Penerapan ECC El Gamal pada aplikasi *Chatting* dengan memanfaatkan Socket.IO pada perangkat Android

Moch Ginanjar Busiri
13513041
Teknik Informatika
Institut Teknologi Bandung
gbusiri@gmail.com

Abstrak – Pada paper ini dijelaskan penerapan dari algoritma enkripsi kunci publik yaitu Elliptic Curve Cryptography El Gamal (ECCEG) pada aplikasi *chatting* berbasis Android. Library yang digunakan untuk membangun aplikasi *chatting* ini adalah Socket.IO. Aplikasi *chatting* yang dimaksud adalah aplikasi *chatting* yang mempunyai karakteristik berumur pendek. Artinya tidak ada penyimpanan pada memori karena setelah meninggalkan *room chatting* tersebut semua histori akan dihapus. Hal ini sangat cocok dengan karakteristik ECCEG yang memiliki keterbatasan memori.

Kata Kunci : ECCEG, Socket, Android, Enkripsi, Dekripsi, Public Key, Private Key.

I. Pendahuluan

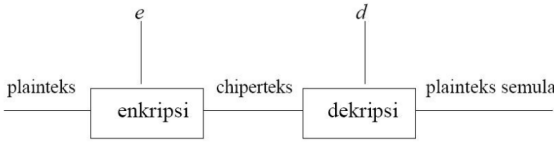
Perkembangan teknologi selalu beriringan dengan bertambahnya pengguna dan peredaran data. Meningkatnya kedua hal tersebut memicu tingginya resiko akan kejahatan dari para peretas yang mecoba mencuri atau menyalahgunakan data tersebut. Untuk mengatasi itu, para kriptografer mencoba mengembangkan suatu metode-metode kriptografi untuk mengamankan data dari para peretas sehingga muncul algoritma krigtografi yang sampai sekarang terus berkembang dari mulai kriptografi klasik sampai modern.

Salah satu algoritma kriptografi modern adalah algoritma kriptografi kunci public atau sering disebut dengan algoritma kunci asimetris. Algoritma kunci public adalah algoritma yang mempunyai dua kunci dalam pemrosesan enkripsi dan dekripsi. Pertama adalah kunci privat, kunci ini bersifat rahasia dan tidak boleh ada seorang pun yang mengetahuinya. Kedua adalah kunci public, kunci ini bersifat umum dan boleh diketahui orang lain untuk mengekstraksi pesan dari pengirim. Sehingga untuk mendekripsi pesan cukup dengan mengetahui kunci public orang yang mengirim pesan tersebut. Algoritma yang termasuk ke dalam algoritma asimetris ini salah satunya adalah ECC El Gamal.

II. Dasar Teori

a. Kriptografi Kunci Publik

Algoritma kunci public atau biasa disebut dengan algoritma asimetri adalah suatu algoritma dimana kunci enkripsi yang digunakan tidak sama dengan kunci dekripsi. Pada algoritma ini menggunakan dua kunci yakni kunci publik (public key) dan kunci privat (private key). Mekanisme algoritma kunci public dapat dilihat pada diagram berikut:



Gambar 1. Sistem kriptografi kunci-publik.

Plainteks dienkripsi dengan kunci e (kunci public) lalu menghasilkan cipherteks untuk dikirim kepada penerima. Lalu disisi penerima dia mendekripsi cipherteks tersebut dengan kunci d (kunci privat) menghasilkan plainteks semula. Algoritma kunci public mempunyai kelebihan dan kelemahan baik dalam keamanan maupun dalam kompleksitas waktu. Kelebihan algoritma kumci public:

- Masalah keamanan pada distribusi kunci dapat lebih baik
- Hanya kunci privat yang perlu dijaga kerahasiaannya oleh setiap entitas yang berkomuniaksi (tetapi, otentikasi kunci publik tetap harus terjamin). Tidak ada kebutuhan mengirim kunci kunci privat sebagaimana pada system simetri.
- Pasangan kunci publik/kunci privat tidak perlu diubah, bahkan dalam periode waktu yang panjang.
- Dapat digunakan untuk mengamankan pengiriman kunci simetri.
- Beberapa algoritma kunci-publik dapat digunakan untuk memberi tanda tangan digital pada pesan

Kelemahan algoritma kunci public:

