Enkripsi SMS menggunakan ECC

Eko Mardianto, Isbat Uzzin, **Yuliana Setiowati**
Politeknik Elektronika Negeri Surabaya
Institut Teknologi Sepuluh Nopember
Kampus PENS-ITS Keputih Sukolilo Surabaya 60111
Telp (+62)31-5947280, 5946114, Fax. (+62)31-5946114
Email :cyber_code8388@yahoo.com

Abstrak

ECC (Elliptic Curve Cryptography) merupakan suatu metode enkripsi alternatif yang dapat digunakan untuk menjaga kerahasiaan layanan SMS. Metode ini memungkinkan terjadinya enkripsi pesan dari telepon seluler pengirim ke telepon seluler penerima yang dituju. Dengan memanfaatkan metode ini, layanan SMS dapat menjadi lebih aman karena algoritma enkripsi yang digunakan cukup tangguh dan kunci yang dipakai sukar dipecahkan. Algoritma ini merupakan algoritma kunci publik, akan lebih aman jika dibandingkan dengan algoritma kunci private. Pada Aplikasi ini, sistem menggabungkan antara ECC dengan algoritma kunci private, supaya mendapatkan hasil yang lebih maksimal. Selain itu pada aplikasi ini juga dilengkapi dengan kompresi untuk mengurangi pembengkakan jumlah karakter hasil enkripsi. Metode Kompresi yang digunakan adalah metode Huffman.

Kata Kunci : Kriptografi, ECC, Elliptic curve, ECIES, J2ME, Kompresi Huffman, SMS.

1. Pendahuluan

1.1 Latar Belakang

Saat Telepon selular merupakan alat komunikasi yang sudah dipakai oleh sebagian besar orang di dunia. Telepon selular menyediakan media komunikasi yang beragam dan salah satu diantaranya adalah media SMS (*Short Message Service*). SMS atau Short Message Service merupakan suatu layanan pengiriman pesan singkat melalui telepon genggam. Walaupun merupakan bagian dari kemampuan standard GSM fase pertama, SMS masih merupakan layanan yang banyak digunakan oleh masyarakat. Bahkan, berdasarkan survei yang dilakukan oleh Nielsen Mobile di Amerika pada kuartal 2 tahun 2008, pelanggan telepon seluler di Amerika Serikat lebih banyak menggunakan SMS dibanding melakukan percakapan telepon [REA08]. Berbagai kemudahan yang ditawarkan oleh SMS antara lain informasi sesuai permintaan, pengunduhan nada dering, sampai dengan transaksi perbankan atau mobile banking. Secara umum SMS tidak menjamin kerahasiaan dan keutuhan pesan yang dikirimkan oleh pengguna. Oleh karena pesan-pesan teks yang dikirim pengguna terkadang merupakan pesan yang rahasia dan pribadi, sehingga kerahasiaan pesan menjadi sangat penting untuk dijaga dari orang-orang yang tidak berhak mendapatkannya. Sehingga dibutuhkan suatu sistem keamanan dalam menyampaikan pesan tersebut.

Aplikasi enkripsi dan dekripsi pesan dapat meningkatkan tingkat keamanan pada layanan yang memerlukan kerahasiaan pesan. Hal ini dapat mengurangi bocornya informasi kepada pihak-pihak yang tidak berkepentingan seperti operator telepon seluler. Untuk meningkatkan keamanan dalam aplikasi enkripsi dan dekripsi pesan SMS, perlu digunakan algoritma yang handal.

ECC (Elliptic Curve Cryptography) merupakan suatu metode enkripsi alternatif yang dapat digunakan untuk menjaga kerahasiaan layanan SMS. Metode ini memungkinkan terjadinya enkripsi pesan dari telepon seluler pengirim ke telepon seluler penerima yang dituju. Dengan memanfaatkan metode ini, layanan SMS dapat menjadi lebih aman karena algoritma enkripsi yang digunakan cukup tangguh dan kunci yang dipakai sukar dipecahkan.

1.2 Rumusan Permasalahan

Berdasarkan uraian tersebut diatas, dalam pengerjaan proyek akhir ini timbul beberapa masalah diantaranya adalah :

1. Bagaimana mengimplementasikan teknologi enkripsi dan dekripsi pesan sms pada mobile device dengan menggunakan algoritma ECC(Elliptic Curve Cryptography).

