Perbandingan Antar Algoritma Untuk Mencari Algoritma Kriptografi Kunci-Publik Dan Pembangkit Bilangan Acak Terbaik

James Filipus - 13507087
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if17087@students.if.itb.ac.id

Abstract*—Pada suatu kegiatan saling mengirim pesan, keamanan dan kerahasiaan dari pesan sangatlah penting oleh karena itu algoritma enkripsi pesan yang kuat banyak dicari orang. Akan tetapi salah satu kelemahan dari algoritma ennkripsi biasa yaitu cara penyampaian kunci yang harus secara langsung sehingga mungkin disadap orang lain. Dan bila kunci diketahui orang lain maka cipher teks akan dengan mudah dipecahkan. Oleh karena itu digunakanlah algoritma kriptografi kunci-publik sehingga tidak perlu ada proses memberi tahu kunci secara langsung. Akan tetapi manakah algoritma yang terbaik, sehingga dapat memenuhi kebutuhan banyak orang yang mencari algoritma kriptografi yang kuat. Pada makalah ini akan dilakukan perbandingan antar beberapa algoritma kunci-publik, kemudian dilakukan analisis untuk mencari yang terbaik. Diantaranya yaitu akan dibahas pula algoritma persutujuan kunci. Semakin rumit algoritma hubungan antara kunci privat dengan kunci publik maka akan membuat suatu algoritma kunci-publik semakin baik, kuat dan sulit untuk dipecahkan. Serta algoritma kunci-publik masih berkaitan erat dengan pembangkit bilangan acak yang biasa digunakan untuk membangkitkan pasangan kunci privat dan kunci publik, maka pada makalah ini juga akan dibahas mengenai pembangkit bilangan acak dan kemampuannya mendukung algoritma kunci-publik. Setelah hasil perbandingan diketahui maka akan dianalisis untuk menjadi referensi dalam membangun algoritma kunci-publik terbaik sebagai ide baru dan kontribusi dari penulis. Diharapkan dengan algoritma kunci-publik yang dibangun akan meningkatkan rasa aman dan privasi baik bagi pengirim maupun penerima pesan.

***Index Terms*—**keamanan, kerahasiaan, algoritma enkripsi, disadap, algoritma kunci-publik, pembangkit bilangan acak.

I. PENDAHULUAN

Dalam kehidupan sehari-hari, sudah merupakan suatu kebutuhan primer bagti manusia untuk slaing bertukar informasi baik secara langsung maupun melalui berbagi media perantara. Informasi tersebar bebas dan dapat diakses dengan mudah dengan teknologi yang ada sekarang ini, terutama dengan adanya internet berbagai informasi dapat diperoleh oleh siapapun, dari manapun dan kapanpun. Hal tersebut justru menjadikan proses beertukar informasi menjadi rentan terhadap penyadapan atu bahkan pembajakan informasi dengan tujuan mengacaukan alur dan isi dari informasi yang ada. Dampak dari hal ini yaitu penyampaian informasi yang bersifat pribadi maupun informasi yang bersifat rahasia menjadi tidak terjamin lagi kerahasiaan dan keamanan dari informasi yang disampaikan. Untuk mengatasi hal itu manusia telah mencoba membuat pesan yang hendak disampaikan menjadi kode-kode atau susunan karakter lain yang tidak memiliki arti dengan algoritma enkripsi. Cara ini cukup efektif karena informasi / pesan menjadi sulit dipecahkan dan tidak dapat dipahami oleh orang yang tidak berkepentingan. Akan tetapi kelemahan dari algoritma enkripsi pada umumnya yaitu digunakannya kunci untuk mendekripsi dan mengenkripsi pesan. Dan tentu saja kunci ini harus diketahui oleh pengirim serta penerima pesan agar pesan yang sudah berhasil terkirim dapat didekripsi dan pesan yang asli dapat dipahami oleh penerima. Apabila kunci ini diketahui oleh orang yang tidak berkepentingan maka dengan mudah pesan yang asli akan diketahui. Dan kemungkinan untuk bocornya kunci ini menjadi semakin besar karena pengirim perlu memberitahukan kunci yang digunakan kepada penerima secara langsung maupun menggunakan media informasi. Seperti yang sudah dibahas di atas, maka cara penyampaian pesan menggunakan algoritma enkripsi biasa menjadi semakin rentan terhadap serangan karena penyampaian kunci tersebut.

Oleh karena itu digunakanlah kriptografi dengan kunci publik.

II. KRIPTOGRAFI KUNCI PUBLIK

Kriptografi kunci publik adalah pendekatan kriptografi yang melibatkan penggunaan algoritma kunci asimetrik, Tidak seperti algoritma kunci simetrik, algoritma kunci publik ini lebih aman karena tidak membutuhkan pertukaran dari satu atau lebih kunci rahasia sebelum memulia pengiriman dan penerimaan pesan untuk kedua belah pihak baik pengirim maupun penerima. Algoritma kunci asimetrik digunakan untuk menciptakan sepasang kunci yang berhubungan secara matematis, yaitu sebuah kunci pribadi rahasia (kunci privat) dan kunci publik yang tidak bersifat rahasia melainkan diumumkan (bisa

diketahui semua orang). Kedua kunci ini pada umunya digunakan untuk dua tujuan, pertama yaitu unutuk menjaga keaslian sebuah pesan dengan memberi tanda tangan digital pada sebuah pesan menggunakan kunci privat dan kemudian keaslian pesan dapat diperiksa dengan mendekripsi tanda tangan digital tersebut menggunakan kunci publik dari pengirim pesan dan dicocokan dengan tanda tangan digital asli dari pengirim. Tujuan kedua yaitu untuk mengenkripsi dan menjaga kerahasiaan sebuah pesan dengan mengenkripsi pesan menggunakan kunci publik dari orang yang akan dikirimi pesan, kemudian pesan tersebut dapat didekripsi oleh penerima pesan menggunakan kunci privatnya sendiri.

