritma. *Ciphertext* biasa

dilambangkan dengan karakter C.

3. Enkripsi adalah proses mengubah *plaintext* menjadi *ciphertext*.
4. Dekripsi adalah proses mengubah *ciphertext* kembali menjadi *plaintext*.
5. *Cipher* adalah suatu set aturan untuk melakukan proses enkripsi dan dekripsi.
6. *Key* atau kunci adalah suatu nilai yang digunakan untuk melakukan proses enkripsi dan/atau dekripsi.

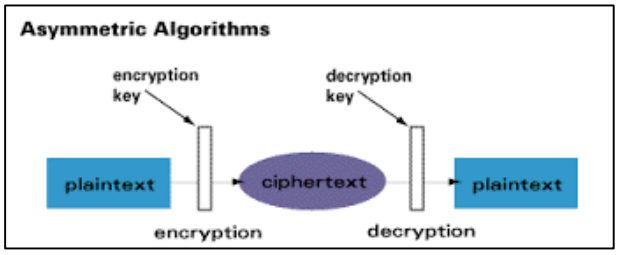
Secara umum, berbagai *cipher* yang ada dapat dibagi menjadi dua, yaitu:

1. Simetrik. *Cipher* jenis ini menggunakan kunci yang sama untuk proses enkripsi dan dekripsinya. Contoh *cipher* jenis ini adalah AES, RC4, dan DES. Caesar cipher dan berbagai *cipher* klasik lainnya juga merupakan *cipher* simetrik.



Gambar 1 Algoritma Simetrik [1]

2. Asimetrik. *Cipher* jenis ini menggunakan sepasang kunci yang berbeda untuk proses enkripsi dan dekripsinya. Kunci untuk proses enkripsi biasa disebut *public key* dan bersifat tidak rahasia. Sedangkan kunci untuk proses dekripsi biasa disebut *private key* dan bersifat rahasia. Contoh *cipher* jenis ini adalah RSA beserta variannya serta Elgamal.



Gambar 2 Algoritma Asimetrik [1]

B. RSA (Rivest–Shamir–Adleman)

RSA diciptakan oleh Ron Rivest, Adi Shamir, dan Leonard Adleman pada tahun 1977. RSA termasuk ke dalam *cipher* asimetrik yang paling dikenal dan banyak digunakan saat ini. Implementasi RSA didasarkan pada aritmatika modulo dan fakta bahwa proses pemfaktoran bilangan yang sangat besar merupakan proses yang susah dan membutuhkan waktu yang sangat lama. Sebaliknya, proses pencarian *private key* dan *public key* dengan diberikan dua bilangan prima (atau dalam varian RSA lainnya lebih dari dua) adalah mudah.

Terdapat beberapa istilah dan notasi yang umum digunakan dalam RSA, yaitu:

1. Modulus, atau sering dilambangkan dengan N, adalah suatu bilangan yang digunakan sebagai modulus dalam perhitungan di RSA. Modulus N merupakan bagian dari *public* dan *private key*. Ukuran N biasanya dinyatakan dalam jumlah bitnya.
2. Bilangan prima, dalam variasi RSA original terdapat dua bilangan prima p dan q sehingga berlaku hubungan $N = pq$.
3. *Public exponent*, atau sering dilambangkan dengan e, adalah suatu bilangan yang digunakan untuk melakukan proses enkripsi *plaintext*. *Public exponent* e merupakan bagian dari *public key*.
4. *Private exponent,* atau sering dilambangkan dengan d, adalah suatu bilangan yang digunakan untuk melakukan proses dekripsi *ciphertext. Private exponent* d merupakan bagian dari *private key.*

Algoritma untuk melakukan enkripsi dan dekripsi RSA bisa dibilang sederhana, prosesnya adalah sebagai berikut:

1. Enkripsi.
   Proses enkripsi memanfaatkan nilai e dan N untuk mengubah M menjadi C sesuai persamaan:
   $$C = M^e \bmod N \quad \ldots (1)$$
   Dalam proses enkripsi ini nilai M haruslah bernilai kurang dari N dan tidak negatif.
2. Dekripsi.
   Proses dekripsi memanfaatkan nilai d dan N untuk mengembalikan C menjadi M sesuai persamaan:
   $$M = C^d \bmod N \quad \ldots (2)$$

