ADVANCED ENCRYPTION STANDARD (AES) DENGAN ONE TIME PASSWORD UNTUK KEAMANAN LAYANAN SMS BANKING

Satya Fajar Pratama – NIM : 13506021

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if16021@students.if.itb.ac.id

Abstrak

SMS (*Short Messaging Service*) adalah suatu layanan pesan singkat yang disediakan oleh operator-operator telepon seluler melalui jaringan GSM. SMS merupakan teknologi yang sudah umum digunakan sejak *second generation mobile phone*. Awalnya, SMS hanya menjadi suatu fitur standar yang dimiliki oleh semua telepon seluler. Akan tetapi, kemudahan prosedur pengaksesan dan penggunaan SMS telah menjadikan fitur ini sebagai fitur telepon seluler yang paling diminati.

SMS Banking merupakan suatu mekanisme yang disediakan oleh bank bagi para nasabahnya untuk melakukan transaksi perbankan hanya dengan menggunakan layanan SMS. Layanan ini ditujukan agar para nasabah bank dapat melakukan transaksi tanpa terkendala dengan masalah ruang dan waktu. Dengan adanya layanan SMS Banking, transaksi-transaksi perbankan dapat segera dilakukan sehingga ketersampaian hasil transaksi dari pengirim kepada pihak penerima menjadi lebih cepat.

Untuk menjaga keberhasilan setiap transaksi perbankan yang dilakukan, maka aspek keamanan perlu diberikan perhatian secara khusus. Namun, pada kenyataannya, kasus-kasus kriminalitas cukup banyak terjadi pada layanan SMS Banking. Hal ini disebabkan kerahasiaan pesan pada layanan SMS Banking belum terjaga dengan baik. Proses enkripsi pesan yang terlibat dalam layanan tersebut hanyalah terjadi antara telepon seluler dengan BTS (*Base Transceiver Station*). Oleh karena itu, celah atau kesempatan untuk melakukan serangan (*attack*) terhadap layanan SMS Banking masih terbuka.

Advanced Encryption Standard (AES) dengan *One Time Password* merupakan suatu metode enkripsi alternatif yang dapat digunakan untuk menjaga kerahasiaan layanan SMS Banking. Metode ini memungkinkan terjadinya enkripsi pesan antara telepon seluler dengan server bank yang dituju (*end-to-end communication*). Dengan memanfaatkan metode ini, layanan SMS Banking dapat menjadi lebih aman karena algoritma enkripsi yang digunakan cukup tangguh dan kunci yang dipakai sukar dipecahkan.

***Kata kunci**: Advanced Encryption Standard, One Time Password*, SMS, SMS Banking, *Mobile Banking*

1. Pendahuluan

Jaringan Global System for Mobile communication (GSM) awalnya didesain khusus untuk *voice communication*. Akan tetapi, seiring dengan meningkatnya penggunaan telepon seluler di dunia, para pemilik telepon seluler tersebut mulai menggunakannya untuk melakukan pengiriman data.

Short Message Service (SMS) adalah jenis transmisi data yang paling sering digunakan oleh para pemilik telepon seluler. Layanan ini sangat memudahkan para pengguna telepon seluler untuk melakukan pengiriman pesan kepada pihak yang dituju. Selain itu, layanan SMS juga relatif murah apabila dibandingkan dengan layanan-layanan telepon seluler lainnya. Hal inilah yang membuat layanan SMS menjadi sangat populer di dunia.

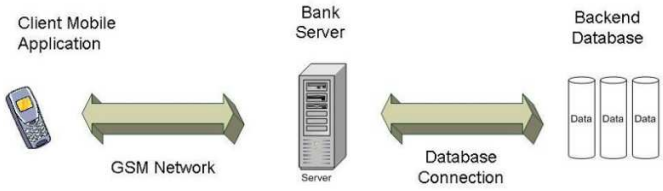
SMS Banking merupakan suatu mekanisme perbankan yang memungkinkan para nasabah untuk melakukan transaksi keuangan hanya dengan menggunakan layanan SMS. Layanan ini ditujukan untuk memudahkan para nasabah dalam melakukan transaksi kapanpun dan dimanapun. Layanan SMS Banking akan mewujudkan harapan para nasabah untuk melakukan transaksi secara *real time*.

Masalah-masalah yang terdapat pada SMS Banking disebabkan pesan SMS yang belum sepenuhnya aman. Masih banyak terdapat kelemahan pada arsitektur GSM yang dapat mengakibatkan ***security shortfalls*** pada SMS Banking. Salah satu contoh kelemahan GSM tersebut adalah proses enkripsi pesan SMS yang hanya terjadi antara telepon seluler dengan BTS (*Base Transceiver Station*).