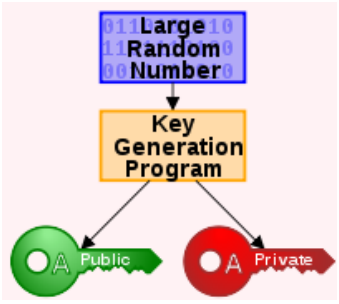
Pada makalah ini akan dibahas lebih mengenai fungsi kedua dari kriptografi kunci publik yaitu untuk menjaga kerahasiaan dari pesan (enkripsi kunci publik).

III. CARA KERJA ENKRIPSI KUNCI-PUBLIK

Dalam mengenkripsi sebuah pesan dengan menggunakan kunci publik, ada dua langkah utama yang perlu diperhatikan yaitu membangkitkan kunci publik dan kunci privat dan menggunakan algoritma untuk mengenkripsi pesan.

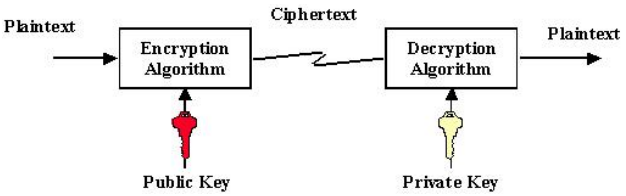
Algoritma enkripsi kunci publik meruapakan algoritma enkripsi asimetrik, dimana kunci untuk enkripsi berbeda dengan kunci yang digunakan untuk dekripsi. Dalam konteks yang kita bahas sekarang ini yaitu kunci publik penerima untuk enkripsi dan kunci privat penerima untuk dekripsi pesan. Kunci publik dimumkan sehingga orang yang hendak mengirim pesan rahasia pada penerima dapat menggunakannya untuk mengenkripsi pesan, sedangkan kunci privat harus dijaga oleh penerima kerahasiaannya agar hanya penerima yang dapat mendekripsi pesan yang ditujukan baginya.

Setiap orang yang terlibat dalam kirim-mengirim pesan rahasia menggunakan kriptografi kunci publik mempunyai kunci publik dan kunci privat masing-masing yang saling berhubungan secara matematis. Kunci privat yang merupakan rahasia tidak dapat diturunkan dari kunci publiknya yang diumumkan meskipun pada dasarnya kedua kunci tersebut saling berhubungan. Untuk ememcahkan hubungan kedua kunci ini perlu diketahui algoritma yang digunakan untuk membangkitkan kedua kunci ini. Oleh karena itu semakin rumit dan semakin tidak umum algoritma yang digunakan maka akan semakin sulit untuk memecahkannya.



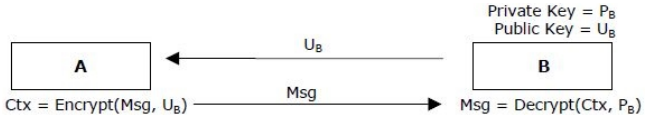
Gambar 1. Pembangkitan kunci publik dan kunci privat

Pada algoritma pembangkitan kunci, biasanya dibutuhkan angka acak sebagai bibit dari kunci yang akan dibangkitkan.



Gambar 2. Cara kerja kriptografi kunci-publik

Cara kerja dari algoritma enkripsi kunci publik dapat dilihat dari gambar di atas, untuk lebih jelasnya maka akan dibahas sebuah contong pengenkripsian pesan.



Gambar 3. Pengenkripsian pesan dengan kritografi kunci publik

B adalah seorang penerima pesan dengan kunci publik U_B dan kunci privat P_B. Karena U_B merupakan kunci publik maka semua orang dapat mengaksesnya dan mengetahuinya. Bagi setiap orang (A) yang ingin mengirimkan pesan rahasia (Msg) kepada B maka ia dapat menggunakan U_B untuk mengenkripsi Msg menggunakan algoritma enkripsi tertentu sehingga mengasilkan cipherteks (Ctx). Pesan dikirim ke B dan pesan tersebut hanya bisa didekripsi menggunakan P_B yang hanya dimiliki oleh B maka tidak ada orang lain selain B bahkan A pun tidak bis mendekripsi pesan tersebut.

IV. ENKRIPSI KUNCI PUBLIK

Berikut akan dibahas beberapa algoritma kunci publik:

A. RSA

RSA (Rivest-Shamir-Adelman) adalah algoritma kunci publik yang paling umum digunakan. RSA dapat digunakan baik unruk enkripsi pesan maupun untuk tanda tangan digital dari pesan, tetapi yang akan dibahas pada makalah ini adalah RSA untuk enkripsi pesan.

RSA umumnya cukup aman bila kunci yang digunakan cukup panjang (512 bit tidak aman, 768 bit cukup aman, 1024 bit 2048 bit aman dan harus tetap aman untuk masa mendatang). Semua algoritma kunci publik membutruhkan kunci yang sangat panjang karena semakin panjang kunci maka semakin aman, tidak seperti algoritma kunci privat yang tidak membutuhkan kunci yang panjang. Algoritma kunci privat dengan panjang kunci 128 bit akan lebih aman daripada sistem kunci publik dengan panjang kunci 1024 bit, akan tetapi resiko bocornya kunci bagi algoritma kunci privat tetap lebih besar.