2. Bagaimana menguji tingkat keamanan enkripsi pesan dengan menggunakan algoritma ECC(Elliptic Curve Cryptography).

1.3 Batasan Masalah

Pada penyelenggaraan proyek akhir ini, batasan permasalahannya adalah :

1. Perangkat lunak tidak dapat melakukan akses ke memory di dalam kartu SIM.
2. Perangkat lunak yang dibangun hanya dapat dijalankan pada telepon selular yang dapat mendukung aplikasi berbasis java.
3. Perangkat lunak yang dibangun hanya bisa dijalankan pada ponsel yang memiliki sistem operasi yang mengijinkan perangkat lunak secara penuh untuk mengakses pesan yang ada pada ponsel.

1.4 Tujuan Proyek

Tujuan dari proyek akhir ini adalah membangun sebuah sistem baru yang terintegrasi dengan telpon seluler yang mampu menjaga keamanan dan kenyamanan pemakai telpon seluler dalam menggunakan jasa layanan short message service (sms).

2. Teori Penunjang

2.1 Algorima ECC (Elliptic Curve Cryptography)

Kriptografi kurva eliptik termasuk kedalam sistem kriptografi kunci publik yang mendasarkan keamanannya pada permasalahan matematis kurva eliptik. Tidak seperti permasalahan matematis logaritma diskrit (*Discrete Logarithm Problem*, DLP) dan pemfaktoran bilangan bulat (*Integer Factorization Problem*, IFP), tidak ada algoritma waktu subeksponensial yang diketahui untuk memecahkan permasalahan matematis logaritma diskrit kurva eliptik (*Elliptic Curve Discrete Logarith Problem*, ECDLP). Karena alasan tersebut, algoritma kriptografi kurva eliptik mempunyai keuntungan jika dibandingkan dengan algoritma kriptografi kunci publik lainnya yaitu dalam hal ukuran panjang kunci yang lebih pendek tetapi memiliki tingkat keamanan yang sama. Ada tiga protokol ECDLP yang diketahui saat ini yaitu *Elliptic Curve Digital Signature Algorithm* (ECDSA), *Elliptic Curve Diffie Hellman* (ECDH), dan *Elliptic Curve ElGamal* (ECElgamal). *Elliptic Curve Cryptography* (ECC) adalah salah satu pendekatan algoritma kriptografi kunci publik berdasarkan pada struktur aljabar dari kurva ellips pada daerah finite.

Penggunaan kurva ellips dalam kriptografi dicetuskan oleh Neal Koblitz dan Victor S. Miller pada tahun 1985. Kurva ellips juga digunakan pada beberapa algoritma pemfaktoran integer yang juga memiliki aplikasinya dalam kriptografi, seperti *Lenstra Elliptic Curve Factorization.* Algoritma kunci publik berdasarkan pada variasi perhitungan matematis yang terbilang sangat sulit dipecahkan tanpa pengetahuan tertentu mengenai bagaimana perhitungan tersebut dibuat. Pembuat algoritma menyimpan kunci privat dan menyebarkan kunci publiknya. Algoritma kunci publik digunakan untuk mengenkripsi pesan dimana hanya pembuat algoritma yang dapat memecahkannya. Sistem kunci publik awal, seperti algoritma RSA, menggunakan dua bilangan prima yang sangat besar: pengguna memilih dua bilangan prima random yang besar sebagai kunci privatnya, dan mempublikasikan hasil dari perhitungannya sebagai kunci publik. Pemfaktoran bilangan-bilangan besar yang sangat sulit dapat menjaga kerahasiaan kunci privat dari publik. Persoalan lain menyangkut perhitungan aljabar ab = c, dimana a dan c diketahui. Perhitungan semacam itu menyangkut bilangan kompleks atau real yang dapat dengan mudah dipecahkan menggunakan logaritma. Tetapi dalam kumpulan bilangan finite yang besar, menemukan solusi untuk perhitungan semacam itu sangat sulit dan dikenal sebagai "*discrete logarithm problem*". Kurva ellips dapat ditulis dengan perhitungan matematis sebagai berikut:

$$y^2 = x^3 + ax + b \qquad (1)$$

Kumpulan titik pada kurva dapat membentuk kumpulan abelian (dengan titik pada tak terhingga sebagai elemen identitas). Jika nilai x dan y dipilih dari daerah finite yang besar, solusi akan membentuk suatu kumpulan abelian finite. Permasalahan logaritma diskrit pada kumpulan kurva ellips tersebut dipercaya lebih sulit dibandingkan permasalahan yang sama (perkalian bilangan tidak nol) dalam daerah finite. Selain itu, kunci dalam algoritma kriptografi kurva ellips dapat dipilih lebih pendek panjangnya untuk keamanan yang cukup tinggi.

Sebagai salah satu kriptosistem kunci publik, belum ada pembuktian matematis untuk tingkat kesulitan dari ECC yang telah dipublikasikan sampai tahun 2006. Tetapi *U.S. National Security Agency* telah mengesahkan teknologi ECC sebagai algoritma kriptografi yang dianjurkan.

