Penggunaan Beberapa Algoritma Kriptografi Kunci-Simetri untuk Mensimulasikan Kriptografi Kunci-Publik

Fatur Rahman 13517056[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13517056@std.stei.itb.ac.id

***Abstract*—Kriptografi kunci-publik menggunakan dua kunci dalam proses enkripsi dan dekripsinya sedangkan kriptografi kunci-simetri hanya menggunakan satu kunci saja. Karena menggunakan dua kunci yang berbeda, kriptografi kunci-publik tentu lebih aman dalam menjaga kerahasiaan pesan. Namun, algoritma-algoritma kriptografi kunci-publik juga lebih kompleks dan memiliki waktu eksekusi lebih lama dibandingkan algoritma-algoritma kriptografi kunci-simetri. Dengan demikian, pada makalah ini, penulis akan menjelaskan bagaimana beberapa algoritma kriptografi kunci-simetri dapat digunakan untuk mensimulasikan kriptografi kunci-publik.**

***Keywords*—Dekripsi, Enkripsi, Kriptografi, Kunci-Publik, Kunci-Simetri.**

I. Pendahuluan

Kriptografi adalah ilmu dari menjaga rahasia rahasia [1]. Asumsikan pengirim yang mengacu pada *Alice* ingin mengirimkan pesan *m* ke pengirim yang mengacu pada *Bob*. Alice menggunakan saluran komunikasi yang tidak aman sehingga terdapat masalah ketika pesan yang dikirimkan bersifat rahasia. Pesan dapat disadap dan dibaca oleh penyadap atau lebih buruknya pesan itu diubah oleh musuh, yang mengacu pada *Eve,* tanpa disadari oleh *Bob*. Salah satu tujuan dari kriptografi adalah untuk menyediakan metode-metode untuk mencegah serangan-serangan tersebut. Secara umum, pesan atau *plaintext* akan dienkripsi terlebih dahulu menjadi *ciphertext* dengan kunci tertentu sebelum dikirim dan harus didekripsi kembali dengan kunci tertentu menjadi *plaintext* sebelum dibaca penerima.

Kita berbicara enkripsi simetris karena kedua pihak pada pasangan komunikasi menggunakan kunci *k* yang sama untuk enkripsi dan dekripsi [1]. Algoritma enkripsi dan dekripsi *E* dan *D* diketahui secara publik. Siapapun bisa mendekripsi sebuah *ciphertext*, jika dia mengetahui kuncinya. Sehingga, kunci *k* harus dijaga kerahasiaannya.

Masalah dasar dari skema simetris adalah bagaimana Alice dan Bob bisa setuju atas kunci rahasia bersama *k* dalam cara yang aman dan efisien [1]. Untuk pertukaran kunci ini, dibutuhkan metode-metode kriptografi kunci-publik. Tidak ada solusi atas masalah ini hingga konsep revolusioner dari kriptografi kunci-publik ditemukan sekitar 50 tahun yang lalu. Contoh algoritma kriptografi kunci-simetri adalah *Vernam cipher, DES,* dan *AES*.

Ide dasar dari kriptografi kunci-publik adalah kunci-kunci publik [1]. Kunci setiap orang dibagi menjadi dua bagian, yaitu kunci publik untuk enkripsi yang tersedia untuk siapapun dan kunci rahasia untuk dekripsi yang dijaga kerahasiaannya oleh pemilik kunci tersebut. Contoh algoritma kriptografi kunci-publik adalah *RSA*, *ElGamal*, dan *Rabin*.

Namun, tentu saja algoritma-algoritma kriptografi kunci-publik lebih kompleks dibandingkan algoritma-algoritma kriptografi kunci-simetri. Kriptografi kunci-publik juga membutuhkan waktu eksekusi yang lebih lama dibandingkan kriptografi kunci-simetri. Jadi, pada makalah ini penulis akan mencoba memberikan skema bagaimana penggunaan beberapa algoritma kunci-simetris dapat mensimulasikan skema kriptografi kunci-publik.

II. Dasar Teori

A. Kriptografi

Kriptografi adalah ilmu dari menjaga rahasia rahasia [1]. Asumsikan pengirim yang mengacu pada *Alice* ingin mengirimkan pesan *m* ke pengirim yang mengacu pada *Bob*. Alice menggunakan saluran komunikasi yang tidak aman. Sebagai contoh, saluran dapat berupa jaringan komputer atau jalur telepon. Terdapat masalah ketika pesan yang dikirimkan bersifat rahasia. Pesan dapat disadap dan dibaca oleh penyadap atau lebih buruknya pesan itu diubah oleh musuh, yang mengacu pada *Eve,* tanpa disadari oleh *Bob*.