Bagian yang cukup kompleks dalam RSA adalah proses pembangkitan kunci yang diperlukan. Proses ini secara umum tersusun atas beberapa langkah, yaitu:

1. Pilih dua bilangan prima p dan q.
   Tahap pemilihan bilangan ini harus diperhatikan mengingat pemilihan bilangan prima yang kurang kuat bisa menyebabkan celah. Misalkan, nilai p dan q yang terlalu kecil sehingga proses pemfaktoran N menjadi mudah. Selain itu, bilangan prima yang nilainya berdekatan juga menyebabkan celah dan memungkinkan proses pemfaktoran Fermat bisa dilaksanakan secara efisien.
2. Hitung nilai N.
   Gunakan persamaan:
   $$N = pq \quad \ldots (3)$$
3. Hitung nilai fungsi euler totient dari N.
   Fungsi euler totien, biasa dilambangkan dengan $\phi(N)$, didefinisikan sebagai jumlah bilangan integer positif k yang relatif prima terhadap N, dengan $1 \leq k \leq N$.
   Jika p dan q relatif prima, maka berlaku hubungan:
   $$\phi(pq) = \phi(p)\phi(q) \quad \ldots (4)$$
   Kemudian, jelas bahwa untuk suatu bilangan p prima, berlaku:
   $$\phi(p) = p - 1 \quad \ldots (5)$$
   Oleh karena itu, dari persamaan (3), (4), dan (5) bisa kita dapatkan persamaan untuk mencari nilai fungsi euler totient dari N, yaitu:
   $$\phi(N) = (p-1)(q-1) \quad \ldots (6)$$

4. Pilih nilai e.
   Nilai e harus memenuhi dua syarat berikut:
   a. $1 < e < \phi(N)$
   b. $\gcd(e, \phi(N)) = 1$
5. Hitung nilai d.
   Nilai d dapat dihitung menggunakan persamaan berikut:
   $$d \equiv e^{-1} (mod\ \phi(N)) \quad \dots (7)$$
   Solusi dari persamaan (7) bisa dicari secara efisien menggunakan Extended Euclidean Algorithm.
6. *Private* dan *public key*.
   Private key tersusun atas (d, N) dan *public key* tersusun atas (e, N)

C. Teorema Coppersmith

Teorema 1 (Coppersmith) *Untuk N sebuah integer dan* $f \in \mathbb{Z}[x]$ *adalah suatu polinom monik dengan derajat d. Kita tetapkan* $X = N^{\frac{1}{d}-\epsilon}$ *untuk suatu* $\epsilon \geq 0$. *Kemudian, diberikan pasangan* $\langle N, f \rangle$, *terdapat suatu algoritma yang efektif untuk mendapatkan semua integer* $|x_0| < X$ *dengan* x_0 *memenuhi* $f(x_0) = 0 \bmod N$. *Waktu komputasi metode tersebut didominasi oleh waktu komputasi dari algoritma LLL dalam suatu lattice dimensi O(w) dengan* $w = min\left\{\frac{1}{\epsilon}, log_2 N\right\}$.

Sebagai catatan, polinom monik adalah suatu polinom dengan satu variabel dengan koefisien variabel derajat tertingginya bernilai 1. Secara umum, bentuknya adalah sebagai berikut:

$$x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_1 x + c_0 = 0$$

Kemudian, algoritma LLL (Lenstra–Lenstra–Lovász) adalah suatu algoritma yang dipakai untuk melakukan reduksi terhadap basis suatu *lattice*. Algoritma tersebut berjalan dalam waktu polinomial sehingga prosesnya cukup cepat untuk algoritma sejenis. *Lattice* sendiri secara sederhana adalah suatu ruang vektor yang merupakan hasil kombinasi linier basisnya terhadap suatu koefisien integer. Pembahasan mengenai algoritma LLL dan *lattice* dicukupkan sampai titik ini, untuk pembahasan yang lebih mendalam penulis serahkan pada sumber lain.

