Kriptografi untuk Transmisi Data

Vincent Chuardi / 13517103
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517103@std.itb.ac.id

***Abstract*—Teknologi komunikasi jarak jauh pada zaman milenial sudah berkembang pesat dan mudah dijangkau oleh masyarakat umum. Teknologi komunikasi sudah mulai meresap ke dalam kehidupan masyarakat umum. Terlebih pada tahun pandemi ini, tahun 2020, mayoritas pekerjaan dilakukan dengan WFH (*Work From Home*), termasuk kegiatan belajar-mengajar. Semua kegiatan ini sangat bergantung kepada koneksi jaringan internet. Hal ini menyebabkan keamanan data yang bersifat sensitif harus lebih dijaga dan diperhatikan. Kemanan data dijaga dengan menerapkan kripto pada data.**

***Keywords*—Komunikasi, internet, kriptografi**

I. PENDAHULUAN

Internet sudah digunakan secara umum oleh masyarakat dalam kehidupan sehari-hari. Kegiatan seperti kegiatan belajar-mengajar, kegiatan pembelian makanan/minuman, kegiatan pembayaran transaksi, bahkan sampai pekerjaan kantoran dilakukan secara WFH melalui jaringan internet. Mayoritas kegiatan tersebut dilakukan dengan menggunakan aplikasi berbasis website.

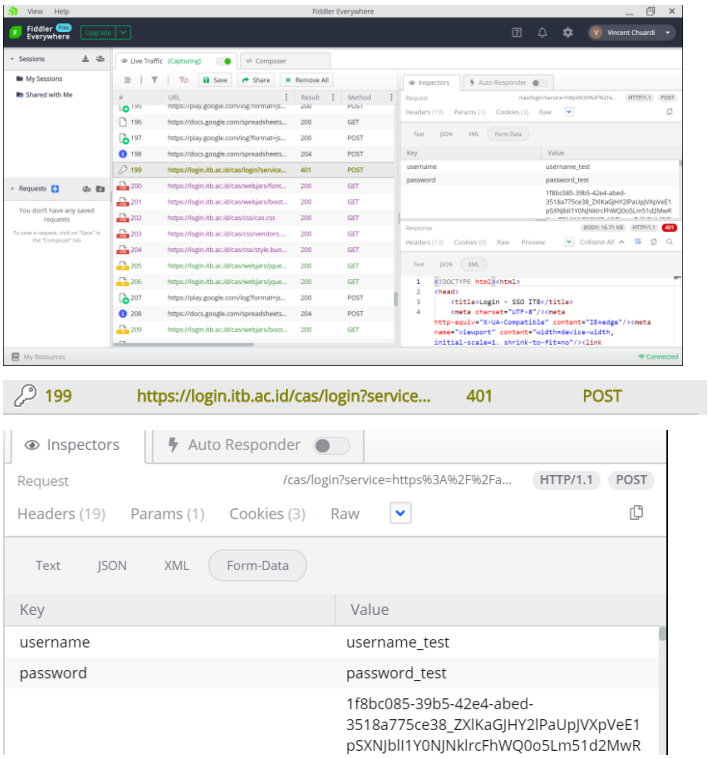
Berlandaskan pada paragraf diatas, kemunculan kegiatan tidak etik berupa kejahatan siber semakin banyak. Untuk menjaga data agar tetap aman, perlu dilakukan kripto pada data yang perlu dijaga kemanannya.

Kriptografi secara singkat adalah ilmu yang mempelajari cara mengamankan data dari pihak-pihak yang tidak berwenang. Terdapat dua cara umum pengamanan data, yaitu dengan melakukan sandian pada data dan melakukan stegano pada data. Penyandian adalah kegiatan pengubahan data awal menjadi bentuk yang tidak dapat dibaca dengan mudah oleh pihak-pihak yang tidak berwenang. Stegano adalah kegiatan mengamankan pesan dengan melakukan penyisipan pesan pada media yang umum digunakan dan memiliki arti.

Pada makalah ini, kriptografi yang digunakan adalah kriptografi penyandian. Secara spesifik, kriptografi penyandian yang digunakan adalah kriptografi berbasis kunci publik.