Tugas dasar dan klasik dari kriptografi adalah menyediakan kerahasiaan dengan metode-metode enkripsi [1]. Pesan yang akan ditransmisikan (misalnya teks, data numerik, program *executable*, atau bentuk informasi lain) disebut dengan *plaintext*. Alice mengenkripsi *plaintext m* dan memperoleh *ciphertext c*. *Ciphertext* kemudian ditransmisikan kepada Bob. Untuk mendekripsi, Bob butuh informasi rahasia tertentu, yaitu sebuah kunci dekripsi rahasia. Musuh Eve mungkin masih bisa menyadap *ciphertext*. Namun, enkripsi harus menjamin kerahasiaan dan mencegah Eve untuk mengambil informasi apapun tentang *plaintext* dari *ciphertext* yang diamati.

Menyediakan kerahasiaan (*confidentiality*) bukan

satu-satunya tujuan kriptografi. Kriptografi juga digunakan untuk menyediakan solusi dari masalah-masalah lain [1]:

1. *Data integrity*. Penerima pesan harus bisa memeriksa apakah pesan diubah selama transmisi, entah itu secara tidak sengaja maupun tidak sengaja.
2. *Authentication*. Penerima pesan harus bisa memverifikasi asal pesan tersebut.
3. *Non-repudiation*. Pengirim harus tidak bisa nantinya untuk menyangkal kalau dia yang mengirim pesan.

B. Kriptografi Kunci-Simetri

Untuk memasang saluran komunikasi yang aman, Alice dan Bob pertama-tama harus setuju atas sebuah kunci k [1]. Mereka menjaga kerahasiaan kunci bersama mereka. Sebelum mengirim pesan m ke Bob, Alice mengenkripsi m dengan menggunakan algoritma enkripsi E dan kunci k. Dia memperoleh *ciphertext* $c = E(k, m)$ dan mengirim c ke Bob. Dengan menggunakan algoritma dekripsi D dan kunci k yang sama, Bob melakukan dekripsi terhadap c untuk memperoleh kembali *plaintext* $m = D(k, c)$.

Kita berbicara enkripsi simetris karena kedua pihak pada pasangan komunikasi menggunakan kunci k yang sama untuk enkripsi dan dekripsi [1]. Algoritma enkripsi dan dekripsi E dan D diketahui secara publik. Siapapun bisa mendekripsi sebuah *ciphertext*, jika dia mengetahui kuncinya. Sehingga, kunci k harus dijaga kerahasiaannya.

Masalah dasar dari skema simetris adalah bagaimana Alice dan Bob bisa setuju atas kunci rahasia bersama k dalam cara yang aman dan efisien [1]. Untuk pertukaran kunci ini, dibutuhkan metode-metode kriptografi kunci-publik yang akan dijelaskan kemudian. Tidak ada solusi atas masalah ini hingga konsep revolusioner dari kriptografi kunci-publik ditemukan sekitar 50 tahun yang lalu.

Kita membutuhkan bahwa *plaintext* m yang dienkripsi bisa dipulihkan secara unik dari *ciphertext* c [1]. Artinya, untuk kunci tetap k, peta enkripsi harus bijektif. Secara matematis, enkripsi simetris dapat dianggap sebagai berikut.

$$E : K \times M \rightarrow C,$$

sedemikian rupa sehingga untuk setiap $k \in K$, peta

$$E_k : M \rightarrow C,\ m \mapsto E(k,m)$$

adalah *invertible* (bisa dibalik). Elemen-elemen $m \in M$ adalah *plaintext*. C adalah kumpulan *ciphertext* atau *cryptogram,* elemen-elemen $k \in K$ adalah kunci. E_k disebut fungsi enkripsi dengan sehubungan dengan kunci k. Fungsi kebalikan $D_k := E_k^{-1}$ disebut fungsi dekripsi. Diasumsikan bahwa algoritma-algoritma efisien untuk menghitung E_k dan D_k itu ada.