Keamanan RSA bergantung pada kesulitan memfaktorkan bilangan integer besar, integer besar ini adalah hubungan matematis antara kunci publik dan kunci privat seseorang. Jika ada seorang matematikawan yang dapat membuat kemajuan dramatis secara tiba-tiba dengan menemukan teknik untuk memfaktorkan integer besar maka segera akan membuat lebih mudah untuk memecahkan RSA. RSA sangat rentan terhadap serangan plaintext yang dipilih. Ada juga serangan waktu baru yang dapat digunakan untuk memecahkan banyak implementasi RSA. Algoritma RSA diyakini aman bila digunakan dengan benardan teapat, tetapi kita harus berhati-hati ketika menggunakannya untuk menghindari serangan. Berikut adalah langkah-langkah dalam enkripsi RSA :

Pembangkitan parameter

- Pilih dua bilangan prima p and q.
- Hitung n=p*q, Dimana n adalah modulus yang dibuat publik dan panjang dari n dianggap sebagai panjang kunci RSA.
- Pilih bilangan acak ‘e’ sebagai kunci publik dalam lingkup 0<e<(p-1)(q-1) dimana gcd(e,(p-1)(q-1))=1.
- Hitung kunci privat d dimana ed≡1(mod (p-1)(q-1)).

Enkripsi

Dimisalkan A hendak mengirim pesan kepada B.

- e adalah kunci publik B. Karena e bersifat publik maka A dapat mengakses e.
- untuk mengenkripsi pesan M, representasikan pesan M sebagai integer(ASCII) dan dibagi-bagi dengan panjang 0<M<n.
- Cipher text C = M^emod n, dimana n adalah modulusnya.

Dekripsi

- C adalah cipher teks yang diterima B dari A.
- Kalkulasi pesan M = C^d mod n, dimana d adalah kunci privat dari B dan n adalah modulus.

B. Pailier

Pembangkitan Kunci

- Pilih dua bilangan prima besar p dan q secara random dan independen satu sama lain sehingga gcd(pq,(p − 1)(q − 1)) = 1. Properti in dapat dipastikan bila kedua bilangan prima memiliki panjang yang ekivalen, $p, q \in 1||\{0,1\}^{s-1}$untuk parameter keamanan s.
- Hitung *n* = *pq* and $\lambda = \mathrm{lcm}(p-1, q-1)$.
- Pilih integer acak g dimana $g \in \mathbb{Z}^*_{n^2}$
- Memastikan n membagi order dari g dengan memeriksa keberadaan dari invers multiplikatif modular berikut : $\mu = (L(g^\lambda \mod n^2))^{-1} \mod n$ dimana fungsi L didefinisikan sebagai berikut $L(u) = \frac{u-1}{n}$. Perlu diperhatikan bahwa notasi a/b tidak menyatakan multiplikasi modular dari a kali invers multiplikatif modular dari b tetapi hasil bagi dari a dibagi bdoes not denote the modular multiplication of *a* times the modular multiplicative inverse of *b* but rather the quotient of *a* divided by byaitu nilai integer terbesar $v \geq 0$ untuk memenuihi hubungan $a \geq vb$.

- ✓ Kunci publik : (*n*,*g*).
- ✓ Kunci privat : (λ,μ).

Bila menggunakan p dan q dengan panjang ekivalen, maka varian yang lebih sederhana dari pembangkitan kunci di atas dapat digunakan

$$g = n + 1, \lambda = \varphi(n)$$
$$\mu = \varphi(n)^{-1} \mod n$$
$$\varphi(n) = (p-1)(q-1)$$

Enkripsi

- m merupakan pesan yang akan dienkripsi dimana $m \in \mathbb{Z}_n$
- Pilih bilangan acak *r* dimana $r \in \mathbb{Z}^*_n$
- Hitung cipherteks $c = g^m \cdot r^n \mod n^2$

Dekripsi

- Cipherteks $c \in \mathbb{Z}^*_{n^2}$
- Hitung plainteks $m = L(c^\lambda \mod n^2) \cdot \mu \mod n$

C. Merkle-Hellman Knapsack

Pembangkitan Kunci

Dalam Merkle-Hellman, kuncinya adalah knapsack. Kunci publik adalah knapsack ‘keras’, dan kunci pribadi adalah knapsack 'mudah' atau superincreasing dikombinasikan dengan dua nomor tambahan, pengali dan modulus, yang digunakan untuk mengkonversi knapsack superincreasing ke knapsack keras. Angka-angka yang sama digunakan untuk mengubah jumlah dari subset dari knapsack keras ke dalam jumlah subset dari knapsack mudah, yang dipecahkan dalam waktu polinomial.

Enkripsi

Untuk mengenkripsi pesan, sebuah subset dari knapsack keras dipilih dengan membandingkannya dengan suatu set bit (plaintext), sama panjang dengan kunci, dan membuat setiap istilah kunci publik yang berkaitan dengan suatu 1 dalam sebuah plaintext elemen subset, sementara mengabaikan ketentuan terkait ke 0 istilah dalam plaintext. Unsur-unsur dari subset ini ditambahkan bersama-sama, dan jumlah yang dihasilkan adalah ciphertext.

Dekripsi

Dekripsi mungkin karena multiplier dan modulus digunakan untuk mengubah ransel, mudah superincreasing ke kunci publik juga dapat digunakan untuk mengubah nomor mewakili ciphertext ke dalam jumlah elemen yang sesuai dari ransel superincreasing. Kemudian, dengan algoritma serakah sederhana, ransel mudah bisa diatasi dengan menggunakan O (n) operasi aritmatika, yang mendekripsi pesan.

D. El Gamal

Pembangkitan Kunci

- Misalkan p adalah bilangan prima dan g adalah generator dari Z_p.
- Kunci privat adalah bilangan antara 1 dan p – 2.
- Dengan $y = g^Z \bmod p$.
- Kunci publik dari algoritma el gamal merupakan sebuah triplet (p,g,y)

Enkripsi

- Plainteks M hendak dienkripsi dan pilih bilangan acak k yang relative prima terhadap p-1, lalu hitung

$$\begin{aligned} a &\leftarrow g^k \bmod p \\ b &\leftarrow My^k \bmod p \end{aligned} \quad \text{(El Gamal encryption).} \tag{1}$$

- Cipherteks C mengandung pasangan (a,b)

Dekripsi

- Cipherteks C = (a,b).

$$M \leftarrow b/a^z \bmod p \quad \text{(El Gamal decryption).} \tag{2}$$

E. Cramer-Shoup

Pembangkitan Kunci

- A membangkitkan deskripsi efisien dari cyclic group G dengan order q dengan dua bilangan acak yang berbeda g1 dan g2
- A memilih nilai acak(x_1,x_2,y_1,y_2,z) dari {0,…,q-1}
- Hitung $c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}, h = g_1^z$.
- A mengumumkan (c,d,h) bersama dengan G,q,g_1,g_2 sebagai kunci publik dan menjaga (x_1,x_2,y_1,y_2,z) sebagai kunci privat. Group dapat dibagikan antara anggota dari tim.