II. RUMUSAN MASALAH

Permasalahan pada makalah ini adalah algoritma penyandian yang cocok digunakan dalam transmisi data di dalam sebuah aplikasi berbasis website. Transmisi data yang perlu dienkripsi adalah pada saat data dari suatu gawai ditransmisi ke router/modem. Transmisi data dari router/modem sampai server sudah aman jika protokol yang digunakan berupa https.



Gambar 1. Hasil tangkapan Fiddler

Pada Gambar 1, terlihat bahwa data form login terlihat tanpa ada pengamanan apapun. Fiddler adalah aplikasi yang dapat memonitor *traffic* koneksi yang dilakukan sebuah gawai yang di monitor saja. Aplikasi Fiddler bukan merupakan aplikasi yang dapat digunakan untuk melakukan *sniffing* jaringan dan kejahatan siber lainnya.

Masalah yang dibahas pada makalah ini adalah algoritma penyandian yang cocok untuk melakukan pengamanan data dari gawai menuju jaringan internet. Pengamanan yang dilakukan dalam permasalahan ini akan meningkatkan keamanan data. Walaupun gawai yang dipakai memiliki *malware* yang dapat mengawasi *traffic* jaringan, data sensitif tetap terjaga.

III. DASAR TEORI

Berikut dijelaskan beberapa algoritma kriptografi kunci publik yang akan menjadi kandidat algoritma yang digunakan

A. RSA

RSA (Rivest-Shamir-Adleman) adalah algoritma kripto kunci publik yang paling terkenal dan paling banyak diaplikasikan. Algoritma RSA ditemukan oleh tiga peneliti dari MIT

(Massachussets Institute of Technology), yaitu Ronal Rivest, Adi Shamir, dan Len Adleman pada tahun 1976. Algoritma RSA adalah algoritma kunci publik yang memanfaatkan karakteristik dari pemfaktoran prima dari bilangan-bilangan bulat yang besar. Algoritma RSA berprinsipkan dari Teorema Euler. Rumus enkripsi dan dekripsi algoritma RSA terdapat pada persamaan (1) dan (2) secara berurutan.

$$\text{Enkripsi} \quad : E_e(m) = c = m^e \ mod \ n \quad (1)$$

$$\text{Dekripsi} \quad : D_d(c) = m = c^d \ mod \ n \quad (2)$$

Prosedur pembangkitan kunci yang dipakai RSA terdiri dari lima tahap, yaitu:

1. Pilih dua bilangan prima p dan q.
2. Hitung persamaan (3)

$$n = p * q \qquad (3)$$

3. Hitung persamaan (4)

$$\phi(\mathrm{n}) = (\mathrm{p} - 1) * (\mathrm{q} - 1) \qquad (4)$$

4. Pilih bilangan relatif prima e terhadap ϕ(n) sebagai kunci enkripsi publik.
5. Hitung kunci dekripsi (d) melalui persamaan (5)

$$d = e^{-1} \ mod \ (\phi(\mathrm{n})) \qquad (5)$$

Kunci publik yang diperlukan untuk melakukan enkripsi pesan berupa pasangan (e, n). Kunci privat yang digunakan untuk melakukan dekripsi *ciphertext* berupa pasangan (d, n).

Cara melakukan enkripsi adalah dengan mempartisi pesan menjadi blok-blok kecil, dengan syarat nilai setiap blok bernilai $0 \leq \text{ukuran\_block} < (\mathrm{n}-1)$

.Blok-blok kecil tersebut akan dimasukan ke dalam persamaan enkripsi (1), kemudian setiap hasil enkripsi dikonkatenasi dan dikirimkan.

Cara melakukan dekripsi adalah dengan mempartisi *ciphertext* sesuai yang dilakukan enkripsi pada tahap sebelumnya. Setiap blok *ciphertext* akan dimasukan ke dalam rumus dekripsi (2) dan dikonkatenasi hasil dekripsi. Hasil konkatenasi adalah pesan awal.

Tingkat keamanan algoritma RSA bergantung terhadap nilai n. Menurut referensi [1], "Usaha untuk mencari faktor bilangan 200 digit membutuhkan waktu komputasi selama 4 milyar tahun, sedangkan untuk bilangan 500 digit membutuhkan waktu 10^{25} tahun".

Algoritma RSA memiliki kelemahan berupa waktu eksekusi yang cenderung lambat.