2.1.1 Aturan Penjumlahan Titik pada Kurva Elips

Untuk membentuk elliptic curve cryptoystem (ECC) diperlukan aturan penjumlahan dua titik pada kurva elips E(Fp) yang menghasilkan titik ke tiga pada kurva elips. Aturan ini dapat dijelaskan secara geometris sebagai berikut :

1. P + O = O + P = P untuk setiap $P \in E(\mathrm{F}_p)$.
2. Jika $P = (x,y) \in E(\mathrm{F}_p)$ maka (x,y) + (x,-y) = O. (Titik (x,-y) dinyatakan dengan –P, dan disebut negatif P. Tentunya –P merupakan sebuah titik dalam kurva).

3. Diberikan $P = (x_1,y_1) \in E(\mathrm{F_p})$ dan $Q = (x_2,y_2) \in E(\mathrm{F_p})$, dimana P = -Q. Maka P+Q = (x3,y3) diperoleh dengan mengambil garis L yang melewati titik P dan Q atau garis singgung L untuk P=Q. Misalkan garis L :

$y = \lambda x + \beta$ di mana :

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{untuk } P=Q \\ \dfrac{3x_1^2 + a}{2y_1} & \text{untuk } P \neq Q \end{cases} \quad (2)$$

maka diperoleh (x3,y3) sebagai berikut :

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 + x_3) - y_1 \quad (3)$$

Tidak seperti kriptografi kunci publik konvensional yang didasari oleh aritmatika pada bidang yang terbatas (*finite field*), ECC beroperasi pada kumpulan poin pada sebuah kurva elips yang didefinisikan diatas sebuah bidang terbatas. Operasi kriptografi utamanya adalah *scalar point multiplication*, yang akan menghitung $Q = k\ P$ (sebuah titik P dikalikan dengan suatu bilangan bulat k menghasilkan sebuah titik Q pada kurva). Perkalian skalar dilakukan lewat sebuah kombinasi dari penambahan titik dan *doubling* titik. Sebagai contoh, $11P$ dapat dijabarkan menjadi $11P = (2((2(2P)) + P)) + P$.

2.4. KODE HUFFMAN

2.4.1. Sejarah Kode Huffman

Kode Huffman menggunakan tabel dengan variasi kode panjang untuk melakukan *encoding* dari sebuah simbol. Tabel variabel kode panjang tersebut telah dibuat terlebih dahulu secara terpisah berdasarkan nilai kekerapan munculnya suatu simbol. Metode ini ditemukan oleh David A. Huffman ketika ia melakukan studi Ph.D di MIT. Kode ini dipublikasikan pada tahun 1952 pada tulisannya yang berjudul "*A Method for the Constuction of Minimim-Redudancy Codes*".

2.4.2. Pembentukan Kode Huffman

Kode Huffman menggunakan metode spesifik untuk merepresentasikan setiap simbol yang menghasilkan suatu kode prefix. Kode prefix ini merupakan sekumpulan kode biner yang pada kode ini tidak mungkin terdapat kode prefix yang menjadi awalan bagi kode biner yang merepresentasikan simbol lain. Hal ini akan mencegah timbulnya keraguan dalam proses *decoding.* Dalam kode Huffman, kode biner untuk simbol dengan kekerapan lebih besar akan memiliki kode yang lebih pendek daripada untuk symbol dengan kekerapan lebih kecil. Membentuk suatu kode Huffman dimulai dengan membuat suatu pohon biner yang disebut pohon Huffman. Pohon ini akan disimpan pada suatu tabel, dengan ukuran yang bergantung pada jumlah dari simbol tersebut. Suatu simpul pada pohon biner dapat berupa simpul daun (simpul yang memiliki jumlah anak nol) ataupun simpul dalam (simpul yang mempunyai anak). Pada awalnya, semua simpul merupakan simpul daun, yang mengandung simbol itu sendiri serta bobotnya (frekuensi kekerapan) dari simbol teresbut dan bisa juga mengandung *link* ke simpul orangtua yang akan memudahkan pembacaan kode (secara terbalik) dimulai dari simpul daun. Pada simbul dalam terdapat bobot dan *link* ke dua simpul anak dan bisa ke simpul orangtua. Sebagai perjanjian, bit '0' akan merepresentasikan anak kiri dan bit '1' akan merepresentasikan anak kanan. Pohon yang telah selesai akan memiliki *n* buah simpul daun dan *n-1* buah simpul dalam . Sutu pohon Huffman dapat dibentuk dengan cara sebagai berikut :