Kunci k dibagikan antara pasangan komunikasi dan dijaga rahasia [1]. Kebutuhan keamanan dasar dari peta enkripsi E adalah, tanpa mengetahui kunci k, seharusnya mustahil untuk sukses menjalankan fungsi dekripsi D_k. Contoh-contoh penting dari skema enkripsi kunci-simetris adalah *Vernam's one-time pad, DES,* dan *AES*.

Dari semua algoritma enkripsi, algoritma enkripsi kunci-simetri memiliki implementasi tercepat dalam baik perangkat keras maupun perangkat lunak. Sehingga, mereka sangat cocok digunakan untuk enkripsi data berjumlah besar. Jika Alice dan Bob ingin menggunakan skema enkripsi kunci-simetri, mereka harus berbagi kunci rahasia terlebih dahulu. Untuk ini, mereka harus menggunakan sebuah saluran komunikasi yang aman. Metode-metode enkripsi kunci-publik sering digunakan untuk tujuan ini. Skema enkripsi kunci-publik adalah kurang efisien dan akibatnya tidak cocok untuk data berjumlah besar. Sehingga, enkripsi kunci-simetri dan kunci-publik melengkapi satu sama lain untuk menyediakan *cryptosystem* yang praktis. Kriptografi kunci-simetri dapat dibedakan menjadi dua, yaitu *stream cipher* dan *block cipher*

C. Stream Cipher

Misalkan K adalah himpunan dari kunci dan M adalah himpunan dari *plaintext* [1]. Dalam konteks ini, elemen-elemen dari M disebut dengan karakter. Sebuah *stream cipher*

$$E^* : K^* \times M^* \rightarrow C^*,\ E^*(k, m) := c := c_1c_2c_3...$$

mengenkripsi sebuah *stream* $m := m_1m_2m_3... \in M^*$ dari karakter-karakter *plaintext* $m_i \in M$ sebagai sebuah stream $c := c_1c_2c_3... \in C^*$ dari karakter-karakter $c_i \in C$ dengan menggunakan sebuah kunci *stream* $k := k_1k_2k_3... \in K^*$, $k_i \in K$.

Stream plaintext $m = m_1m_2m_3...$ dienkripsi secara karakter demi karakter

[1]. Untuk tujuan ini, terdapat sebuah peta enkripsi

$$E : K \times M \rightarrow C,$$

yang mengenkripsi satu karakter-karakter *plaintext* m_i dengan karakter kunci k_i yang bersesuaian:

$$c_i = E_{ki}(m_i) = E(k_i, m_i),\ i = 1,2,....$$

Umumnya, karakter-karakter dalam M dan C dan elemen-elemen kunci dalam K adalah bilangan biner atau *byte* [1].

Tentu saja, mengenkripsi karakter-karakter *plaintext* dengan E_{ki} harus merupakan peta bijektif, untuk setiap karakter kunci $k_i \in K$ [1]. Mendekripsi sebuah *stream ciphertext* $c := c_1c_2c_3...$ dilakukan secara karakter demi karakter dengan menerapkan peta dekripsi D dengan *stream key* $k := k_1k_2k_3...$ yang sama yang digunakan untuk enkripsi:

$$c := c_1c_2c_3... \mapsto D(k, c) := D_{k1}(c_1)D_{k2}(c_2)D_{k3}(c_3)....$$

Tentu saja, kunci *stream* dalam *stream cipher* harus dijaga rahasia [1]. Tapi kunci tersebut tidak harus kunci rahasia yang dibagi antara pasangan komunikasi, kunci *stream* bisa dibangkitkan dari kunci rahasia bersama dengan sebuah pembangkit bilangan acak semu.

D. Block Cipher

Sebuah *block cipher* adalah sebuah skema enkripsi kunci-simetris dengan $M = C = \{0, 1\}^n$ dan ruang kunci $K = \{0, 1\}^r$ [1]:

$$E :\ \{0, 1\}^r \times \{0, 1\}^n \rightarrow \{0, 1\}^n,\ (k, m) \mapsto E(k, m).$$

Dengan menggunakan sebuah kunci rahasia k dengan panjang biner r, algoritma enkripsi E mengenkripsi blok-blok *plaintext* m dari panjang biner tetap n dan menghasilkan blok-blok *ciphertext* $c = E(k, m)$ yang juga memiliki panjang n [1]. n disebut dengan panjang blok dari *cipher*.