Kelemahan-kelemahan yang dimiliki oleh arsitektur GSM tersebut dapat ditangani menggunakan metode enkripsi *Advanced Encryption Standard* (AES) dengan *One Time Password* untuk menjaga kerahasiaan layanan SMS Banking yang dilakukan. Metode ini akan menyediakan enkripsi pesan antara telepon seluler dengan server bank yang dituju sehingga faktor keamanan pada layanan SMS Banking dapat meningkat.

2. SMS Banking *Overview*

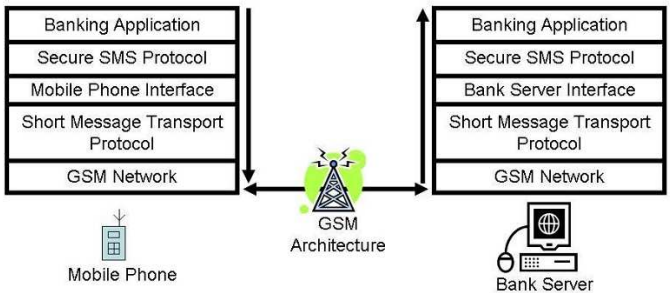
Arsitektur suatu sistem SMS Banking yang aman akan terbagi menjadi 3 bagian. Bagian pertama adalah aplikasi telepon seluler yang akan menghasilkan pesan SMS dan mengirimkan pesan tersebut melalui jaringan GSM kepada server bank yang dituju. Bagian berikutnya adalah server bank yang akan menerima pesan SMS tersebut dan menerjemahkannya ke dalam struktur format internal. Bagian terakhir adalah basis data (*backend database*) yang akan menyimpan seluruh transaksi yang pernah dilakukan oleh seorang nasabah. Gambar berikut akan mengilustrasikan arsitektur SMS Banking yang telah dijelaskan sebelumnya.



Gambar 1 Arsitektur SMS Banking

2.1 Layer Protokol SMS Banking

Pada sistem SMS Banking, para pemilik telepon seluler akan selalu menjadi pihak yang memulai terjadinya suatu komunikasi. Berikut adalah penjelasan mengenai layer-layer protokol yang telah diilustrasikan pada gambar 2 :



Gambar 2 Layer Protokol SMS Banking

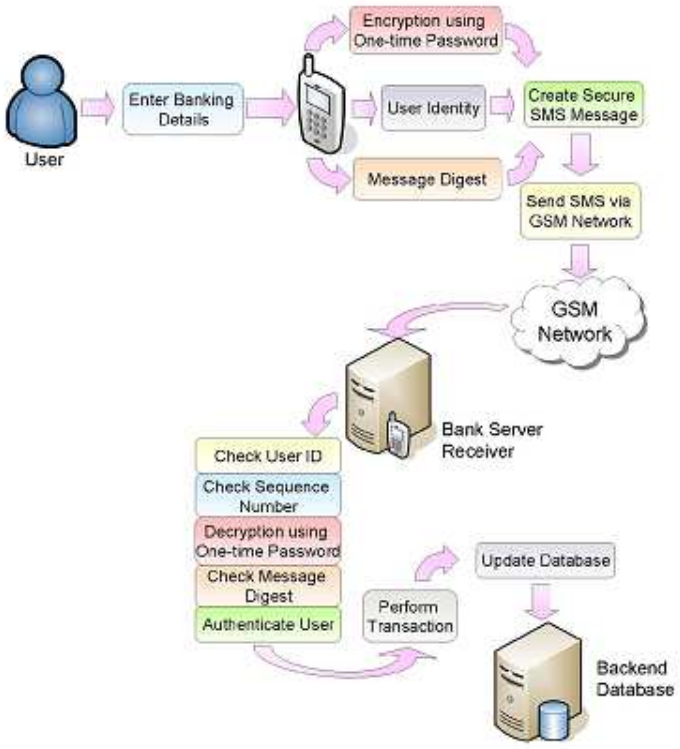
- Layer *Banking Application* akan menyediakan antarmuka (*interface*) antara pengguna dengan aplikasi.
- Layer *Secure SMS Protocol* akan menerjemahkan informasi yang diperoleh dari layer *Banking Application* ke dalam format standar pesan SMS.
- Layer *Mobile Phone Interface* adalah suatu platform telepon seluler yang telah didefinisikan oleh pihak pembuat telepon seluler tersebut.
- Layer *Bank Server Interface* adalah suatu antarmuka (*interface*) program aplikasi pada server bank.
- Layer *Short Message Transport Protocol* adalah protokol transmisi standar untuk pesan SMS yang digunakan oleh jaringan GSM.

Ketika server bank yang dituju menerima pesan SMS yang telah dikirimkan, server bank tersebut akan mentranslasikan pesan yang diterima ke dalam format yang dapat dimengerti oleh aplikasi server supaya transaksi perbankan dapat dilakukan.

2.2 *Protocol Sequence* pada SMS Banking

Pada jaringan GSM, pesan SMS akan dikirimkan secara *asynchronous*. Pihak pengirim tidak akan mengetahui apakah pesan yang dikirim tersebut telah diterima tanpa adanya notifikasi dari server berupa *delivery report*. Gambar 3 akan ngilustrasikan protokol yang digunakan dalam SMS Banking.