1. Membuat simpul daun sebanyak sejumlah simbol
2. Memilih dua simbol dengan peluang terkecil dan dikombinasikan sebagai suatu simpul orangtua
3. Membuat simpul yang merupakan simpul orangtua dari dua simpul dengan peluang terkecil
4. Memilih sebuah simpul berikutnya (termasuk simpul baru) yang memiliki peluang terkecil
5. Melakukan prosedur yang sama pada dua symbol berikutnya yang memilki peluang terkecil

Tabel 1. Kode ASCII

Simbol	Kode ASCII
A	1000001
B	1000010
C	1000011
D	1000100

Maka, rangkaian bit untuk string '*ABACCDA*' :

01000001010000100100000101000011010000110
100010001000001

Berdasarkan metode pengkodean ASCII, setiap huruf direpresentasikan dalam 8-bit sehingga untuk merepresentasikan 7 huruf akan membutuhkan 7 × 8 = 56-bit. Berikut ini merupakan contoh pengkodean Huffman untuk melakukan kompresi *string.* Tabel 2 akan memperlihatkan tabel kekerapan dan kode Huffman untuk *string* '*ABACCDA*'.

Tabel 2. Kekerapan dan kode Huffman *string* '*ABACCDA*'

Simbol	Kekerapan	Peluang	Kode Huffman
A	3	3/7	0
B	1	1/7	110
C	2	2/7	10
D	1	1/7	111

Sehingga, dengan menggunakan kode Huffman rangkaian bit untuk string

'*ABACCDA*' : 0110010101110

2.4.3. *Encoding*

Proses untuk melakukan pembentukan kode dari suatu data tertentu disebut *encoding*. Dalam hal ini, kode Huffman akan tebentuk sebagai suatu kode biner. Kode Huffman didapatkan dengan membaca setiap kode dari daun simbol tersebut hingga ke akarnya. Ketika suatu kode Huffman telah dibentuk, suatu data dapat akan mudah di *encode* dengan mengganti setiap simbol menggunakan kode yang telah dibentuk.

2.4.4. *Decoding*

Decoding merupakan proses yang mengembalikan suatu data dari suatu kode tertentu. Proses *decoding* ini merupakan kebalikan dari proses *encoding*. Terdapat dua cara yang cukup cepat untuk melakukan *decoding* simbol :

1. Membaca dari pohon Huffman

   Hal ini dapat dilakukan dengan membaca sebuah bit dari kode binernya dan menelusuri hingga sampai pada simpul daun yang mengandung simbol tersebut untuk setiap bitnya. Ketika suatu bit sampai pada daun suatu pohon, suatu simbol yang terkandung dalam daun tersebut ditulis untuk *decoded* data tersebut dan mengulanginya kembali dari akar pohon terebut.

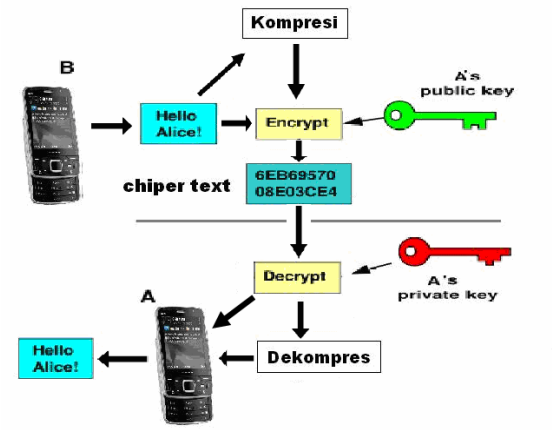
2. Menggunakan tabel kode Huffman

   Decoding cara ini dilakukan dengan menyimpan setiap kode pada suatu tabel yang terurut berdasarkan panjang kode dan mencari kesamaan dari setiap bit yang dibaca. Metode membaca dari pohon Huffman lebih cepat dilakukan untuk menangani kasus terburuk *encoding*, yaitu ketika seluruh simbolnya memiliki panjang 8-bit. 8-bit kode akan menunjuk pada simbol dengan kedalaman 8, tetapi pencarian kode biner untuk 256 simbol adalah O(log2(256)) atau sekitar 16 langkah.

2.4.5. Kompleksitas Algoritma Huffman

Dalam melakukan satu kali proses iterasi untuk menggabungkan dua buah pohon dengan frekuensi terkecil ada sebuah akar, waktu yang dibutuhkan adalah O(log *n*). Proses tersebut akan dilakukan sebanyak *n* kali hingga terbentuk sebuah pohon Huffman. Sehingga, waktu untuk kompleksitas waktu algoritma Huffman adalah O(*n* log *n*).