Proses pencarian akar suatu polinom dalam modulo N bukanlah proses yang mudah. Teorema tersebut menyatakan bahwa terdapat suatu algoritma yang efektif untuk mendapatkan semua akar dari suatu polinom monik berderajat d yang nilainya kecil. Definisi nilai kecil di sini adalah $|x_0| < N^{\frac{1}{d}-\epsilon}$ untuk suatu $\epsilon \geq 0$.

Metode pencarian akar dari Teorema Coppersmith tidak akan dibahas dalam makalah ini. Akan tetapi, makalah ini akan merujuk pada repositori Github milik David Wong, yaitu pada alamat *https://github.com/mimoo/RSA-and-LLL-attacks*. Dalam repositori tersebut, David Wong telah membuat implementasi metode Coppersmith dalam *script* sage python. Selain itu, David Wong juga membuat metode mirip milik Coppersmith yang diciptakan oleh Boneh-Durfee. Selain itu, metode ini juga telah diimplementasikan dalam SageMath yang dibuat berdasarkan bahasa python.

Makalah ini akan lebih fokus pada landasan matematis mengenai implementasi *broadcast* attack dalam RSA ini.

D. Teorema Hastad

Teorema 2 (Hastad) *Untuk* $N_1, N_2, N_3, \dots, N_k$ *suatu integer yang saling relatif prima. Kita tetapkan* $N_{min} = \min(N_1, N_2, N_3, \dots, N_k)$. *Terdapat* $g_i \in \mathbb{Z}_{N_i}[x]$ *adalah sebuah polinom dengan derajat d. Misalkan terdapat suatu nilai unik* $M < N_{min}$ *yang memenuhi*

$$g_i(M) = 0 \bmod N_i \quad \text{untuk semua } i = 1,2,3,\dots,k$$

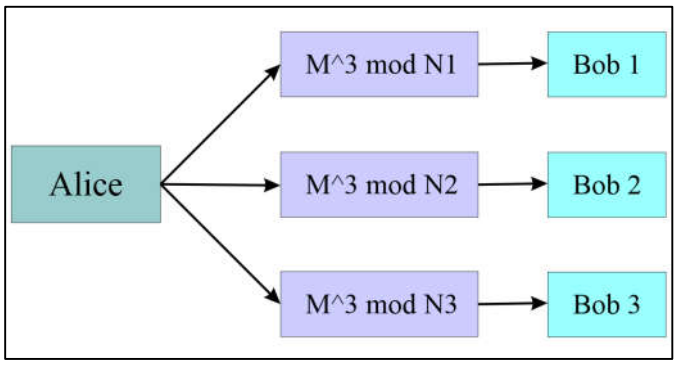
Dengan asumsi $k > d$, *kita dapat menemukan M secara efisien dengan diberikan* $\langle N_i, g_i \rangle_{i=1}^{k}$.

Teorema Hastad ini mendasarkan idenya pada metode pemfaktoran dari Coppersmith. Teorema ini akan berguna saat kita telah melakukan transformasi pada bentuk polinom yang merepresentasikan enkripsi RSA.

III. Broadcast Attack

A. Broadcast RSA versi 1

Misalkan seorang bernama Alice ingin mengirimkan suatu pesan M secara masal kepada keluarga Bob. Proses pengiriman pesan itu dia enkripsi menggunakan algoritma RSA karena telah yakin akan keamanannya. Saat pertama kali mencoba, Alice menggunakan cara yang naif, yaitu dengan mengenkripsi pesan tersebut menggunakan berbagai modulus dan *public exponent* yang telah tersedia. Kesalahan yang cukup fatal adalah Alice menggunakan *public exponent* yang terlalu kecil, dalam hal ini $e = 3$.



Gambar 3 *Broadcast* RSA versi 1

Di sisi lain terdapat Eve yang ingin meretas informasi yang Alice kirim. Melalui berbagai cara, Eve berhasil mendapatkan ketiga pasang $\langle C, N \rangle$ yang dikirimkan oleh Alice. Eve pun mengetahui bahwa Alice tidak memberikan tambahan apapun sebagai *padding* pada *plaintext* yang ia enkripsi.