- Enkripsi dan dekripsi data umumnya lebih lambat daripada sistem simetri, karena enkripsi dan dekripsi menggunakan bilangan yang besar dan melibatkan operasi perpangkatan yang besar.
- Ukuran cipherteks lebih besar daripada plainteks (bisa dua sampai empat kali ukuran plainteks).
- Ukuran kunci relatif lebih besar daripada ukuran kunci simetri.
- Karena kunci publik diketahui secara luas dan dapat digunakan setiap orang, maka cipherteks tidak memberikan informasi mengenai otentikasi pengirim.

b. Elliptic Curve Discrete Logarithm Problem (ECDLP)

ECDLP adalah dasar permasalahan dari ECC. Terdapat suatu persamaan umum kurva eliptik yang digunakan dalam pemrosesan algoritma ECC yaitu:

$$y = x^3 + ax + b \bmod p$$

Modulo dari persamaan diatas (p) harus bernilai prima. Ada beberapa hal unik yang terdapat pada persamaan kurva elips tersebut yaitu jika diambil 2 titik sembarang pada kurva tersebut, lalu 2 titik tersebut jika dioperasikan hasilnya akan berupa titik yang terdapat pada persamaan kurva tersebut pula. Berikut penjelasan formula untuk perhitungan 2 titik:

- Penjumlahan

Misalkan titik P(x_p, y_p) dan Q(x_q, y_q). Penjumlahan P+Q = R.

Koordinat titik R :

$$x_r = \lambda^2 - x_p - x_q \bmod p$$

$$y_r = \lambda(x_p - x_r) - y_p \bmod p$$

Dengan gradiennya adalah :

$$\lambda = \frac{y_p - y_q}{x_p - x_q} \bmod p$$

- Pengurangan

Misalkan titik P(x_p, y_p) dan Q(x_q, y_q). Penjumlahan P-Q = P+(-Q) = R. yang dalam hal ini -Q(x_q, -y_q) mod p.

- Penggandaan

Misalkan titik P(x_p, y_p) yang dalam hal ini y_p tidak nol. Penggandaan titik 2P = R.

Koordinat titik R:

$$x_r = \lambda^2 + 2x_p \bmod p$$

$$y_r = \lambda(x_p - x_r) - y_p \bmod p$$

Dengan gradiennya adalah :

$$\lambda = \frac{3x_p - a}{2y_p} \bmod p$$

Pada tahun 1985, Neal Koblitz dan Victor Miller secara terpisah mengajukan konsep elliptic curve cryptography (ECC). ECC berdasarkan pada DLP dalam suatu kelompok yang didefinisikan oleh titik-titik pada sebuah kurva elips terhadap suatu finite field. Implementasi dari ECC mencakup elliptic curve analogs of DSA (ECDSA), ElGamal, dan Diffie-Hellman.

c. Algoritma El Gamal

Keamanan dari algoritma ElGamal terletak pada sulitnya menghitung logaritma diskrit pada bilangan modulo prima yang besar. Pada algoritma ElGamal ini juga terdapat tiga proses, yaitu proses pembentukan kunci, proses enkripsi, dan proses dekripsi. Algoritma pembangkitan kunci:

- Pilih sembarang bilangan prima p (p dapat di-share di antara anggota kelompok).
- Pilih dua buah bilangan acak, g dan x, dengan syarat g < p dan $1 \leq x \leq p - 2$.
- Hitung $y = g^x \bmod p$.

Hasil dari algoritma ini adalah kunci publik, tripel (y, g, p) dan kunci privat, pasangan (x, p).

Algoritma enkripsi:

- Susun plainteks menjadi blok-blok m1, m2, ... , (nilai setiap blok di dalam selang [0, p-1]).
- Pilih bilangan acak k, yang dalam hal ini $1 \leq k \leq p - 2$.
- Setiap blok m dienkripsi dengan rumus

$$a = g^k \bmod p$$
$$b = y^k m \bmod p$$

Pasangan a dan b adalah cipherteks untuk blok pesan m, karena itu ukuran

cipherteks menjadi dua kali ukuran plainteksnya.