Panjang blok umumnya adalah 64 (seperti dalam *DES*) atau 128 (seperti dalam *AES*), panjang kunci umumnya aalah 56 (seperti dalam *DES*) atau 128, 192, dan 256 (seperti dalam *AES*) [1].

Misalkan sebuah *block cipher* E dengan panjang blok n dan

panjang kunci r [1]. Terdapat 2^n blok-blok *plaintext* dan 2^n blok-blok *ciphertext* dengan panjang n. Untuk kunci tetap k, fungsi enkripsi $E_k : m \mapsto E(k, m)$ memetakan $\{0, 1\}^n$ secara bijektif ke $\{0, 1\}^n$, ini adalah permutasi dari $\{0, 1\}^n$. Sehingga, untuk memilih sebuah kunci k, artinya adalah untuk memilih sebuah permutasi E_k dari $\{0, 1\}^n$, dan permutasi ini kemudian digunakan untuk mengenkripsi blok-blok *plaintext*. Permutasi-permutasi 2^r dari E_k, dengan k berjalan melalui himpunan $\{0, 1\}^r$ dari kunci-kunci, membentuk sebuah subset kecil yang hampir bisa diabaikan dalam himpunan yang sangat besar dari semua permutasi dari $\{0, 1\}^n$, yang terdiri dari $2^n!$ elemen. Jadi, ketika kita memilih sebuah kunci r-bit untuk E secara acak, maka kita membatasi pilihan kita terhadap permutasi enkripsi menjadi sebuah subset yang sangat kecil.

Berdasarkan pertimbangan-pertimbangan ini, disimpulkan bahwa kita tidak bisa memiliki *block cipher* ideal dengan kerahasiaan sempurna dalam praktiknya [1]. Kerahasiaan sempurna adalah hasil dari keacakan maksimum, yaitu untuk setiap bit pesan, sebuah kunci bit acak dipilih. Disimpulkan bahwa level maksimal dari keamanan dalam sebuah *block cipher* juga membutuhkan sebuah keacakan maksimum, dan ini berarti bahwa ketika memilih sebuah kunci, kita harus memilih sebuah elemen acak dari himpunan semua permutasi dari $\{0, 1\}^n$. Sayangnya, ini ternyata menjadi sangat tidak praktis. Kita bisa mencoba untuk mengenumerasi semua permutasi π dari $\{0, 1\}^n$, $\pi_1\pi_2\pi_3\ldots$, dan kemudian dipilih satu secara acak dengan memilih sebuah indeks secara acak (indeks ini akan menjadi kunci). Karena terdapat $2^n!$ permutasi, kita butuh $\log_2(2^n!)$-bit bilangan untuk *encode* indeks-indeksnya. Oleh formula penaksiran Stirling $k! \approx \sqrt{2\pi k}\left(\frac{k}{e}\right)^k$, kita menurunkan bahwa $\log_2(2^n!) \approx (n - 1.44)2^n$. Ini adalah angka yang sangat besar. Untuk sebuah panjang blok n dari 64 bit, kita membutuhkan sekitar 2^{67} *byte* untuk menyimpan sebuah kunci. Tidak terdapat medium penyimpanan dengan kapasitas tersebut.

Sehingga, dalam sebuah *block cipher* nyata, kita harus membatasi ke jumlah kunci yang jauh lebih kecil dan memilih permutasi enkripsi E_k untuk sebuah kunci k dari himpunan yang jauh lebih kecil dari 2^r permutasi, dengan r umumnya dalam jangkauan 56 hingga 256 [1]. Meskipun demikian, perancang-perancang dari sebuah *block cipher* mencoba untuk menaksir yang ideal. Idenya adalah untuk mendapatkan sebuah fungsi enkripsi yang berperilaku seperti sebuah fungsi yang dipilih secara acak dari himpunan yang sangat besar dari semua permutasi.

E. Kriptografi Kunci-Publik

Dalam skema enkripsi kunci-publik, pasangan komunikasi tidak berbagi sebuah kunci rahasia [1]. Setiap pengguna memiliki satu pasang kunci, yaitu kunci rahasia *sk* yang hanya diketahui olehnya dan kunci publik *pk* yang diketahui semua orang.