Oleh karena itu, mudah saja bagi Eve untuk mendapatkan *plaintext* awal si Alice. Dari persamaan (1) Eve mendapatkan 3 set persamaan, yaitu:

$$C_1 \equiv M^3 \ (mod\ N_1)$$
$$C_2 \equiv M^3 \ (mod\ N_2)$$
$$C_3 \equiv M^3 \ (mod\ N_3)$$

Kemudian, menggunakan Chinese Remainder Theorem, Eve bisa menyusun persamaan berikut:

$$M^3 \equiv C_1\alpha\alpha^{-1} + C_2\beta\beta^{-1} + C_3\gamma\gamma^{-1} (mod\ N_1N_2N_3) \ldots(8)$$

dengan:

$$\alpha = N_2N_3$$
$$\alpha\alpha^{-1} \equiv 1\ (mod\ N_1)$$
$$\beta = N_1N_3$$
$$\beta\beta^{-1} \equiv 1\ (mod\ N_2)$$
$$\gamma = N_1N_2$$
$$\gamma\gamma^{-1} \equiv 1\ (mod\ N_3)$$

Karena $M < N_1$, $M < N_2$, dan $M < N_3$ sesuai syarat untuk melakukan enkripsi RSA, maka berlaku hubungan:

$$M^3 < N_1N_2N_3 \quad \ldots (9)$$

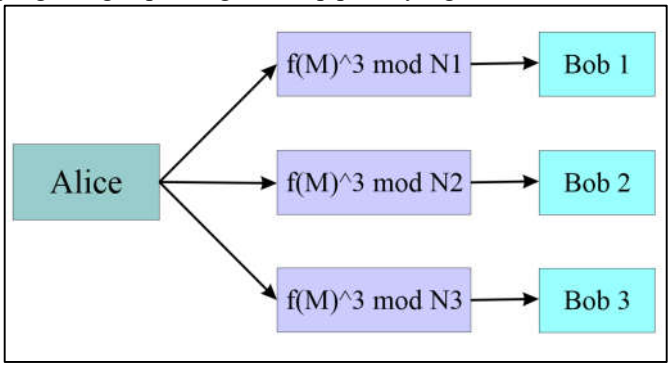
Menggunakan dasar persamaan (9), nilai M dari persamaan (8) dapat dihitung dengan mudah, yaitu dengan mencari akar pangkat 3 dari nilai M^3 yang berhasil dihitung sebelumnya.

B. Broadcast RSA versi 2

Mengetahui kegagalannya, dalam pengiriman pesan berikutnya, Alice memberikan *padding* linier kepada nilai M. Alice mendefinisikan fungsi untuk melakukan *padding* sebagai berikut:

$$f_i(M) = a_iM + b_i \quad \ldots (10)$$

Fungsi tersebut dibuat publik agar Bob bisa melakukan proses penghilangan padding terhadap pesan yang Alice kirimkan.



Gambar 4 *Broadcast* RSA versi 2

Sekilas, metode pengiriman di atas jauh lebih baik dan aman dibandingkan dengan versi 1 sebelumnya. Akan tetapi, metode ini jauh dari kata aman. Sesuai teorema 2 (Hastad), Eve bisa mendapatkan *plaintext* yang dikirim oleh Alice. Bahkan, untuk pesan yang dikirim menggunakan nilai e yang berbeda-beda, *padding* linier tidak akan membantu mengamankan pesan awal.

Syarat agar serangan ini bisa berhasil adalah Eve mendapatkan pasangan $\langle C, N, f \rangle$ yang cukup. Lebih tepatnya, Eve minimal membutuhkan sejumlah k pasang, dengan k adalah $\max(e_i . \deg(f_i))$.