Enkripsi

- m adalah pesan yang akan dikirim dan B menjadikan m elemen dari G
- B memilih k secara acak dari {0,…,q-1}

$$\begin{aligned} u_1 &= g_1^k, u_2 = g_2^k \\ e &= h^k m \\ \alpha &= H(u_1, u_2, e), \\ v &= c^k d^{k\alpha} \end{aligned} \tag{3}$$

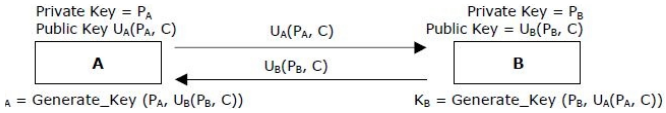
  Dimana H() adalah fungsi hash collition resistant
- B mengirimkan cipherteks (u_1,u_2,e,v) ke A

Dekripsi

- A mengitung $\alpha = H(u_1, u_2, e)$ dan memastikan kalau $u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^\alpha = v$.
- Plainteks dapat diperoleh $m = e/(u_1^z)$.

V. ALGORITMA PERSETUJUAN KUNCI

Algoritma persetujuan kunci adalah suatu cara untuk membangkitkan kunci yang digunakan dalam kirim-mengirim pesan dalam sebuat jaringan. Jadi apabila menggunakan algoritma persetujuan kunci ini maka hanya orang yang bersangkutan dan sudah sepakat untuk saling berkirim pesan rahasia. Jadi metode ini melibatkan orang – orang yang terlibat dalam jaringan dan membuat sebuah rahasia bersama tanpa perlu saling bertukar data rahasia yang sebenarnya. Dalam metode ini setiap anggota jaringan yang terlibat akan membuat sebuah rahasia bersama cukup hanya dengan berbagi kunci publik masing-masing saja, metode ini dapat digunakan untuk dua orang yang ingin saling bertukar pesan dalam sebuah jaringan, mereka saling bertukar kunci publik satu sama lain dan dengan menggunakan algoritma pembangkitan kunci bersama dengan kunci privat masing-masing untuk memperoleh rahasia bersama.



Gambar 4. Contoh algoritma persetujuan kunci

P adalah kunci privat untuk masing-masing orang dalam jaringan dan U(P, C) adalah kunci publik. Karena kunci publik yang dihasilkan dengan menggunakan kunci privat, representasi U (P, C) menunjukkan bahwa kunci publik mengandung komponen-komponen kunci pribadi P dan konstanta C, dimana C diketahui oleh semua orang yang mengambil bagian dalam komunikasi.

Beradsarkan gambar 4 di atas ada dua orang yaitu A dan B yang hendak saling berkomunikasi. P_A dan $U_A(P_A, C)$ merupakan kunci privat dan kunci publik milik A serta P_B dan $U_B(P_B, C)$ merupakan kunci privat dan kunci publik milik B. A dan be saling bertukar kunci publik mereka. Setelah A memperoleh kunci publik B, A menggunakan kunci privatnya untuk mengkalkulasi rahasia bersama K_A=Generate_Key(P_A, $U_B(P_B, C)$). Demikian pula dengan B menggunakan kunci publik A dan kunci privatnta sendiri untuk mengkalkulasi rahasia bersama K_B=Generate_Key(P_B, $U_A(P_A, C)$).

Fungsi Generate_Key yang digunakan tersebut akan menghasilkan rahasia bersama yang sama baik untuk A maupun B yaitu K_A=K_B=K(P_A, P_B, C). Karena tidak mungkin bagi orang lain untuk mengetahui kunci privat dari kunci publik yang dipertukarkan maka orang lain tidak akan bisa untuk mendapatkan rahasia bersama K. Berikut akan dijelaskan beberapa algoritma persetujuan kunci :

A. RSA Persetujuan Kunci

Proses pengenkripsian pada RSA persetujuan kunci sama dengan proses enkripsi pada algoritma RSA biasa. Hanya kuncinya yang berbeda yaitu :

Key agreement algorithm

Untuk membuat rahasia bersama antara A dan B

- Bangkitkan bilangan acak dan kunci pada A.
- Enkripsi kunci dengan algtoritma enkripsi RSA menggunakan kunci publik B dan kirimkan cipherteks ke B.
- Di B dekripsi cipherteks menggunakan kunci privat B untuk mendapatkan kuncinya

B. Persetujuan Kunci Diffie-Hellman

Diffie–Hellman (D–H) adalah algoritma persetujuan kunci yang membantu dua orang / perangkat untuk menyetujui rahasia bersama di antara mereka tanpa perlu bertukar rahasia atau informasi privat apapun untuk bersama-sama membentuk kunci rahasia bersama melalui saluran komunikasi yang belum tentu aman. Kunci ini dapat digunakan untuk mengenkripsi pesan dengan kriptografi kunci simetris biasa.