Misalkan Bob memiliki pasangan kunci (*pk, sk*), dan Alice ingin mengenkripsi sebuah pesan m untuk Bob [1]. Seperti semua orang lainnya, Alice tahu kunci publik *pk* Bob. Alice menghitung *ciphertext* $c = E(pk, m)$ dengan menerapkan fungsi enkripsi E dengan kunci publik *pk* Bob. Seperti sebelumnya, kita menunjukkan enkripsi dengan kunci tetap *pk* dengan E_{pk}, yaitu, $E_{pk}(m) := E(pk, m)$. Jelas, skema enkripsi hanya dapat aman jika tidak mungkin secara praktis untuk menentukan m dari $c = E_{pk}(m)$. Namun bagaimana Bob kemudian memulihkan pesan m dari *ciphertext* c? Di sinilah digunakan kunci rahasia Bob. Fungsi enkripsi E_{pk} harus memiliki properti bahwa gambar awal m dari *ciphertext* $c = E_{pk}(m)$ adalah mudah dihitung menggunakan kunci rahasia *sk* Bob. Karena hanya Bob yang mengetahui kunci rahasia, hanya dia yang bisa melakukan dekripsi pesan. Bahkan Alice, yang mengenkripsi pesan m, tidak bisa memperoleh m dari $E_{pk}(m)$ jika dia kehilangan m. Tentu saja, algoritma-algoritma efisien harus ada untuk melakukan enkripsi dan dekripsi.

Kita simpulkan kebutuhan-kebutuhan dari kriptografi kunci-publik [1]. Kita mencari sebuah rumpun fungsi $(E_{pk})_{pk \in PK}$ sedemikian rupa sehingga setiap fungsi E_{pk} adalah bisa dihitung oleh sebuah algoritma efisien. Rumpun $(E_{pk})_{pk \in PK}$ tersebut disebut dengan rumpun dari fungsi satu-arah atau fungsi satu arah singkatnya. Di sini, *PK* menunjukkan kumpulan kunci publik yang tersedia. Untuk setiap fungsi E_{pk} dalam rumpun, harusnya terdapat informasi *sk* tertentu untuk dijaga rahasia yang memungkinkan komputasi efisien dari *inverse* dari E_{pk}. Informasi rahasia ini disebut dengan informasi *trapdoor*. Fungsi-fungsi satu-arah dengan properti ini disebut dengan fungsi *trapdoor*.

Setiap partisipan dalam *cryptosystem* kunci-publik butuh kunci pribadi $k = (pk, sk)$, terdiri dari bagian publik dan bagian rahasia (juga disebut *private*) [1]. Untuk menjamin keamanan dari *cryptosystem*, harus tidak dimungkinkan untuk menghitung kunci rahasia *sk* dari kunci publik *pk*, dan harus dimungkinkan untuk memilih kunci-kunci k secara acak dari ruang parameter yang besar. Sebuah algoritma efisien harus tersedia untuk melakukan pemilihan acak ini. Jika Bob ingin ikut dalam *cryptosystem*, dia memilih kunci $k = (pk, sk)$ secara acak, menjaga kerahasiaan *sk* dan mengumumkan *pk*. Sekarang semua orang bisa menggunakan *pk* untuk mengenkripsi pesan untuk Bob. Contoh algoritma-algoritma kriptografi kunci-publik adalah *RSA, ElGamal,* dan *Rabin*.

Untuk mendiskusikan ide dasar dari tanda tangan digital, kita asumsikan bahwa kita memiliki sebuah rumpun $(E_{pk})_{pk \in PK}$ dari fungsi *trapdoor* dan bahwa setiap fungsi E_{pk} adalah bijektif [1]. Rumpun permutasi *trapdoor* tersebut bisa digunakan untuk tanda tangan digital. Misalkan *pk* menjadi kunci publik Alice. Untuk menghitung *inverse* E_{pk}^{-1} dari E_{pk}, kunci rahasia *sk* Alice dibutuhkan. Jadi Alice adalah satu-satunya yang bisa melakukan hal ini. Jika Alice ingin menandatangani pesan m, dia menghitung $E_{pk}^{-1}(m)$ dan mengambil nilainya sebagai tanda tangan s dari m. Semua orang bisa memverifikasi tanda tangan s Alice dengan menggunakan kunci publik *pk* Alice dan menghitung $E_{pk}(s)$. Jika $E_{pk}(s) = m$, Bob yakin bahwa Alice benar-benar menandatangani m karena hanya Alice yang bisa menghitung $E_{pk}^{-1}(m)$.