Asumsikan bahwa semua persyaratan telah dipenuhi oleh Eve. Eve bisa memandang persamaan (1) menjadi suatu polinom $g(x)$, sesuai definisi berikut:

$$g_i(x) = (x^{e_i} - C_i)\ mod\ N_i \quad \ldots (11)$$

Akan tetapi, dalam enkripsi yang dilakukan Alice bukan x yang dipangkatkan dengan e, tetapi $f(x)$, dengan f adalah fungsi *padding* yang dipakai Alice. Sehingga persamaan (11) dimodifikasi menjadi:

$$g_i(x) = (f_i(x)^{e_i} - C_i)\ mod\ N_i \quad \ldots (12)$$

sehingga didapat bentuk:

$$g_i(M) = 0\ mod\ N_i, \quad M < N_i \quad \ldots (13)$$

Sesuai Teorema Hastad, Eve bisa mendapatkan nilai M secara efisien.

Eve bisa memulai serangan dengan membuat polinom $p(x)$ sebagai berikut:

$$p(x) = \sum_{i=1}^{k} T_i(f_i(x)^{e_i} - C_i) \quad \ldots (14)$$

dengan:

$$T_i \equiv \begin{cases} 1\ (mod\ N_j), & i = j \\ 0\ (mod\ N_j), & i \neq j \end{cases}$$

dengan $1 \leq j \leq k$. Nilai T_i bisa dicari menggunakan Chinese Remainder Theorem sehingga terbentuk kongruensi dalam modulo $N_{prod} = N_1, N_2, N_3, \ldots, N_k$. Selain itu, f_i adalah fungsi *padding* yang dipakai Alice.

Polinom $p(x)$ memiliki properti yang menarik, yaitu:

$$p(M) \equiv 0\ (mod\ N_i) \quad \text{untuk } 1 \leq i \leq k \ldots (15)$$

Karena pasti salah satu komponen penyusun suku sigma untuk $p(x)$ bernilai 0 untuk setiap i dalam rentang yang ditentukan.

Misalkan saat $i = 1$ dan j adalah indeks suku sigma penyusun $p(x)$ di persamaan (14). Saat $i \neq j$, berlaku hubungan $T_j \equiv 0\ (mod\ N_i)$ dari definisi nilai T_i. Selain itu, saat $i = j$, nilai $g_i(M) \equiv f_i(x)^e - C_i \equiv 0\ (mod\ N_i)$ dari persamaan (13). Hal ini juga berlaku untuk nilai i lain dalam rentang $1 \leq i \leq k$.

Dari persamaan (15), dapat disimpulkan bahwa:

$$p(M) \equiv 0\ (mod\ N_{prod}) \quad \ldots (16)$$

atau dengan kata lain nilai M merupakan akar dari polinom $p(x)$ dalam modulo N_{prod}.

Ingat bahwa nilai $M < N_i$ untuk setiap i dengan $1 \leq i \leq k$ sehingga berlaku:

$$M^k < N_{prod}$$

atau dengan kata lain:

$$M < N_{prod}^{1/k}$$

Akibatnya proses pencarian akar dari persamaan (16) ini memenuhi Teorema 1 (Coppersmith) dan dapat disimpulkan bahwa terdapat suatu metode yang efisien untuk mendapatkan nilai M.

C.Intisari Program Serangan Broadcast RSA versi 2

Misalkan Alice ingin mengirimkan pesan M kepada keluarga Bob. Alice mendefinisikan fungsi *padding* yang dia pakai adalah:

$$f_i(x) = 2 * i * x + i \quad \ldots (17)$$

Fungsi tersebut bersifat publik.