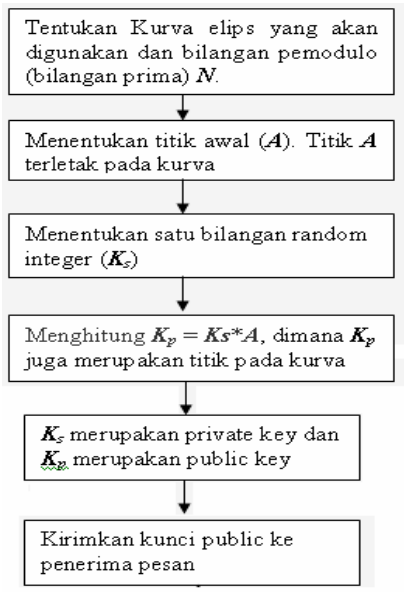
3. Rancangan Sistem



Gambar 1. Rancangan Sistem

3.1 Pembuatan Kunci

Tapan ini merupakan tahapan yang paling penting, proses pembuatan kunci akan dijelaskan pada gambar 2.



Gambar 2. *Blok diagram Pembuatan Kunci*

Keterangan :

1. Pengirim dan penerima harus menyepakati sebuah kurva elips yang nantinya akan digunakan untuk pembuatan key. Selain itu kedua pihak juga harus menyepakati sebuah bilangan prima sebagai pemodulo. Persamaan kurva elips yang disepekati memiliki bentuk sebagai berikut:

   $Y^2 = X^3 + ax + b$; untuk a dan b adalah konstanta.

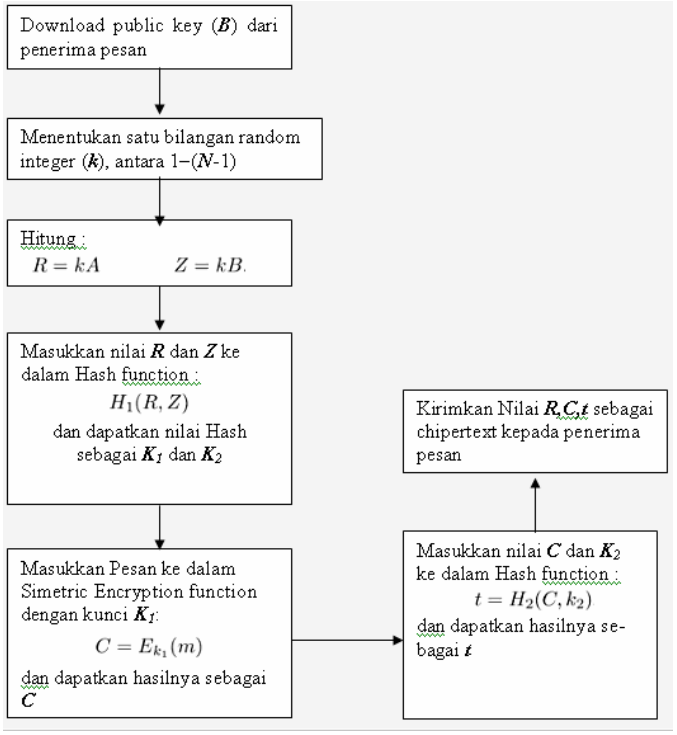
2. Setelah memperoleh persamaan kurva dan bilangan pemodulo, maka sistem akan meng-generate titik yang merupakan anggota dari kurva.
3. Kemudian kedua belah pihak juga menentukan sebuah titik awal yang nantinya akan digunakan untuk menghitung kunci publik dari masing-masing pihak. Titik awal merupakan anggota dari titik-titik yang dihasilkan pada step 2.
4. Masing-masing pihak memilih suatu bilangan integer yang nantinya akan digunakan sebagai kunci private. Bilangan integer yang dipilih harus lebih kecil dari bilangan pemodulo.
5. Menghitung kunci publik dengan cara mengalikan bilangan integer (yang diperoleh pada step 4) dengan titik awal (yang diperoleh pada step 3).
6. Kirimkan public key ke calon penerima pesan.

Pada tahap 1-3 akan digenerate sistem secara otomatis, untuk mengurangi tingkat kesalahan user.

3.*2* Enkripsi Pesan

Proses Enkripsi yang ada pada modul ini memiliki proses yang bertahap, yaitu meng-generate key terlebih dahulu untuk proses enkripsi selanjutnya kemudian baru mengenkripsi pesan dengan kunci yang telah diperoleh. Seteleh itu chipertext yang dihasilkan akan dihashing, hasil dari hashing tersebut digunakan untuk memeriksa apakah chipertext yang dikirimkan akan diterima sebagai mana mestinya (autentikasi/ digital signature). Proses hashing dilakukan sebanyak 2 kali, yaitu sebelum dan sesudah proses enkripsi pesan. Proses enkripsi pesan akan dijelaskan pada gambar 3.