B. Elgamal

Algoritma Elgamal adalah algoritma kunci publik yang didasari pada logaritma diskrit. Algoritma Elgamal dibuat oleh Taher Elgamal pada tahun 1985. Algoritma Elgamal mengandalkan kesukaran menghitung logaritma diskrit.

Tahap pembangkitan kunci yang diperlukan Elgamal adalah:

1. Pilih sembarang bilangan prima *p*
2. Pilih dua buah bilangan acak, g dan x, dengan syarat $g < p$ dan $1 \leq \mathrm{x} \leq \mathrm{p} - 2$
3. Hitung persamaan (6)

$$y = g^x mod \ p \qquad (6)$$

Kunci publik adalah pasangan tripel (y, g, p), sedangkan kunci privat adalah pasangan (x, p).

Prosedur enkripsi pesan menggunakan algoritma Elgamal adalah

1. Partisi pesan menjadi blok kecil *m* dalam selang [0, p-1]
2. Pilih bilangan acak *k* dalam selang [1, p-2]
3. Setiap blok *m* dienkripsi dengan rumus (7) dan (8)

$$a = g^k \ mod \ p \qquad (7)$$

$$b = y^k \ m \ mod \ p \qquad (8)$$

4. Hasil enkripsi akan dikonkatenasi menjadi *ciphertext* utuh.

Prosedur dekripsi *ciphertext* hasil enkripsi algoritma Elgamal adalah

1. Gunakan kunci privat x untuk menghitung persamaan (9)

$$(a^x)^{-1} = \ a^{p-1-x} \ mod \ p \qquad (9)$$

2. Hitung plainteks *m* dengan persamaan (10)

$$m = \frac{b}{a^x} mod \ p = b(a^x)^{-1} \ mod \ p \qquad (10)$$

Keamanan algoritma Elgamal terdapat pada bilangan prima *p* yang digunakan sebagai modulus. Semakin besar nilai *p*, semakin aman hasil enkripsi nya.

Kelebihan algoritma Elgamal adalah *ciphertext* yang dihasilkan proses enkripsi berukuran dua kali pesan awal. Sehingga waktu yang dibutuhkan untuk mendekripsikan *ciphertext* tanpa kunci privat dua kali lebih banyak dibanding algoritma lainnya.

Kelemahan algoritma Elgamal adalah membutuhkan *resource* yang besar karena *ciphertext* yang dihasilkan dua kali panjang pesan awal. Selain itu, memerlukan mesin komputasi yang besar untuk dapat melakukan perhitungan logaritma perpangkatan besar.

C. NaCl (Networking and Cryptography library)

NaCl adalah library kriptografi yang bisa melakukan enkripsi, dekripsi, dan digital signature. NaCl dibuat oleh matematikawan and programmer Daniel J. Bernstain, beserta tim utamanya yang terdiri dari Tanja Lange dan Peter Schwabe. Library NaCl dibuat dan bekerja dengan stabil pada tahun 2011.

Fungsi enkripsi *box* yang digunakan merupakan fungsi *high-level* yang terdiri dari

1. Konversi pesan menjadi bentuk *array* Uint8.
2. Mencari *shared key*.
3. Men-*generate long stream.*
4. Menggunakan *long stream* untuk enkripsi pesan.

Fungsi dekripsi *box* yang digunakan merupakan fungsi *high-level* yang terdiri dari

1. Mencari *shared key*.
2. Men-*generate long stream*.
3. Menggunakan *long stream* untuk membalikkan *ciphertext*.
4. Konversi hasil dekripsi yang berupa array Uint8 menjadi UTF-8.

Kunci privat dan publik adalah kumpulan karakter dengan panjang 32 karakter. Sebuah kunci berukuran sebesar 64 byte. Kunci privat *a* dapat dipilih secara random, dan kunci publik *A* didapat melalui fungsi kurva eliptik Curve25519 dengan masukan kunci privat yang diproses oleh fungsi ClampC dan angka sembilan (9). Cara mendapatkan kunci publik dapat dilihat pada persamaan (11).

$$A = Curve25519(ClampC(a), 9) \qquad (11)$$

Fungsi Curve25519 menggunakan persamaan (12) sebagai fungsi elips. Persamaan (12) merupakan sebuah kurva Montgomery, yang memiliki *prime field* berupa $2^{255} - 19$ dan *base point* x = 9. Dengan menggunakan konfigurasi demikian, serangan dari algoritma *Pohlig-Hellman* dapat dicegah. Protokol ini menggunakan *compressed elliptic point*, sehingga penggunaan *Montgomery ladder* untuk ECDH (*Elliptic Curve Diffie-Hellman*) dapat digunakan secara efisien dengan menggunakan koordinat XZ.