Algoritma Persetujuan Kunci

Untuk membentuk rahasia bersama antara dua perangkat A dan B, kedua perangkat setuju akan sebuah konstanta publik p dan g. dimana p adalah bilangan prima dan g adalah generator yang kurang dari p. Berikut adalah langkah-langkah pembentukan kunci rahasia bersama :

- Dimisalkan a dan b adalah kunci privat dari A dan B, kunci privat adalah bilangan acak yang lebihj kecil dari p.
- Dimisalkan g^amod p dan g^bmod p adalah kunci publik dari A dan B.
- A dan B saling bertukar kunci publiknya.
- A menghitung (g^bmod p)a mod p yang sama dengan g^{ba}mod p.
- B menghitung (g^amod p)b mod yang sama dengan g^{ab}mod p.
- Karena K = g^{ba}mod p=g^{ab}mod p, maka diperoleh kunci rahasia bersama K.

VI. KRIPTOGRAFI ELLIPTIC-CURVE

Kriptografi kurva eliptik (ECC) Teknologi relatif baru dibandingkan dengan kriptografi kunci publik seperti RSA. tombol prosesor aritmatika beroperasi pada ukuran tombol lebih kecil. Sebuah kunci 160-bit ECC dianggap sebagai dijamin sebagai kunci 1024 bit dalam RSA. ECC beroperasi pada titik-titik dalam kurva eliptik y2 = x3 + ax + b, di mana 4a3 27 b2 $\neq 0$.

Persamaan di atas kurva eliptik dalam koordinat nyata. Untuk membuat kurva eliptik operasi yang efisien dan akurat kurva eliptik dapat didefinisikan dalam bidang terbatas. kurva prosesor aritmatika dalam dua bidang yang terbatas, lapangan utama dan lapangan biner, yang didefinisikan oleh standar. Dalam operasi perdana lapangan persamaan kurva eliptik adalah diubah sebagai p y2mod = x3 + ax + b p mod, di mana 27 b2 4a3 mod p$\neq$ 0.

Domain parameter merupakan konstanta publik tertentu yang dibagi antara pihak yang terlibat dalam komunikasi yang aman dan terpercaya ECC. Ini termasuk parameter kurva a, b, titik generator G dalam kurva dipilih, p modulus, urutan n kurva dan kofaktor h.

Titik perkalian adalah pusat operasi di ECC. Dalam perkalian titik titik P pada kurva eliptik dikalikan dengan skalar k menggunakan persamaan kurva eliptik untuk mendapatkan titik lain Q pada kurva eliptik yang sama. dgn kata lain k * P = Q. Titik multiplikasi dicapai oleh dua operasi dasar kurva eliptik :

- Penambahan titik, J menambahkan dua poin dan K menggunakan persamaan kurva eliptik untuk mendapatkan titik lain yaitu L, L = J + K.
- Penggandaan titik, menambahkan titik J kepada dirinya sendiri menggunakan persamaan kurva eliptik untuk mendapatkan titik L lain dengan kata lain L = 2J.

Berikut ini adalah contoh sederhana dari perkalian titik. Misalkan P sebuah titik pada kurva eliptik. Mari k menjadi skalar yang dikalikan dengan titik P untuk mendapatkan titik lain pada kurva dgn kata lain Q = k * P. Jika k = 23 maka k * P = 23 * P = 2 (2 (2 (2P) + P) + P) + P. Dalam penjelasan ECC diberikan di bawah huruf atas menunjukkan sebuah titik dalam kurva eliptik dan huruf kecil menunjukkan skalar.

Fungsi satu arah dari ECC, keamanan ECC tergantung pada kesulitan prosesor aritmatika Masalah Curve Logaritma Diskrit. Misalkan P dan Q menjadi dua titik pada kurva eliptik sehingga k * P = Q, di mana k adalah skalar. Q dapat dengan mudah diperoleh dari P dan k tetapi mengingat P dan Q, maka komputasi layak untuk mendapatkan k, jika k cukup besar. k adalah logaritma diskrit Q untuk dasar P.

ECDH-Elliptic Curve Diffie-Hellman

Untuk membangkitkan rahasia bersama antara A dan B menggunakan ECDH, keduanya harus setuju dengan domain parameter dari Elliptic CurveBerikut adalah langkah-langkah dari ECDH:

Key Agreement Algorithm

Untuk membangkitkan rahasia bersama antara dua perangkat A dan B

- Misalkan d_A and d_B adalah kunci privat dari perangkan A dan B, kunci privat adalah bilangan acak yang kurang dari n dimana n adalah parameter domain.
- Misalkan $Q_A = d_A*G$ dan $Q_B = d_B*G$ adalah kunci publik untuk perangkat A dan B, G adalah domain parameter
- A dan B bertukar kunci publik.
- A menghitung $K = (x_K, y_K) = d_A*Q_B$
- B menghitung $L = (x_L, y_L) = d_B*Q_A$
- Karena K=L, rahasia bersama yaitu x_K

VII. PEMBANGKITAN BILANGAN ACAK

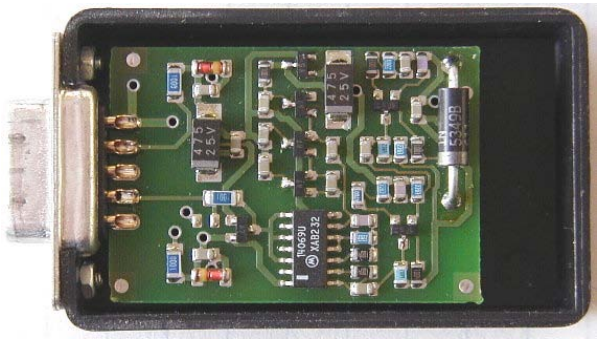
Dalam algoritma kunci publik pada umumnya setiap algoritma membutuhkan sebuah bilangan acak baik itu bilangan prima maupun tidak. Semakin beragam dan panjang suatu bilangan acak yang dibangkitkan akan mempengaruhi algoritma kunci publik yaitu algoritma kunci publik akan menjadi semakin sulit untuk dipecahkan, maka dari itu kita akan melihat beberapa algoritma untuk membangkitkan bilangan acak, namun pada makalah ini tidak akan dibahas secara detail. Berikut

adalah beberapa algoritma pembangkit bilangan acak :