Algoritma dekripsi:

- Gunakan kunci privat x untuk menghitung $(a^x)^{-1} = a^{p-1-x} \bmod p$.
- Hitung plainteks m dengan persamaan :

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

d. Socket.IO Android

Socket.IO Android adalah salah satu library yang digunakan pada platform android untuk melakukan *chatting.* Socket.IO menyediakan event-oriented API yang berfungsi selama jaringan antara percakapan berlangsung. Socket.IO juga dapat diintegrasikan dengan platform web. Library ini mempunyai beberapa fitur diantaranya:

- Dapat digunakan sebagai fitur *chatting* yang bersifat personal maupun grup.
- Mengirim pesan kepada penerima yang bergabung dalam suatu room tertentu
- Dapat memberitahu user jika ada teman atau seseorang yang join atau leave grup pada room tertentu
- Dapat memberitahu user jika teman sedang mengetik.

Langkah – langkah untuk memasang Socket.IO dalam aplikasi Android adalah sebagai berikut:

- Inisialisasi atau definisikan Socket.IO

```
private Socket mSocket;
{
    try {
        mSocket = IO.socket("http://chat.socket.io")
    } catch (URISyntaxException e) {}
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mSocket.connect();
}
```

url http://chat.socket.io adalah url default yang digunakan untuk percobaan. Lalu pada method onCreate(), socket pun diaktifkan dengan cara memanggil connect().

- Kemudian untuk mengirim pesan melalui socket tersebut menggunakan emit() sehingga pesan yang dikirim akan langsung dikirim ke server dan diteruskan ke penerima.
- Lalu terdapat listener yang berfunsgi untuk memantau semua aksi yang dilakukan user. Socket.IO bersifat bidirectional artinya user dapat mengirim ke server bersamaan dengan server mengirim pesan ke user dalam waktu yang sama.
- Dan terakhir terdapat suatu socket manager yang digunakan untuk memutuskan koneksi jika *chatting* selesai yaitu dengan memanggil method socket.off() pada saat aplikasi dimatikan atau leave dari room. Hal inilah yang menjadi kelebihan dari socket sehingga tidak perlu menyimpan ke dalam memori dalam waktu yang lama karena setiap kali chatting berakhir maka semua histori akan dihapus.

Salah satu kekurangan pada library Socket.IO ini adalah belum adanya fitur enkripsi selama pengiriman pesan, oleh karena itu pada paper ini akan dikembangkan fitur enkripsi menggunakan algoritma ECC El Gamal.

III. Perancangan Aplikasi

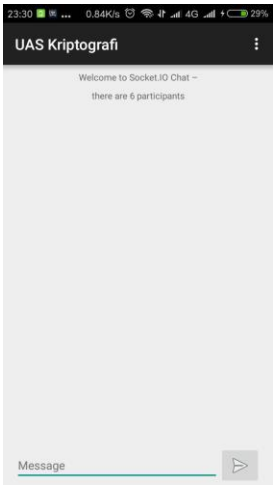
Aplikasi chatting Socket.IO yang diimplementasikan pada paper ini adalah aplikasi *client-side*, sedangkan untuk *server-side* digunakan server milik Socket.IO yang disediakan untuk demo atau percobaan yaitu http://chat.socket.io.

Perancangan aplikasi yang dibuat beserta contoh antarmukanya adalah sebagai berikut:

- Login to room

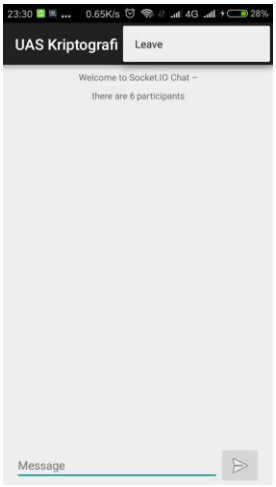
Pertama user harus memasukkan username atau nickname untuk masuk kedalam room. Setelah itu maka tekan tombol join untuk mulai *chatting*.

- Setelah masuk kedalam room, user bisa langsung memulai chat. Pada awal masuk akan terlihat notifikasi berapa orang yang sedang berada didalam room tersebut.



Tertulis ada 6 orang yang berada dalam room. 6 orang tersebut tidak diketahui (kemungkinan orang lain yang sedang memakai url yang sama) karena url server yang dipakai merupakan default dari socket.io itu sendiri, yang memungkinkan orang lain bisa se-room. Saat user memasukkan message, maka akan terlihat oleh seluruh orang yang berada di room.