$$y^2 = x^3 + 486662x^3 + x \qquad (12)$$

Fungsi ClampC adalah fungsi yang bertujuan untuk melakukan mapping karakter masukan $(a_0, a_1, \dots, a_{30}, a_{31})$ menjadi $(a_0 - (a_0 \bmod 8), a_1, \dots, a_{30}, 64 + (a_{31} \bmod 64))$. Fungsi ClampC menghilangkan bit (7,0,…,0,0,128) dan menetapkan bit (0,0,…,0,0,64).

Enkripsi yang dilakukan oleh NaCl adalah berdasarkan prinsip *Diffie-Hellman* sehingga membutuhkan kunci publik dari penerima dan dekripsi yang dilakukan membutuhkan kunci publik dari pengirim. Kunci *sharing* dinamakan kunci *k*. Kunci *sharing k* ini akan dikonversikan menggunakan HSalsa20(*k*,0). Kunci hasil konversi HSalsa20 tersebut yang akan digunakan untuk melakukan proses enkripsi, dekripsi, dan validasi (*digital sign*).

Nonce adalah sekumpulan nilai yang maksimal berjumlah 24 buah. Nonce adalah angka unik pada pesan yang tidak boleh dipakai ulang pada saat pertukaran paket oleh pihak bersangkutan. Nonce *n* dapat menggunakan kunci *shared k* untuk mengembangkan *n* ke dalam *long stream*. Kunci tingkat pertama HSalsa20(*k*, 0) menggunakan 16 byte (n_1) dari nonce *n* untuk mengkomputasi kunci tingkat kedua HSalsa20(HSalsa20(*k*,0),n_1). Sisa 8 byte (n_2) dari nonce *n* digunakan untuk mengkomputasi *long stream* Salsa20(HSalsa20(HSalsa20(*k*,0),n_1), n_2). Stream ini yang akan digunakan untuk enkripsi dan autentikasi paket pesan.

Word adalah elemen dari selang $[0, 2^{32} - 1]$. Penjumlahan yang dilakukan antar *word* adalah $u + v = u + v \bmod 2^{32}$. XOR yang dilakukan antar *word* adalah penjumlahan *u* dan *v* dengan *carries* ditahan. *C-bit left rotation* dari sebuah *word u* (dilambangkan dengan $u<<<c$) adalah *word* unik non-nol yang kongruen kepada $2^c u \bmod (2^{32} - 1)$, kecuali jika 0<<<c=0.

Fungsi quaterround. Jika

$y = (y_0, y_1, y_2, y_3) \in \{0,1, \dots, 2^{32} - 1\}^4$ maka

$quarterround(y) = (z_0, z_1, z_2, z_3)$ dimana

$$z_1 = y_1 \oplus ((y_0 + y_3) \lll 7),$$
$$z_2 = y_2 \oplus ((z_1 + y_0) \lll 9),$$
$$z_3 = y_3 \oplus ((z_2 + z_1) \lll 13),$$
$$z_0 = y_0 \oplus ((z_3 + z_2) \lll 18).$$

Fungsi rowround. Jika

$y = (y_0, y_1, y_2, y_3, \dots, y_{15}) \in \{0,1, \dots, 2^{32} - 1\}^{16}$ maka

$rowround(y) = (z_0, z_1, z_2, z_3, \dots, z_{15})$ dimana

$$(z_0, z_1, z_2, z_3) = quarterround(y_0, y_1, y_2, y_3),$$
$$(z_5, z_6, z_7, z_4) = quarterround(y_5, y_6, y_7, y_4),$$
$$(z_{10}, z_{11}, z_8, z_9) = quarterround(y_{10}, y_{11}, y_8, y_9),$$
$$(z_{15}, z_{12}, z_{13}, z_{14}) = quarterround(y_{15}, y_{12}, y_{13}, y_{14}),$$

Fungsi columnround. Jika

$x = (x_0, x_1, x_2, x_3, \dots, x_{15}) \in \{0,1, \dots, 2^{32} - 1\}^{16}$ maka