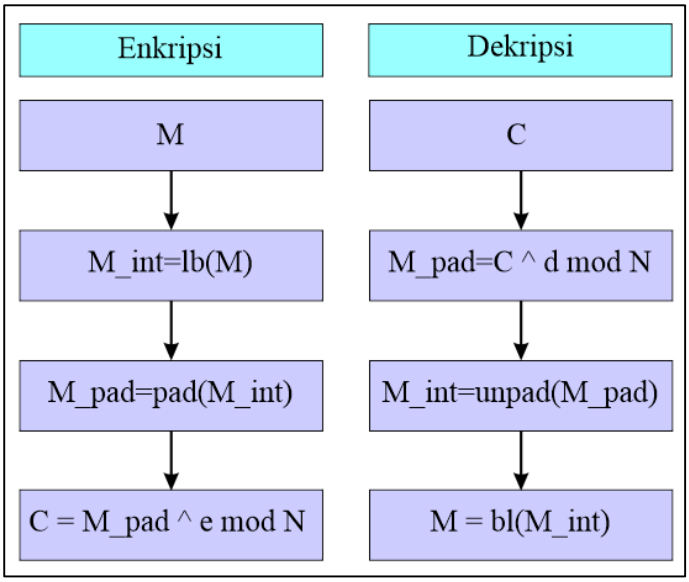
Kemudian dia pun memilih nilai $e = 5$. Selain itu, dia sudah membuat kode untuk melakukan enkripsi menggunakan bahasa python. Berikut ini adalah gambaran kode yang dibuat oleh Alice, tanpa modulus N dan pesan yang ia kirimkan.

```
from Crypto.Util.number import long_to_bytes as
lb, bytes_to_long as bl

def pad(x, i):
    return ((2 * i * x) + i)
```

```
def unpad(x, i):
    return ((x - i) / (2 * i))

if __name__ == '__main__':
    pt = b"MESSAGE"
    pt = bl(pt)

    e = 5
    NList = [...]

    ptList = [pad(pt, i + 1) for i in
              range(5)]
    ctList = [pow(ptList[i], e, NList[i]) for
              i in range(5)]
```

Dengan *script* yang ia buat, ia berhasil mendapatkan 5 buah *ciphertext* hasil enkripsi pesan dalam variabel pt menggunakan nilai $e = 5$ dan 5 buah modulus yang berbeda milik keluarga Bob sehingga . *Ciphertext* tersebut terdapat dalam list ctList dan modulus keluarga Bob yang bersesuaian berada di NList. Alice pun mengirimkan *ciphertext* tersebut ke pemilik modulus yang bersesuaian.

Selanjutnya, tiap anggota keluarga Bob bisa menghitung nilai *private exponent* d, mengingat mereka mempunyai bilangan prima p dan q pembentuk N. Oleh karena itu, tiap anggota keluarga Bob bisa melakukan proses dekripsi pada pesan yang telah dikirim Alice.



Gambar 5 Proses enkripsi dan dekripsi pesan

Di sisi lain, terdapat Eve, yang saat ini mempunyai informasi mengenai pasangan nilai $\langle C, N, f \rangle$ sejumlah 5 pasang. Jumlah ini cukup untuk melakukan serangan mengingat $\max(e_i . \deg(f_i)) = 5$.

Dengan informasi yang ia miliki, Eve pun membuat polinom $p(x)$ sesuai persamaan (14). Kemudian untuk mencari akar polinom $p(x)$, Eve menggunakan metode Coppersmith yang telah diimplementasikan dalam SageMath, yaitu pada metode small_root(). Intisari *script* sage yang dibuat oleh Eve adalah sebagai berikut:

```
# Fungsi pad
def pad(x, i):
    return ((2 * i * x) + i)

# List modulus N dan ciphertext C
NList = [...]
ctList = [...]

# Buat polinomial ring dalam modulo Nprod
Nprod = 1
for i in range(5):
    Nprod *= NList[i]
P.<x> = PolynomialRing(Zmod(Nprod))

# List Chinese Remainder Coefficient
crtHelperList = [[1,0,0,0,0], [0,1,0,0,0],
                 [0,0,1,0,0], [0,0,0,1,0],
                 [0,0,0,0,1]]
TList = [crt(crtHelperList[i], NList) for i in
         range(5)]


# Polinom p(x), misal namanya pol
pol = 0
for i in range(5):
    pol += TList[i] * ((pad(x, i + 1) ** 5) -
                                    ctList[i])
pol = pol.monic()
roots = pol.small_roots()
```