Sebuah aplikasi mudah yang penting dari *cryptosystem* kunci-publik adalah distribusi dari kunci-kunci sesi[1]. Sebuah kunci sesi adalah sebuah kunci rahasia yang digunakan dalam skema enkripsi simetris klasik untuk mengenkripsi pesan dari sebuah sesi komunikasi tunggal. Jika Alice mengetahui kunci publik Bob, maka dia bisa membangkitkan sebuah kunci sesi,

mengenkripsinya dengan kunci publik Bob dan mengirimkannya ke Bob. Tanda tangan digital digunakan untuk menjamin kebenaran dari kunci publik dengan otoritas sertifikasi. Otoritas sertifikasi menandatangani kunci publik dari setiap pengguna dengan kunci rahasia Alice. Tanda tangan dapat diverifikasi dengan kunci publik dari otoritas sertifikasi. Protokol kriptografis untuk otentikasi pengguna dan protokal kriptografis lanjut, seperti skema komitmen bit, transfer tidak sadar dan sistem bukti interaktif berpengetahuan nol, telah dikembangkan. Hari ini mereka menjadi sangat penting untuk komunikasi Internet dan perdagangan elektronik. Kriptografi kunci-publik juga penting untuk ilmu komputer teoritis, teori keamanan dikembangkan dan dampaknya terhadap teori kompleksitas harus disebutkan.

III. Rancangan dan Protokol Solusi

Solusi dapat menggunakan berbagai algoritma kriptografi kunci-simetri secara bersamaan, seperti algoritma *Vernam cipher, AES, DES*, *Blowfish,* dan lain-lain. Semua algoritma tersebut akan digunakan untuk mengenkripsi pesan. Solusi akan memerlukan kunci publik dan kunci rahasia terlebih dahulu.

Kunci publik adalah sebuah *string* dengan panjang karakter berapapun. Kunci publik akan dibagi menjadi beberapa *substring* yang masing-masingnya akan digunakan dalam ketiga algoritma kunci-simetri. Pembentukan *substring* tergantung pada kunci rahasia.

Kunci rahasia adalah sebuah bilangan yang didalamnya terkandung informasi-informasi. Informasi-informasi tersebut adalah sebagai berikut.

1. Bagaimana urutan ketiga algoritma pada proses enkripsi.
   Ada 6 kemungkinan bagaimana algoritma tersebut diurutkan diurutkan jika terdapat 3 algoritma. Atau terdapat *s!* jika terdapat *s* algoritma kriptografi kunci-simetri yang digunakan. Misalkan digunakan 3 algoritma, yaitu algoritma *Vernam cipher*, *AES*, dan *DES*. Jika algoritma *Vernam cipher* ditandai dengan *a*, *AES* ditandai dengan *b*, dan *DES* ditandai dengan *c,* urutan-urutan dari ketiga algoritma tersebut secara alfabetis adalah *abc, acb, bac, bca, cab,* dan *cba*. Jika 6 kemungkinan tersebut diurutkan secara alfabetis dan *sk* adalah kunci rahasia, urutan algoritma *n* yang dipakai dalam proses enkripsi ditentukan dengan
   $$n = sk \bmod 6.$$
   Jadi, jika *n* adalah 3 maka urutan algoritma yang digunakan dengan daftar urutan yang telah disebutkan adalah *bac*, yaitu *AES, Vernam cipher*, dan terakhir *DES*.
2. Bagaimana proses enkripsi dilakukan.
   Jika kunci rahasia adalah *sk*, proses enkripsi dilakukan sebanyak *sk* kali dengan urutan enkripsi sesuai yang telah dijelaskan sebelumnya. Jadi, misalkan *sk* adalah 5 dan urutan algoritma dan algoritma-algoritma yang digunakan seperti contoh yang telah dijelaskan sebelumnya ketika *sk* mod 6 bernilai 3 maka proses enkripsi yang dilakukan secara berturut-turut adalah *Vernam cipher, AES*, *DES*, *Vernam cipher*, dan *AES*. Pesan yang akan dienkripsi juga akan dibagi menjadi *sk* bagian yang akan dienkripsi dengan algoritma dan kunci yang berbeda.
3. Bagaimana *string* kunci publik dibagi menjadi *substring*.
   Jika kunci rahasia adalah *sk,* proses enkripsi akan dilakukan sebanyak *sk* kali. Sehingga agar setiap proses enkripsi dilakukan dengan kunci-kunci yang berbeda, diperlukan *sk* buah kunci. Jika kunci publik adalah *pk* dengan panjang *n*, *pk* akan dibagi menjadi *sk substring* dengan karakter ke-*i* pada *pk* akan menjadi bagian dari *substring* ke-*j* dengan
   $$j = i \bmod sk.$$
   Jika *n* lebih kecil dari *sk* maka *substring* ke $j > n$ diisi oleh karakter ke-*l* dengan
   $$l = j \bmod n.$$
   Jadi, misalkan *pk* adalah abcdefghij dengan panjang *n* adalah 10, *sk* adalah 4, maka *pk* akan dibagi menjadi *substring-substring* aei, bfj, cg, dan dh. Dengan *sk* mod 6 adalah 3 dan urutan algoritma dan algoritma-algoritma yang digunakan seperti yang telah dijelaskan sebelumnya, dilakukan empat kali enkripsi dengan enkripsi pertama menggunakan *Vernam cipher* dengan kunci aei, kemudian *AES* menggunakan kunci bfj, lalu *DES* dengan kunci cg, dan terakhir *Vernam cipher* kembali dengan kunci dh.
4. Bagaimana proses dekripsi dilakukan.
   Tentunya proses dekripsi adalah kebalikan dari proses enkripsi. Dengan kunci publik dan kunci rahasia, dilakukan dekripsi dengan proses dekripsi sesuai dengan urutan algoritma, pembagian *substring* dari kunci publik, proses dekripsi sesuai dengan yang diimplikasikan oleh kunci rahasia. Jadi, dengan contoh sesuai pada poin-poin yang dijelaskan sebelumnya, dengan kunci rahasia yang diberikan, dilakukan dekripsi dengan terlebih dahulu melakukan dekripsi oleh *Vernam cipher* dengan kunci aei, kemudian *AES* menggunakan kunci bfj, lalu *DES* dengan kunci cg, dan terakhir *Vernam cipher* kembali dengan kunci dh.