$columnround(x) = (y_0, y_1, y_2, y_3, \dots, y_{15})$ dimana

$$(y_0, y_4, y_8, y_{12}) = quarterround(x_0, x_4, x_8, x_{12}),$$
$$(y_5, y_9, y_{13}, y_1) = quarterround(x_5, x_9, x_{13}, x_1),$$
$$(y_{10}, y_{14}, y_2, y_6) = quarterround(x_{10}, x_{14}, x_2, x_6),$$
$$(y_{15}, y_3, y_7, y_{11}) = quarterround(x_{15}, x_3, x_7, x_{11}).$$

Fungsi doubleround. Jika $x \in \{0,1, \dots, 2^{32} - 1\}^{16}$ maka

$doubleround(x) = rowround(columnround(x))$

Fungsi littleendian. Fungsi ini dapat dibalik. Jika

$b = (b_0, b_1, b_2, \dots, b_{4k-1}) \in \{0,1,2,3, \dots,255\}^{4k}$ maka

$littleendian(b) = (b_0 + 2^8 b_1 + 2^{16} b_2 + 2^{24} b_3,\ b_4 + 2^8 b_5 + 2^{16} b_7, \dots)$.

Fungsi HSalsa20 adalah

$\text{HSalsa20}: \{0,1, \dots,255\}^{32} \times \{0,1, \dots,255\}^{16} \rightarrow \{0,1, \dots,255\}^{32}$.

Fungsi didefinisikan sebagai berikut:

- $k \in \{0,1, \dots,255\}^{32}$ dan $n \in \{0,1, \dots,255\}^{16}$, definisikan $(x_0, x_1, \dots, x_{15}) \in \{0,1, \dots, 2^{32} - 1\}^{16}$ sebagai berikut:
  - $(x_0, x_5, x_{10}, x_{15}) =$ (0x61707865, 0z3320646e, 0x79622d32, 0x6b206574) adalah konstan Salsa20.
  - $(x_1, x_2, x_3, x_4, x_{11}, x_{12}, x_{13}, x_{14}) = littleendian(k)$

dan

- $(x_6, x_7, x_8, x_9) = littleendian(n)$

- Definisikan
  $(z_0, z_1, \dots, z_{15}) = doubleround^{10}(x_0, x_1, \dots, x_{15})$. Lalu
  $HSalsa20(k, n) =$
  $littleendian^{-1}(z_0, z_5, z_{10}, z_{15}, z_6, z_7, z_8, z_9)$

Fungsi ekspansi Salsa20 adalah

$Salsa20: \{0,1, \dots, 255\}^{32} \times \{0,1, \dots, 255\}^{16} \rightarrow \{0,1, \dots, 255\}^{64}$.

Fungsi didefinisikan sebagai berikut:

- $k \in \{0,1, \dots, 255\}^{32}$ dan $n \in \{0,1, \dots, 255\}^{16}$, definisikan
  $(x_0, x_1, \dots, x_{15}) \in \{0,1, \dots, 2^{32} - 1\}^{16}$ sebagai berikut:
  - $(x_0, x_5, x_{10}, x_{15})$ adalah konstan Salsa20.
  - $(x_1, x_2, x_3, x_4, x_{11}, x_{12}, x_{13}, x_{14}) = littleendian(k)$ dan
  - $(x_6, x_7, x_8, x_9) = littleendian(n)$

- Definisikan
  $(z_0, z_1, \dots, z_{15}) = doubleround^{10}(x_0, x_1, \dots, x_{15})$. Lalu
  $Salsa20(k, n) = littleendian^{-1}(x_0 + z_0, x_1 +$
  $z_1, \dots, x_{15} + z_{15})$

Fungsi streaming Salsa20 adalah

$Salsa20: \{0,1, \dots, 255\}^{32} \times \{0,1, \dots, 255\}^{8} \rightarrow \{0,1, \dots, 255\}^{2^{70}}$.

Fungsi didefinisikan sebagai berikut:

$$Salsa20(k, n) = Salsa20\left(k, n, \underline{0}\right), Salsa20\left(k, n, \underline{1}\right), \dots$$

Dimana $\underline{b}$ adalah string 8-byte ($b \bmod 256, \left\lfloor \frac{b}{256} \right\rfloor \bmod 256, \dots$)

D. Rabin

Algoritma Rabin adalah algoritma kunci publik yang dibuat oleh Michael Rabin. Algoritma Rabin memanfaatkan karakteristik dari pemfaktoran prima dari bilangan-bilangan bulat yang besar.