Pada gambar 3 terdapat atribut R, C, t yang akan dikirimkan sebagai chipertext namun supaya pembengkakan pesan hasil enkripsi tidak terlalu besar maka atribut R dan t tidak ikut dikirimkan bersamaan dengan pesan. Pada tahap pembuatan key, nilai-nilai yang berkaitan dengan kurva telah dilakukan oleh sistem dan disimpan pada aplikasi yang tertanam pada mobile phone masing-masing sehingga atribut R tidak perlu dikirimkan, dan untuk mengurangi pembengkakan karakter hasil enkripsi yang akan dikirimkan maka variabel t juga tidak dikirimkan, karena jika variabel t dikirimkan maka pada karakter hasil enkripsi harus bertambah minimal 20 karakter.



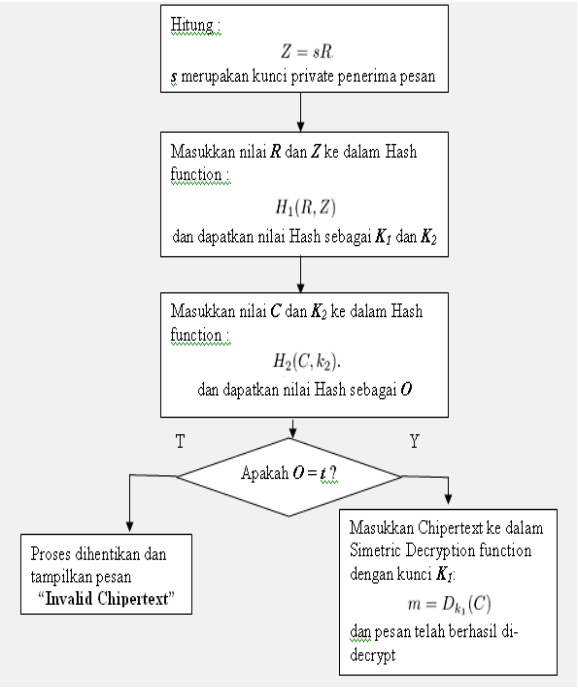
Gambar 3. *Diagram Enkripsi Pesan*

Keterangan :

1. Mendownload kunci publik dari calon penerima pesan.
2. Kemudian memilih sebuah bilangan integer yang lebih kecil dari pemodulo (yang didapat pada proses pembuatan key).
3. Menghitung nilai ***R*** dan ***Z*** yang nantinya akan digunakan untuk meng-generate key.
   R = kA ; ***Z = kB ;***
4. Melakukan hashing pada R dan Z untuk mendapatkan key untuk meng-enkripsi pesan. Dari hasil hashing tersebut didapatkan nilai $\mathbf{k_1}$ dan $\mathbf{k_2}$.
5. Mengenkripsi pesan dengan menggunakan enkripsi simetrik dan $\mathbf{k_1}$ sebagai kunci dan dihasilkan sebuah chipertext.
6. Melakukan hashing pada Chipertext dan $\mathbf{k_2}$ secara bersamaan, hasil dari hashing akan digunakan sebagai Message authentication, yaitu untuk memeriksa apakah pesan yang diterima oleh penerima pesan sama dengan pesan ketika dikirim.
7. Mengirimkan nilai dari chipertext beserta atributnya, yaitu message authentication dan nilai ***R***.

3.3 Dekripsi Pesan

Adapun proses Dekripsi pesan secara keseluruhan adalah sebagai berikut :



Gambar 4. *Blok Diagram Dekripsi Pesan*

Keterangan :

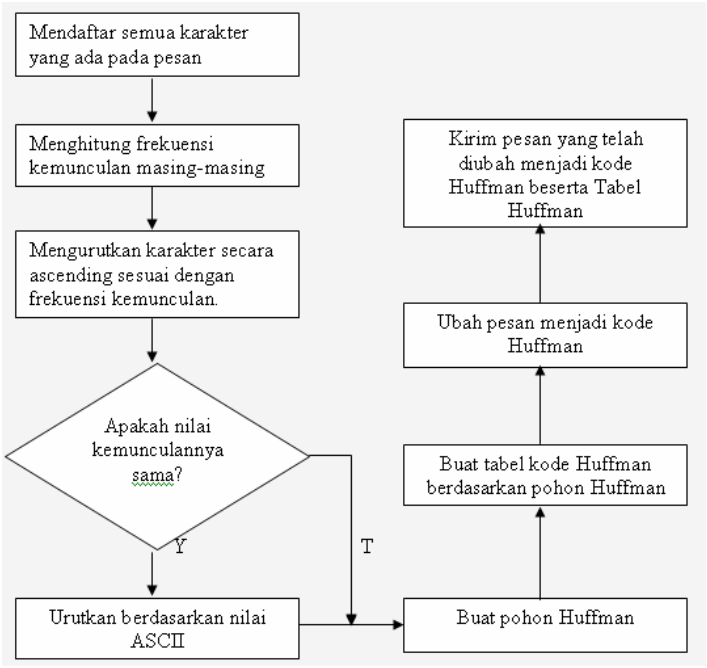
1. Dapatkan pesan yang dikirimkan beserta atributnya, yaitu chipertext, hasil hashing dari chipertext, dan nilai ***R***.
2. Hitung nilai ***Z*** dengan cara mengalikan kunci private dengan ***R***.