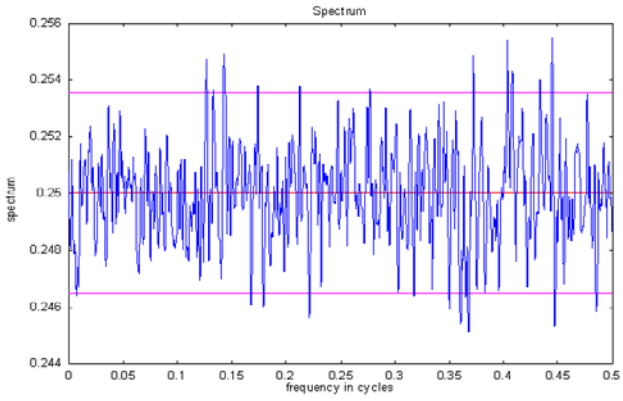
- **Blum Blum Shub.** Based on a formula on prime numbers.
- **Mersenne twister**
  Cepat dengan periode yang tinggi.
- **Lagged Fibonacci generator**
  Pengembangan dari pembangkit kongruen linear yang menggunakan urutan angka Fibonacci.
- **Linear congruential generator**
  Salah satu algoritma pembangkit bilangan acak terua tetapi bukan yang terbaik, menggunakan tiga bilangan untuk membangkitkan sebuat sequence.
- **Yarrow algorithm**
  Aman secara kriptografi pembangkit bilangan acak pseudorandom yang juga dapat digunakan sebagai pembangkit bilangan acak sesuangguhnya, menerima masukkan acak dari sumbar analog acak.
- **Fortuna**
  Sebuah pengembangan dari Yarrow algorithm.
- **Linear feedback shift register**
  Sebuah pergeseran register dimana bit masukkan adalah fungsi linear dari kondisi sebelumnya, kondisi pertama adalah benihnya.

A. Hardware Random Number Generator

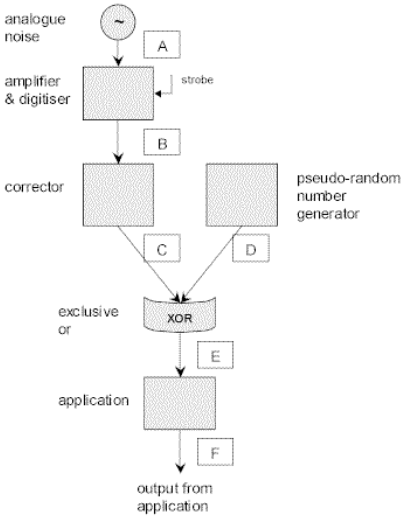
Ada pula pembangkit bilangan acak yang menggunakan hardware dan menggabungkannya dengan pseudo random number generator. Jadi perangkat keras yang digunakan berfungsi untuk menghasilkan random noise.



Gambar 5. Hardware random number generator



Gambar 6. Grafik spectrum dari noise yang dihasilkan



Gambar 7. Cara kerja hardware random number generator

VIII. ANALISIS

Berdasarkan hasil pengamatan dan penelitian mencakup berbagai algoritma kunci publik serta pembangkit bilangan acak yang sudah di bahas di atas. Maka dapat kita analisis yaitu kebanyakan algoritma kunci publik memiliki performa yang cukup buruk dalam hal waktu, contohnya yaitu algoritma kunci publik RSA yang membutuhkan waktu lama untuk mengenkripsi data dengan ukuran yang besar dan juga dengan p dan q yang besar.

Dalam hal waktu semua algoritma kunci publik baik algoritma kunci publik biasa, algoritma kunci publik persetujuan kunci dan algoritma kunci publik elliptical curve, membutuhkan waktu yang relatif lama apabila digunakan untuk mengenkripsi data dalam jumlah yang besar. Oleh karena itu biasanya algoritma kunci publik ini ;lebih banyak digunakan untuk mengenkripsi tanda tangan digital dari sebuah pesan untuk keperluan otentikasi pesan. Krena ukuran dari tanda tangan digital relatif kecil dibandingkan isi pesannya maka waktu yang dibutuhkan cukup cepat.

Tabel 1. Perbandingan performa algoritma kunci publik

Algoritma	Kecepatan	Kekuatan
RSA	★	★★★
Paillier	★	★★
Merkle-Hellman Knapsack	★★	★
El Gamal	★★	★
Cramer Shoup	★★	★
DH	★★	★★
ECDH	★★	★★

Tetapi dalam makalah ini kita membahas mengenai algoritma kunci publik untuk mengenkripsi pesan, jadi kita teteap harus berhadapan dan mencari solusi untuk mengatasi kekurangan performa dari algoritma kunci

publik ini.

Karena waktu yang dibutuhkan oleh semua algoritma kunci publik ini apabila pesan yang akan dienkripsi ukurannya besar relatif lama, maka biasanya orang jadi lebih mementingkan kepastian akan keamanan data yang akan dikirim setelah dienkripsi. Dengan waktu yang lama untuk mengenkripsinya tentu orang ingin hasil yang sebanding yaitu meskipun waktu yang dibutuhkan untuk mengenkripsi cukup lama akan tetapi bila cipherteks yang dihasilkan tidak mungkin dipecahkan maka semuanya menjadi sepadan antara waktu yang dibutuhkan dan keamanan yang dihasilkan.

Oleh karena itu kita akan fokus membahas kekuatan dari algoritma kunci publik tersebut.Letak kekuatan dari algoritma kunci publik pada umumnya yaitu pada panjangnya kunci yang digunakan (terutama pada algoritma RSA) dan belum adanya algoritma untuk memfaktorkan bilangan yang memiliki ukuran sangat besar. Selama belum ditemukannya algoritma tersebut maka keamanan dari algoritma kunci publik akan terjamin.