Protokol komunikasi dari rancangan solusi yang dijelaskan adalah sebagai berikut.

1. Alice dan Bob menyepakati algoritma-algoritma kriptografi kunci-simetri apa saja yang akan digunakan.
2. Bob mengirimi Alice kunci publiknya (kunci publik Bob).
3. Alice mengenkripsi pesannya dengan kunci publik Bob lalu mengirimkannya ke Bob.
4. Bob melakukan dekripsi terhadap pesan dari Alice dengan kunci rahasia miliknya (kunci rahasia Bob).

IV. Analisis Kinerja

Analisis komparatif dari beberapa metrik untuk beberapa algoritma kunci-simetri dapat dilihat pada tabel berikut [2].

Tabel 1. Analisis Komparatif Beberapa Algoritma Kriptografi Kunci-Simetri

Algoritma	Konsumsi waktu	Konsumsi baterai
DES	Lambat	Medium
3DES	Sangat lambat	Tinggi
AES	Cepat	Tinggi
Blowfish	Sangat cepat	Paling rendah
HiSea	Medium	Rendah
RC4	Cepat	Tinggi
RC6	Cepat	Medium
RC2	Cepat	Tinggi
TEA	Cepat	Rendah
CAST	Cepat	Tinggi
Twofish	Lambat	Rendah
Serpent	Cepat	Medium

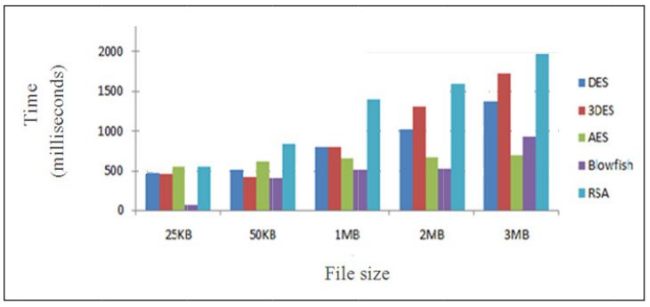
Analisis komparatif dari beberapa metrik untuk beberapa algoritma kunci-publik dapat dilihat pada tabel berikut [2].