$$\boldsymbol{Z} = \mathbf{s}\boldsymbol{R}$$

3. Masukkan nilai ***R*** dan ***Z*** ke dalam hash function sehingga menghasilkan nilai baru yaitu k1dan k2.
4. Melakukan hashing pada Chipertext dan $\mathbf{k_2}$ secara bersamaan, hasil dari hashing akan digunakan sebagai Message authentication, yaitu untuk memeriksa apakah pesan yang diterima sama dengan pesan ketika dikirim.
5. Melakukan pengecekan, jika hasil hashing pada step 4 sama dengan ***t*** (hasil hashing ketika pesan dikirimkan) maka lanjutkan untuk mendekripsi pesan dengan menggunakan k1 sebagai kunci, namun jika hasilnya tidak sama maka keluar dari proses dan keluarkan alert.

3.4 Kompresi Pesan

Setelah melalui proses pengenkripsian maka ukuran text akan membengkak sehingga dibutuhkan suatu kompresi. Metode Kompresi yang digunakan adalah Huffman, karena metode ini terbukti memiliki nilai kompresi yang cukup tinggi namun untuk text yang ukerannya relatif kecil dan karaakter yang digunakan hampir berbeda, maka text yang akan dikirimkan mlh semakin membengkak. Suatu pohon Huffman dapat dibentuk dengan cara sebagai berikut :



Gambar 5. *Blok Diagram Kompresi Pesan*

1. Membuat simpul daun sebanyak sejumlah simbol
2. Memilih dua simbol dengan peluang terkecil dan dikombinasikan sebagai suatu simpul orangtua
3. Membuat simpul yang merupakan simpul orangtua dari dua simpul dengan peluang terkecil
4. Memilih sebuah simpul berikutnya (termasuk simpul baru) yang memiliki peluang terkecil
5. Melakukan prosedur yang sama pada dua symbol berikutnya yang memilki peluang terkecil
6. Ulangi proses diatas hingga semua node terlampaui, hingga didapatkan suatu pohon Huffman.

3.5 Dekompresi Pesan

Untuk proses pengembalian ke file aslinya, kita harus mengacu kembali kepada kode *Huffman* yang telah dihasilkan.

1. Ambillah satu-persatu bit hasil pemampatan mulai dari kiri.
2. Jika bit tersebut termasuk dalam daftar kode, lakukan pengembalian, jika tidak ambil kembali bit selanjutnya dan jumlahkan bit tersebut.

3.6 Analisis

Percobaan yang akan dilakukan adalah sebanyak 5 kali. Data kalimat yang akan dipakai adalah sebagai berikut :

1. Kriptografi
2. Kriptografi pertamakali dipergunakan pada tahun 400 SM di Yunani
3. kriptografi dapat didefinisikan sebagai metode untuk menyamarkan (merahasiakan) isi dari data
4. Proses kriptografi diawali dengan mengubah data dalam bentuk plaintext (tulisan atau pesan awal yang dapat dibaca) menjadi chipertext
5. Kriptografi kurva eliptik termasuk ke dalam sistem kriptografi kunci public yang mendasarkan keamanannya pada permasalahan matematis kurva eliptik

3.6.1 Enkripsi

Ujicoba ini digunakan untuk mengukur tingkat pembengkakan karakter yang disebabkan oleh enkripsi. Panjang pesan akan dianalisa sebelum dan sesudah dienkripsi.