Secara umum, *script* di atas melakukan beberapa hal. Pertama, pendefinisian fungsi *padding* yang dibuat oleh Alice pada persamaan (17). Kedua, membuat list modulus N dan *ciphertext* C yang telah didapatkan. Ketiga, Membuat polinomial ring yang merupakan fitur pada SageMath. Polinomial yang dibuat berada dalam modulo $N_{prod} = N_1 N_2 N_3 N_4 N_5$. Keempat, membuat list Chinese Remainder Coefficient. Sesuai definisi pada persamaan (14). Dalam proses ini memanfaatkan fungsi *crt(list_a, list_n)* pada SageMath untuk menghitung nilai Chinese Remainder Theorem. Kelima, melakukan konstruksi $p(x)$ dalam variabel *pol* sesuai persamaan (14). Kemudian, polinom *pol* harus dibuat menjadi polinom monik sebagai syarat metode Coppersmith. Proses ini dilaksanakan menggunakan metode *monic()*. Terakhir, pencarian akar menggunakan implementasi metode Coppersmith dalam metode *small_roots()*.

IV. KESIMPULAN

RSA merupakan *cipher* yang mutakhir dan sangat banyak digunakan saat ini. RSA memiliki ketahanan yang tinggi terhadap serangan. Akan tetapi, penerapan RSA yang kurang tepat bisa memunculkan celah keamanan. Salah satunya adalah pada proses *broadcast* pesan identik menggunakan nilai modulus yang berbeda. Menggunakan teorema Coppersmith, Hastad berhasil menciptakan cara untuk mengekstrak *plaintext* yang dikirimkan jika diberikan jumlah pasangan *cipher text* dan modulus yang cukup. Sebagai tambahan, padding linear tidak bisa dipakai untuk menangkal celah keamanan tersebut. Terdapat beberapa cara untuk mengatasi permasalahan ini.

Pertama, penggunaan padding yang dibangkitkan secara acak. Kedua, hindari penggunaan nilai e yang terlalu kecil mengingat jumlah pesan yang dibutuhkan sebanding dengan nilai e yang digunakan. Secara umum, nilai $e = 65537$ adalah suatu standar aman yang banyak digunakan saat ini.

V. Ucapan Terima Kasih

Penulis ingin mengucapkan terima kasih kepada berbagai pihak yang telah membantu proses penyelesaian makalah ini. Pertama, kepada Ibu Fariska Zakhralativa Ruskanda, S.T., M.T. selaku dosen mata kuliah IF2120 Matematika Diskrit K-3 yang selalu membimbing kami dalam memahami materi yang dipakai dalam makalah ini. Kedua, kepada penulis sumber referensi makalah ini yang namanya dicantumkan dalam bagian referensi. Ketiga, orang tua penulis yang senantiasa memberi motivasi dan semangat untuk terus belajar. Keempat, teman-teman jurusan Teknik Informatika ITB, khususnya angkatan 2018, yang memberikan semangat untuk bersaing secara sehat selama kuliah di ITB. Penulis berharap semoga kebaikan dari berbagai pihak tersebut di atas akan diberi balasan baik oleh Tuhan Yang Maha Esa.

Referensi

[1] Rani, Sonia and Harpreet Kaur. “Technical Review on Symmetric and Asymmetric Cryptography Algorithms” in *International Journal of Advanced Research in Computer Science*, Volume 8, No. 4. India: IJARCS, 2017.

[2] Boneh, Dan. *Twenty Years of Attacks on the RSA Cryptosystem.* California: Stanford University.

[3] Peikert, Chris and Jacob Alperin-Sheriff. *Coppersmith, Cryptanalysis*. Georgia: Georgia Tech, 2013.

[4] Van Houtven, Laurens. *Crypto101.* Creative Commons Attribution-NonCommercial 4.0 International, 2017.

[5] Lenstra, A. K., H. W. Lenstra Jr., and L Lovász. "Factoring polynomials with rational coefficients". Mathematische Annalen. 261 (4): 515–534. 1982.

[6] Munir, Rinaldi. “Teori Bilangan”. Bandung: Institut Teknologi Bandung, 2015.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 3 Desember 2019

Naufal Dean Anugrah 13518123