Langkah pembangkitan kunci yang digunakan algoritma Rabin:

1. Pilih 2 angka prima *p* dan *q*, dimana *p* dan *q* memenuhi $p \neq q \rightarrow p \equiv q \equiv 3 \ (mod\ 4)$.
2. Hitung *n* dengan $n = p * q$.
3. Nilai *n* akan menjadi kunci publik dan nilai *p* dan *q* sebagai kunci privat.

Langkah melakukan enkripsi data dengan menggunakan algoritma Rabin:

1. Konversi keseluruhan pesan menjadi nilai ASCII.
2. Konversi nilai ASCII menjadi *binary* dan meng-*extend* nilai *binary* dengan dirinya.
3. Ubah *binary* kembali menjadi desimal *m*.
4. Enkripsi dengan persamaan (13)

$$C = m^2 \bmod n \qquad (13)$$

5. Kirim hasil persamaan (13) ke penerima.

Langkah melakukan enkripsi data dengan menggunakan algoritma Rabin:

1. Cari nilai *a* dan *b* dengan *Extended Euclidean GCD* yang memenuhi persamaan (14).
   Extended Euclidean GCD adalah varian ekstensi dari algoritma *Euclidean*. Algoritma *Euclidean* digunakan untuk mencari Faktor Persekutuan Terbesar (FPB), atau dalam bahasa Inggris disebut *Greatest Common Divisor* (GCD), dari dua buah angka. Karena *Extended Euclidean GCD* merupakan ekstensi dari algoritma *Euclidean*, *Extended Euclidean GCD* selain mencari FPB dari dua buah angka, *Extended Euclidean GCD* juga mencari nilai *x* dan *y* yang memenuhi persamaan (15). Persamaan (14) didapat dari persamaan (15) karena FPB dari dua buah bilangan prima adalah 1.

$$a.p + b.q = 1 \qquad (14)$$

$$ax + by = \text{GCD}\ (a, b) \quad (15)$$

2. Cari nilai r dan s dengan persamaan (16) dan (17).

$$r = C^{(p+1)/4} \bmod p \qquad (16)$$

$$s = C^{(q+1)/4} \bmod q \qquad (17)$$

3. Kalkulasi X dan Y dengan persamaan (18) dan (19).

$$X = (a.p.r + b.q.s) \bmod p \qquad (18)$$

$$Y = (a.p.r - b.q.s) \bmod q \qquad (19)$$

4. Empat akar $m_1 = X, m_2 = -X, m_3 = Y, m_4 = -Y$. Keempat akar tersebut ditranformasikan ke bentuk *binary* dan bagi menjadi ½.
5. Tentukan bagian kiri dan kanan yang sama. Simpan satu bagian *binary* dan konversikan ke dalam desimal *m*.
6. Cari karakter ASCII untuk nilai desimal *m*. Hasil karakter tersebut adalah pesan dari pengirim.

Kelebihan dari algoritma Rabin adalah semakin besar nilai kunci publik, semakin susah pencarian faktor yang membentuk nilai kunci publik tersebut. Selain itu, algoritma ini juga melakukan enkripsi langsung terhadap masukan, sehingga jika pesan diintersep dan diubah, pesan akan rusak secara keseluruhan.

Kelemahan dari algoritma Rabin adalah hasil dari fungsi dekripsi Rabin memiliki empat kemungkinan, sehingga membutuhkan perlakukan khusus untuk menentukan pesan yang benar. Selain itu, pesan yang mampu dienkripsi juga memiliki besaran yang terbatas. Jika ukuran data besar dan ukuran kunci publik juga besar, algoritma tidak bisa melakukan enkripsi karena melewati batas maksimal angka yang mampu di komputasi oleh komputer.

IV. EKSPERIMEN

Bagian ini berisi percobaan-percobaan algoritma.

A. RSA

Data percobaan algoritma RSA ini didapatkan dengan menggunakan library `node-rsa` yang dicoba pada aplikasi node js dan diinstall dengan menggunakan *command* `npm install node-rsa`.