Dari semua algoritma kunci publik di atas dan berbagai cara untuk membangkitkan kunci dapat diambil kesimpulan bahwa algoritma RSA merupakan algoritma kunci publik yang paling aman dan paling umum digunakan serta banyak dimanfaatkan oleh organisasi-organisasi besar baik untuk mengenkripsi pesan maupun untuk otentikasi dari pesan. Hal ini disebabkan karena kekuatan dari algoritma RSA ini yang bergatung pada panjangnya bilangan prima p dan q yang digunakan untuk membangkitkan kunci publik dan kunci privat. Semakin panjang p dan q yang digunakan maka semakin kuat algoritma RSA ini. Selain itu karena sampai saat ini dan beberapa tahun ke depan tidak aka nada algoritma yang bisa memfaktorkan bilangan yang ukurannya sangat besar. Meskipun pada akhirnya telah ditemukan algoritma untuk memfaktorkan tersebut, waktu yang dibutuhkan untuk memfaktorkannya akan sangat lama sehingga membuatnya menjadi tidak mungkin dipecahkan.

Panjangnya p dan q juga bergantung pada pembangkit bilangan acak yang digunakan. Pada bagian sebelumnya telah dibahas mengenai algotritma pembangkit bilangan acak dan deskripsi singkatnya, akan tetapi karena konteks yang dibahas dalam algoritma ini yaitu pembangkit bilangan acak untuk menunjang kekuatan dari algoritma kunci publik maka pada umumnya algoritma pembangkit bilangan acak manapun yang digunakan sama saja asalkan dapat menghasilkan bilangan yang benar-benar acak baik prima atau bukan dan dengan ukuran yang panjang. Serta sekarang ini pada setiap tools untuk pemrograman sudah disediakan fungsi untuk mendapatkan bilangan acak yang dapat dimanfaatkan untuk menghasilkan p dan q.

IX. PENERAPAN IDE BARU PADA ALGORITMA RSA

Kelemahan dari algoritma RSA yaitu lamanya waktu yang dibutuhkan untuk membangkitkan bilangan acak prima yang panjang, karena semakin panjang p dan q yang digunakan maka semakin aman dan semakin baiklah hasil enkripsi dari algoritma kunci publik RSA. Selain itu operasi pemangkatan potongan bilangan ASCII dari teks yang akan dienkripsi membutuhkan waktu yang lama pula karena apabila p dan q nya panjang maka semakin besar dan pabjang bilangan hasil pemangkatan tersebut.

Oleh karena itu solusinya adalah dengan menggunakan BigInteger untuk menampung bilangan hasil pemangkatan tersebut dan menggunakan algoritma yang dapat memangkatkan kemudian melakukan operasi mod dengan efektif, efisien dan cepat. Fungsi modPow dari BigInteger yang dapat digunakan untuk memangkatkan sekaligus melakukan operasi mod dengan cepat. (source code dilampirkan di akhir laporan.

Hal ini mengatasi dua kelemahan utama dari algoritma kunci publik RSA yaitu rendahnya performa dari algoritma ini dan mudahnya algoritma ini dipecahkan bila ditemukan suatu algoritma yang dapat memfaktorkan bilangan yang sangat besar dengan cepat. Dengan memanfaatkan BigInteger dan juga fungsi modPow maka operasi enkripsi dengan menggunakan algoritma RSA ini meningkat performanya sehingga waktu yang dibutuhkan menjadi jauh lebih singkat. Sedangkan untuk kekuatan dari algoritma ini tentu saja semakin membaik karena digunakannya BigInteger maka mampu menampung p dan q dengan ukuran yang panjang serta hasil pemangkatan yang besar pula. Hal ini membuat semakin tidak mungkin untuk memfaktorkannya dan memecahkan algoritma RSA dengan kunci yang sangat panjang. Karena meskipun telah ditemukan algoritma untuk memfaktorkannya, wktu yang dibutuhkan untuk memfaktorkannya akan sangat lama dan membuatnya kembali menjadi tidak mungkin dipecahkan.

Oleh krena itu semakin menguatkan alas an mengapa algoritma kunci publik RSA merupakan algoritma kunci publik terbaik. Berikut adalah gambar interface dari program pembangkit kunci dari algoritma RSA yang menggunakan BigInteger dalam bahasa C# menggunakan Microsoft Visual Studio Team System 2008 dan kemudian kunci yang sudah dibangkitkan ini kemudian digunakan untuk mengenkripsi pesan memanfaatkan fungsi modPow dan terbukti bahwa waktu yang diperlukan menurun drastis dari sebelum digunakannya fungsi modPow tersebut.

Gambar 8. Interface program pembangkit kunci dan pengenkripsi dengan algoritma RSA

X. KESIMPULAN

Kriptografi kunci publik adalah sebuah inovasi dan merupakan bagian tak terhindarkan dari hampir semua protokol keamanan dan aplikasi. Mampu melakukan negosiasi rahasia bersama antara dua perangkat online tanpa perlu melakukan pertukaran data rahasia menciptakan sebuah terobosan dalam jaringan aman / komunikasi internet. Meskipun secara teoritis mungkin untuk menemukan rahasia berbagi dari informasi publik yang tersedia, itu akan memakan waktu lebih lama secara eksponensial sehingga praktis tidak mungkin. Menemukan pesamaan matematika untuk membalik fungsi satu arah tidaklah mungkin oleh karena itu algoritma kunci publik masih tetap hidup.

Berdasarkan pernyataan di atas mengapa kita tidak membuang algoritma kunci pribadi dan menggunakan algoritma kunci publik untuk semuanya. Masalahnya terletak pada kinerja, algoritma RSA adalah cukup cepat untuk algoritma kunci publik tetapi tetap masih berada antara 1.000 dan 10.000 kali lebih lambat dari algoritma kunci pribadi. Oleh karena itu hanya tidak mampu memindahkan data dengan ukuran yang sangat besar karena lambat. Jadi dalam praktek biasanya orang akhirnya menggunakan kombinasi teknik-teknik kunci publik dan pribadi untuk mencapai kinerja saluran komunikasi yang aman.