Tabel 2. Analisis Komparatif Beberapa Algoritma Kriptografi Kunci-Publik

Algoritma	Konsumsi waktu	Konsumsi baterai
RSA	Paling lambat	Rendah
Diffie-Hellman	Medium	Tinggi
ECC	Cepat	Medium

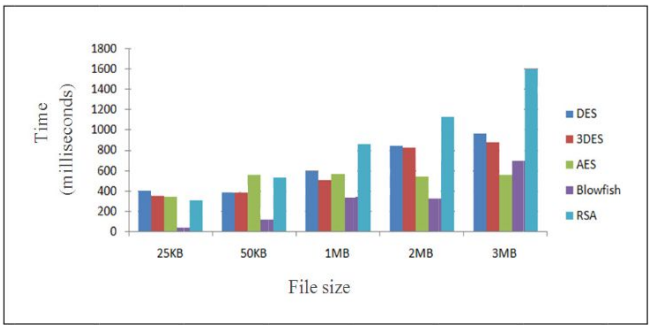
Untuk analisis komparatif yang lebih spesifik terhadap waktu pada proses enkripsi untuk beberapa algoritma kunci-simetris dan kunci-publik dapat dilihat pada gambar berikut [3].

Gambar 1. Waktu enkripsi terhadap ukuran berkas untuk *DES, 3DES, AES, Blowfish,* dan *RSA*



Untuk analisis komparatif yang lebih spesifik terhadap waktu pada proses dekripsi untuk beberapa algoritma kunci-simetris dan kunci-publik dapat dilihat pada gambar berikut [3].

Gambar 2. Waktu dekripsi terhadap ukuran berkas untuk *DES, 3DES, AES, Blowfish,* dan *RSA*



Dari data-data tersebut terlihat bahwa secara umum algoritma-algoritma kunci-simetri memang lebih cepat dibandingkan algoritma-algoritma kunci-publik. Bahkan untuk beberapa algoritma kunci-simetri yang digunakan sekaligus masih lebih cepat dibandingkan algoritma kunci-publik tertentu. Hal inilah yang membuat pada kasus tertentu melakukan enkripsi dengan menggunakan beberapa algoritma kriptografi kunci-simetri bersamaan menjadi lebih baik dibandingkan dengan algoritma kriptografi kunci-publik tertentu. Apalagi dengan panjang pesan dan panjang kunci publik yang sebelumnya cukup panjang yang kemudian dibagi-bagi menjadi beberapa bagian yang lebih singkat, proses enkripsi atau dekripsi dari masing-masing algoritma kunci-simetri bisa menjadi lebih cepat lagi.

V. Kesimpulan dan Saran

Pada makalah ini diberikan rancangan solusi tentang bagaimana penggunaan beberapa algoritma kriptografi kunci-simetri dapat mensimulasikan skema algoritma kriptografi kunci-publik. Penggunaan beberapa algoritma kriptografi kunci-simetri secara bersamaan dapat mensimulasikan skema kriptografi kunci-publik dengan waktu yang dapat dibandingkan dengan algoritma kriptografi kunci-publik biasa bahkan mungkin bisa lebih cepat dengan kompleksitas yang lebih sederhana dibandingkan algoritma kunci-publik. Namun, tentu saja juga terdapat algoritma-algoritma kunci-simetri yang umumnya berjalan

lambat dan penggunaan bersamaan dari algoritma-algoritma tersebut mengakibatkan waktu eksekusi total yang bisa menjadi lebih lambat dibandingkan algoritma-algoritma kunci publik.

Untuk kedepannya, mekanisme kunci rahasia dapat dikembangkan lagi, seperti pada informasi pembagian kunci publik atau pesan agar bisa menjalankan masing-masing algoritma dengan lebih cepat. Informasi rahasia pada kunci rahasia juga bisa ditambahkan seperti informasi untuk pemilihan prioritas algoritma kunci-simetris yang lebih cepat juga bisa dilakukan agar algoritma yang lebih cepat tersebut lebih sering digunakan. Kunci rahasia juga dapat didefinisikan kembali seperti bisa berupa karakter bukan numerik agar terlihat lebih kompleks.

REFERENSI

[1] Delfs, Hans, Helmut Knebl, and Helmut Knebl. *Introduction to cryptography*. Vol. 2. Heidelberg: Springer, 2002..

[2] Al-Shabi, M. A. "A Survey on Symmetric and Asymmetric Cryptography Algorithms in information Security." International Journal of Scientific and Research Publications 9.3 (2019).

[3] Patil, Priyadarshini, et al. "A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and Blowfish." Procedia Computer Science 78.1 (2016): 617-624.

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

Fatur Rahman 13517056