- Leave room

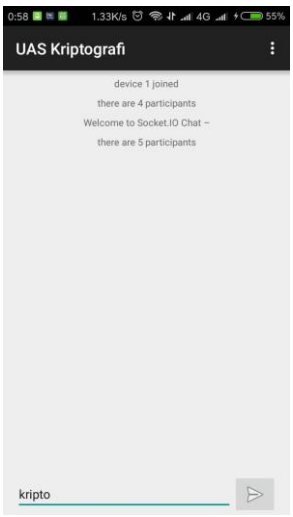


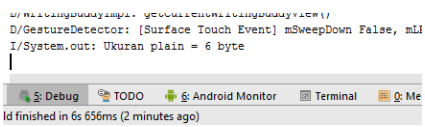
Untuk fitur leave grup diimplementasikan pada posisi menu bar kanan atas. Saat leave grup ditekan maka antarmuka akan kembali ke awal masukan username.

IV. Analisis

Setelah aplikasi selesai diimplementasikan, maka tahap selanjutnya adalah dilakukan analisis baik itu dalam segi kapasitas teks yang dikirim dan keamanan cipher teks.

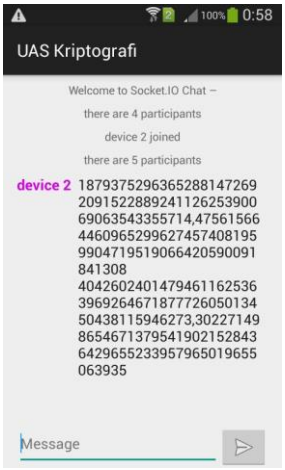
- Kapasitas Pesan Setelah Dienkripsi
  Simulasi dilakukan pada 2 device Android yang berbeda.
  Device 2 mencoba mengirim pesan berisi “kripto” kepada device 1, lalu pada debugger muncul ukuran byte dari pesan tersebut. Key privat dan public sudah di set pada aplikasi sehingga user tidak perlu memasukkan lagi secara manual.

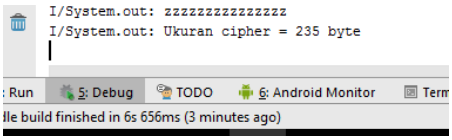




Terlihat ukuran byte adalah 6 byte.

Lalu setelah itu device 1 menerima pesan tersebut.





Terlihat bahwa ukuran dari cipher teks lebih besar dari plain teks awal yaitu 235 byte. Hal ini merupakan suatu kelemahan dari algoritma public key yaitu jumlah cipher teks akan lebih besar dari plain teks.

Pada device 1 (sebagai penerima) secara sengaja tidak dilengkapi dengan fitur dekripsi karena untuk membuktikan apakah pesan benar-benar terenkripsi atau tidak. Jika fitur dekripsi diterapkan maka tentunya pesan terenkripsi diatas akan kembali seperti semula.

- Keamanan Cipher Teks
  Anggap suatu titik Q adalah suatu titik yang berada pada kurva elips. Jika dikalikan konstanta k sehingga menghasilkan titik P itu sangat mudah. Akan tetapi jika diketahui titik P dan Q, untuk menentukan konstanta k sangatlah sulit dan banyak sekali kemungkinannya.

V. Kesimpulan

Algortima kunci public dapat diterapkan pada aplikasi chatting berbasis Android dengan menggunakan library Socket.IO. Penerapan ECC El Gamal sangat cocok dengan aplikasi ini karena cipher yang dihasilkan oleh algoritma ECC El Gamal ukurannya jauh lebih besar dibanding plain teks, sedangkan aplikasi chatting ini penyimpan didalam memori cenderung berumur pendek artinya setiap user leave dari chat tersebut semua data sebelumnya akan dihapus. Sehingga meskipun data yang telah terenkripsi berukuran besar sekali, hal ini tidak akan bertahan lama dan tidak akan memakan memori yang banyak dalam jangka waktu yang lama.

VI. Daftar Referensi

- http://socket.io/blog/native-socket-io-and-android/
- http://developer.android.com
- https://blogteknologiinformasi.wordpress.com/2011/11/01/kriptografi-asimetris/
- http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/kriptografi.htm/

VII. 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, 18 Desember 2016

Moch Ginanjar Busiri

13513041