Tabel 3. *Hasil Analisa Enkripsi*

Kalimat	Jumlah Karakter Sebelum Enkripsi	Jumlah Karakter Setelah Enkripsi	Pembengkakan Karakter (%)
Data 1	11	16	5/11 = 45 %
Data 2	64	64	0%
Data 3	93	96	3/93 = 3,22%
Data 4	133	136	3/133 = 2,25%
Data 5	146	152	6/146 = 4,1%

Dari table 3 telah didapatkan analisa pembengkakan hasil dari enkripsi pesan, namun pembengkakan jumlah karakter bersifat tidak tetap. Dari 5 percobaan telah didapatkan rata-rata pembengkakan karakter setelah enkripsi adalah 20,28%, namun hasil ini bukanlah patokan untuk pengukuran pembengkakan karakter karena jumlah karakter hasil enkripsi sangat bergantung dari jumlah karakter awal. Jumlah karakter hasil enkripsi adalah kelipatan 8 sehingga jika jumlah karakter awal adalah kelipatan 8 maka tidak akan mengalami pembengkakan jumlah karakter.

3.6.2 Kompresi

Ujicoba ini digunakan untuk mengukur tingkat pemangkasan karakter yang disebabkan oleh kompresi menggunakan metode Huffman. Panjang pesan akan dianalisa sebelum dan sesudah dikompresi.

Tabel 4 *Hasil Analisa Kompresi Global*

Kalimat	Jumlah Karakter Sebelum Kompresi	Jumlah Karakter Setelah Kompresi	Penyusutan Karakter (%)
Data 1	11	10	1/11 = 9,09 %
Data 2	64	46	18/64 = 28,125%
Data 3	93	63	30/93 = 32,25%
Data 4	133	91	42/133 = 31,5%
Data 5	146	97	49/146 = 33,56%

Dari tabel 4 telah didapatkan hasil analisa terhadap kompresi pesan Dari 5 percobaan di atas telah didapatkan rata-rata penyusutan karakter setelah kompresi adalah 26,9%.

4.3.3 Kompresi+Enkripsi

Ujicoba ini digunakan untuk mengurangi pembengkakan hasil enkripsi yatu dengan menggabungkannya dengan kompresi kemudian menganalisa perubahan karakternya.

Kalimat	Jumlah Karakter Sebelum	Jumlah Karakter Setelah Kompresi	Jumlah Karakter Setelah Kompresi +Enkripsi	Perubahan Karakter (%)
Data 1	11	10	16	5/11 = 45 %
Data 2	64	46	48	16/64 = 25%
Data 3	93	63	64	29/93 = 31,18%
Data 4	133	91	96	37/133 = 27,8%
Data 5	146	97	104	42/146 = 28,7%

Dari tabel diatas telah didapatkan hasil analisa terhadap kompresi+enkripsi pesan. Dari 5 percobaan di atas telah didapatkan hasil bahwa pembengkakan jumlah karakter dapat direduksi dengan cara menggabungkannya dengan kompresi. Tingkat penyusutan jumlah karakter bias mencapai 31%, meskipun pesan telah dienkripsi.

4. Hasil Dan Kesimpulan

Berdasarkan hasil percobaan pada bab sebelumnya, maka dapat disimpulkan bahwa :

1. Algoritma ECIES yang di implementasikan pada Mobile phone dalam proyek akhir ini dapat meningkatkan tingkat sekuritas pada pesan SMS.
2. Penggunaan enkripsi ini akan mempengaruhi ukuran pesan.
3. Pesan yang dihasilkan setelah dienkripsi merupakan kelipatan 8, sehingga jumlah karakter pesan awal sangat berpengaruh terhadap perubahan hasil enkripsi.
4. Hasil kompresi sangat dipengaruhi oleh banyaknya karakter yang sama pada pesan awal.

Daftar Pustaka

[1] **Kurniawan, Yusuf**. 2004. *Kriptografi keamanan internet dan jaringan komunikasi*. Bandung: INFORMATIKA.

[2] **Komputer, Wahana**. 2003. *Memahami Model Enkripsi & Security Data*. Yogyakarta: ANDI.

[3] **Munir, Rinaldi**. Agustus, 2006. ***Kriptografi***.Bandung

[4] **Heri Purwanto, Anny Kartika Sari.** *Aplikasi Kompresi SMS Teks (Short Message Service) Dengan Menggunakan Algoritma Huffman Kanonik dan LZW (Lempel-Ziv-Welch).* Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Gadjah Mada.

[5] Artikel dan Tutorial pada http://www.certicom.com

[6] Artikel dan Tutorial pada http://books.google.co.id

[7] **Liliana, Lipesik, V.J.,** 2006, *Pembuatan Perangkat Lunak untuk Kompresi File Text Dengan Menggunakan Huffman Tree*, Fakultas Teknologi Industri. Universitas Kristen Petra, Surabaya.

[8] Artikel dan Tutorial pada http://www.informatika.org/~rinaldi

[CV Penulis]

Eko Mardianto, menjalankan studi D4 bidang Teknik Informatika pada Politeknik Elektronika Negeri Surabaya – Institut Teknologi Sepuluh Nopember(PENS-ITS) semester 8.