Selama proses pertukaran kunci kunci publik dapat melewati titik-titik antara yang berbeda. Setiap perantara sehingga dapat merusak atau mengubah kunci publik untuk kunci publik. Oleh karena itu untuk mendirikan berbagi rahasia adalah penting bahwa perangkat A menerima kunci publik yang tepat dari perangkat B dan sebaliknya. Maka dari itu biasanya dipadukan dengan tanda tangan digital untuk memastikan keaslian dari kunci-kunci tersebut.

Sejak kriptografi kunci publik melibatkan operasi matematika dengan bilangan berjumlah besar, algoritma ini adalah sangat lambat dibandingkan dengan algoritma kunci simetrik. Akan tetapi dengan pengembangan dari algoritma RSA yang diberikan pada bab sebelumnya maslah ini dapat teratasi, komunikasi menjadi lebih aman dan lebih cepat dari pada algoritma RSA dengan cara biasa.

Keamanan RSA tergantung pada kesulitan untuk memfaktorkan bilangan besar untuk mendapatkan bilangan prima p dan q. n mudah diperoleh dengan mengalikan p dan q tapi operasi kebalikan dari memfaktorkan n untuk mendapatkan p dan q adalah hampir mustahil jika p dan q cukup besar ukurannya. Selain itu meskipun ditemukan cara untuk memfaktorkannya karena p dan q berukuran sangat besar maka akan membutuhkan waktu yang sangat lama untuk berhasil menemukan p dan q sehingga membuatnya menjadi tidak mungkin difaktorkan.

IX. ACKNOWLEDGMENT

Saya James Filipus sebagai penulis dari makalah IF3058 Kriptografi pengganti UAS dengan judul **Perbandingan Antar Algoritma Untuk Mencari Algoritma Kriptografi Kunci-Publik Dan Pembangkit Bilangan Acak Terbaik** ingin berterima kasi pertama-tama kepada Tuhan Yang Maha Esa, yang saya percayai telah membimbing saya selama penulisan makalah ini sampai selesai. Dan juga saya ingin menyampaikan rasa hormat dan terima kasih saya kepada dosen saya yaitu Pak Rinaldi Munir, Beliau telah membimbing saya mulai dari mata kuliah Struktur Diskrit, Strategi Algoritmik dan Kriptografi serta membuat saya paham akan materi yang diberikan. Dan sekarang ini sampai terselesaikannya makalah ini karena bekal ilmu yang Beliau berikan. Semoga makalah ini dapat membawa manfaat tidak hanya bagi saya tetapi bagi orang lain yang membaca. Tidak lupa saya juga berterima kasih kepada kedua orang tua saya yang selalu membimbinmg dan juga teman0teman yang memberikan masukkan dan indpirasi untuk makalah ini. Terima kasih.

REFERENCES

[1] Ir. Rinaldi Munir, M.T., *Diktat Kuliah IF3058 Kriptografi*, Departemen Teknik Informatika ITB, 2005, halaman 139-151.
[2] N. Ferguson; B. Schneier (2003). *Practical Cryptography*.
[3] J. Katz; Y. Lindell (2007). *Introduction to Modern Cryptography*.
[4] A. J. Menezes; P. C. van Oorschot; S. A. Vanstone (1997). *Handbook of Applied Cryptography*.
[5] www.daimi.au.dk/~ivan/Gen**Paillier**_finaljour.ps
[6] msdn.microsoft.com/.../system.security.cryptography.**randomnumbergenerator**.aspx
[7] Haahr, Mads. *Introduction to Randomness and Random Numbers*

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, 29 April 2010

James Filipus – 13507087

Berikut adalah lampiran source code untuk fungsi modPow BigInteger yang digunakan dalam kalkulasi pada program untuk menghasilkan cipherteks menggunkan algoritma RSA

```
public BigInteger modPow(BigInteger exp, BigInteger n)
        {
                if((exp.data[maxLength-1] & 0x80000000) != 0)
                        throw (new ArithmeticException("Positive exponents
only."));

                BigInteger resultNum = 1;
              BigInteger tempNum;
              bool thisNegative = false;

              if((this.data[maxLength-1] & 0x80000000) != 0)    // negative this
              {
                      tempNum = -this % n;
                      thisNegative = true;
              }
              else
                      tempNum = this % n;   // ensures (tempNum * tempNum) < b^(2k)

              if((n.data[maxLength-1] & 0x80000000) != 0)    // negative n
                      n = -n;

                // calculate constant = b^(2k) / m
                BigInteger constant = new BigInteger();

                int i = n.dataLength << 1;
                constant.data[i] = 0x00000001;
                constant.dataLength = i + 1;

                constant = constant / n;
                int totalBits = exp.bitCount();
                int count = 0;

                // perform squaring and multiply exponentiation
                for(int pos = 0; pos < exp.dataLength; pos++)
                {
                        uint mask = 0x01;
                        //Console.WriteLine("pos = " + pos);

                        for(int index = 0; index < 32; index++)
                        {
                                if((exp.data[pos] & mask) != 0)
                                        resultNum = BarrettReduction(resultNum *
tempNum, n, constant);

                                mask <<= 1;

                                tempNum = BarrettReduction(tempNum * tempNum, n,
constant);


                                if(tempNum.dataLength == 1 && tempNum.data[0] ==
1)
                                {
                                        if(thisNegative && (exp.data[0] & 0x1)
!= 0)    //odd exp
                                                return -resultNum;
                                        return resultNum;
                                }
                                count++;
```

```
                                if(count == totalBits)
                                        break;
                        }
                }

                if(thisNegative && (exp.data[0] & 0x1) != 0)    //odd exp
                        return -resultNum;

        return resultNum;
}
```