Table I. Eksekusi Algoritma RSA

Panjang Kunci	Ukuran Data (KByte)	Waktu Enkripsi (ms)	Waktu Dekripsi (ms)
512 bit	0.1	0.479	0.974
	1	0.681	3.756
	10	6.446	47.332
	100	64.565	372.738
1024 bit	0.1	0.095	0.359
	1	0.314	1.988
	10	1.984	21.421
	100	18.144	181.944
2048 bit	0.1	0.131	1.289
	1	0.209	3.128
	10	1.171	29.485
	100	12.071	243.702

B. Elgamal

Data percobaan algoritma RSA ini didapatkan dengan menggunakan kode `mindofea` yang dicoba pada aplikasi node js.
Sumber:
https://gist.github.com/mindoftea/624f769b193215c534ac

Table II. Eksekusi Algoritma Elgamal

Ukuran Data (KByte)	Waktu Enkripsi (ms)	Waktu Dekripsi (ms)
0.1	0.807	0.538
1	5.291	1.816
10	19.064	50.270
100	167.775	1744.822

C. NaCl

Data percobaan algoritma NaCl ini didapatkan dengan menggunakan library `tweetnacl` yang dicoba pada aplikasi node js dan diinstall dengan menggunakan *command* `npm install tweetnacl`.

Table III. Eksekusi Algoritma NaCl

Ukuran Data (KByte)	Waktu Enkripsi (ms)	Waktu Dekripsi (ms)
0.1	0.132	0.165
1	0.305	0.378
10	4.666	4.628
100	8.926	10.444

D. Rabin

Data percobaan algoritma Rabin ini didapatkan dengan menggunakan kode program java. Panjang key yang digunakan berupa 256-bit. Untuk panjang key berupa 512-bit, kode tidak mampu untuk melakukan enkripsi terhadap data yang berukuran satu KB.
Sumber: https://www.geeksforgeeks.org/rabin-cryptosystem-with-implementation

Table IV. Eksekusi Algoritma Rabin

Ukuran Data (KByte)	Waktu Enkripsi (ms)	Waktu Dekripsi (ms)
0.1	0.274	0.357
1	0.120	0.253
10	0.408	0.235
100	1.154	0.2539

V. ANALISIS

Eksperimen yang dilakukan merupakan hasil pencarian algoritma-algoritma yang tersedia secara publik di internet. Berhubung permasalahan yang hendak diselesaikan berupa permasalahan pengamanan data pada pengiriman awal dari gawai ke router, maka algoritma yang dicoba dapat dijalankan pada javascript, kecuali untuk algoritma Rabin. Algoritma Rabin masih belum tersedia implementasi dengan bahasa *javascript* yang di-*publish* secara publik.

Hanya implementasi algoritma RSA yang memiliki tingkat kebebasan yang terbesar. Tingkat kebebasan yang dimaksudkan adalah kebebasan dalam menentukan panjang bit kunci yang digunakan, kebebasan dalam ukuran data yang bisa dienkripsi. Implementasi algoritma lainnya memiliki keterbatasan sendiri.

Pada Elgamal, keterbatasan pada implementasinya adalah panjang kunci yang tidak dapat berukuran lebih dari 64-bit. Hasil perkalian untuk mendapatkan *ciphertext* juga tidak boleh melebihi 64-bit.

Pada NaCl, keterbatasannya pada implementasinya terletak pada panjang kunci yang digunakan. Panjang kuncinya yang dapat digunakan sebesar 256-bit saja. Untuk ukurannya kunci lainnya tidak dapat digunakan.

Pada Rabin, keterbatasan pada implementasinya terletak pada batasan bahasanya. Pada bahasa java, nilai maksimal sebuah bilangan bulat yang bisa ditampung bergantung pada ukuran RAM gawai. Pada gawai yang dicoba, ukuran panjang kunci yang dapat digunakan sebesar 256-bit dengan ukuran data 100 KB masih dapat dihitung. Untuk kunci sebesar 512-bit, batas ukuran data yang dapat dihitung berjumlah 50 Byte saja.

A. Keamanan

Untuk pengamanan data yang memiliki ukuran kecil, algoritma-algoritma yang dicoba dapat digunakan semuanya. Tingkat keamanan masing-masing algoritma yang dicoba terdapat pada urutan sebagai berikut. Dengan urutan paling atas adalah yang paling aman:

1. NaCl
2. RSA
3. Elgamal
4. Rabin

NaCl terdapat pada urutan teratas keamanan yang paling aman karena NaCl menggunakan ECC untuk mendapatkan kunci yang digunakan. Selain itu, NaCl juga menggunakan banyak tahap pengamanan kunci yang digunakan untuk meng-*generate* kunci alir yang digunakan untuk melakukan enkripsi

terhadap pesan. Meskipun enkripsi pesan dilakukan dengan menggunakan algoritma kunci simetri, kunci yang digunakan untuk membuat kunci alir diproses dengan sangat intensif. Selain itu, dengan menggunakan protokol *Diffie-Hellman*, penyampaian kunci publik dapat dilakukan dengan lebih mudah.

Algoritma RSA merupakan algoritma teraman kedua karena panjang kunci yang digunakan dapat diatur secara bebas, sehingga penyerang tidak dapat dengan mudah menemukan panjang kunci yang diperlukan.

Algoritma Elgamal dan algoritma Rabin tidak begitu berbeda jauh tingkat keamanannya. Algoritma Elgamal berada pada urutan ketiga karena persoalan untuk penyelesaian permasalahan logaritma diskrit lebih susah dilakukan dibandingkan dengan persoalan pemfaktoran bilangan bulat. Selain itu, algoritma Elgamal juga melakukan enkripsi satu *plaintext* menjadi dua *ciphertext*, sehingga upaya penyerang untuk intersep maupun modif lebih susah.

B. Kecepatan

Persoalan kecepatan eksekusi algoritma untuk melakukan enkripsi dan dekripsi dapat diurutkan sebagai berikut:

1. Rabin
2. NaCl
3. Elgamal
4. RSA

Rabin adalah algoritma untuk melakukan enkripsi dan dekripsi tercepat karena implementasi algoritma Rabin langsung mengoperasi perhitungan secara langsung pada keseluruhan masukan. Sehingga hanya dilakukan sekali perhitungan operasi. Ukuran data tidak terlalu berpengaruh selama *resource* yang dibutuhkan bisa dipenuhi. Selain itu, perbedaan bahasa dan kompiler mungkin juga memiliki perbedaan kecepatan eksekusi.

Pada bahasa javascript, NaCl merupakan algoritma tercepat. Peningkatan ukuran data juga tidak menyebabkan peningkatan waktu, secara eksponensial, yang diperlukan untuk melakukan enkripsi.

Pada algoritma Elgamal dan RSA, terjadi peningkatan waktu yang pesat ketika ukuran data yang dilakukan untuk enkripsi dan dekripsi bertambah. Menurut konsep, seharusnya algoritma Elgamal lebih lambat dibandingkan dengan algoritma RSA, tetapi pada implementasi, algoritma Elgamal yang dipakai tidak melakukan partisi sehingga operasi yang dilakukan menjadi lebih sedikit.

VI. KESIMPULAN

NaCL adalah algoritma yang paling cocok digunakan untuk melakukan enkripsi data sensitif. Karena NaCl memiliki tingkat keamanan yang tertinggi dari algoritma yang dipakai dan memiliki waktu enkripsi dan dekripsi yang cenderung kecil.

VI. UCAPAN TERIMA KASIH

Saya ucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, MT. yang telah membimbing saya ilmu-ilmu kriptografi kunci publik. Berbekal pengetahuan yang saya dapatkan, saya berhasil menyusun makalah berjudul ‘Kriptografi untuk Transmisi Data’ ini. Saya juga mengucapkan terima kasih untuk penyedia referensi-referensi yang saya gunakan. Tanpa ilmu-ilmu tambahan yang didapatkan dari referensi yang saya gunakan, makalah ini tidak akan mencapai tahap ini.

REFERENSI

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Algoritma-RSA-2020.pdf
[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Algoritma-Elgamal-2020.pdf
[3] https://payahtidur.com/project/elgamal
[4] https://cr.yp.to/highspeed/naclcrypto-20090310.pdf
[5] https://en.wikipedia.org/wiki/Curve25519
[6] https://www.geeksforgeeks.org/rabin-cryptosystem-with-implementation/
[7] https://mathworld.wolfram.com/EuclideanAlgorithm.html

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, 21 Desember 2020

Vincent Chuardi / 13517103