Aplikasi yang terdapat pada telepon seluler akan meminta seluruh informasi yang dibutuhkan dari pengguna (*user*). Informasi ini akan digunakan untuk menghasilkan suatu pesan SMS yang aman untuk dikirimkan kepada server bank yang dituju. Selanjutnya, enkripsi akan dilakukan pada bagian-bagian pesan SMS tersebut. Protokol



Gambar 3 Protokol SMS Banking

SMS Banking tidak akan melakukan enkripsi pada keseluruhan pesan. Hal ini dilakukan supaya server bank yang dituju tetap dapat mengidentifikasi identitas nasabahnya.

Ketika server bank yang dituju menerima pesan SMS dari jaringan GSM, pesan tersebut akan diterjemahkan ke dalam suatu struktur internal server bank tersebut. Berikutnya, server bank akan membaca identitas nasabah dari pesan yang diterima dan melakukan pengecekan apakah identitas tersebut terdapat dalam basis data. Kemudian, server bank akan mengambil *sequence number* dari account nasabah tersebut dan membandingkannya dengan hasil pembacaan dari pesan yang diterima.

Apabila perbandingan *sequence number* tersebut tepat, server bank akan membaca *one time password* dari basis data dan menggunakannya untuk mendekripsi pesan yang diterima. Setelah proses dekripsi selesai dilakukan, *one time password* tersebut akan dimusnahkan dan *sequence number* pada account nasabah tersebut akan diinkriminasi. Jika seluruh proses pemeriksaan tersebut berhasil dilakukan, maka server bank akan melakukan transaksi perbankan yang diminta.

3. *Advanced Encryption Standard* (*AES*)

3.1 Algoritma Rijndael

Seperti pada *DES*, *Rijndael* menggunakan substitusi dan permutasi, dan sejumlah putaran. Untuk setiap putarannya, *Rijndael* menggunakan kunci yang berbeda. Kunci setiap putaran disebut *round key*. Tetapi tidak seperti *DES* yang berorientasi bit, *Rijndael* beroperasi dalam orientasi *byte* sehingga memungkinkan untuk implementasi algoritma yang efisien ke dalam *software* dan *hardware* [1].
Garis besar algoritma *Rijndael* yang beroperasi blok 128-bit dengan kunci 128-bit adalah sebagai berikut:

1. *AddRoundKey*: melakukan *XOR* antara *state* awal (plainteks) dengan *cipher key*. Tahap ini disebut juga *initial round*.
2. Putaran sebanyak *Nr* – 1 kali. Proses yang dilakukan pada setiap putaran adalah:
   a. *ByteSub*: substitusi byte dengan menggunakan tabel substitusi (*S-box*). Tabel substitusi dapat dilihat pada tabel 2, sedangkan ilustrasi *ByteSub* dapat dilihat pada gambar 9.
   b. *ShiftRow:* pergeseran baris-baris *array state* secara *wrapping*. Ilustarsi *ShiftRow* dapat dilihat pada gambar 10.
   c. *MixColumn*: mengacak data di masing-masing kolom *array state*. Ilustarsi *MixColumn* dapat dilihat pada gambar 11.
   d. *AddRoundKey:* melakukan *XOR* antara *state* sekarang dengan *round key*. Ilustarsi *AddRoundKey* dapat dilihat pada gambar 12.
3. *Final round*: proses untuk putaran terakhir:
   a. *ByteSub*.
   b. *ShiftRow*.
   c. *AddRoundKey*.

Diagram proses enkripsi *AES* dapat dilihat pada Gambar 4.

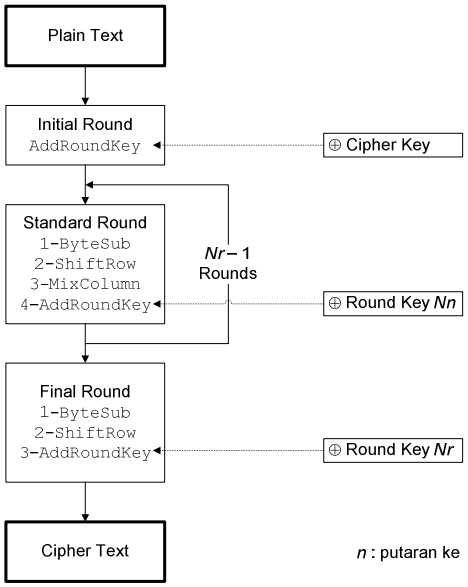
Algoritma *Rijndael* mempunyai 3 parameter sebagai berikut:

1. `plainteks` : *array* yang berukuran 16 *byte*, yang berisi data masukan.

2. `cipherteks` : *array* yang berukuran 16 *byte*, yang berisi hasil enkripsi.
3. `key` : *array* yang berukuran 16 *byte*, yang berisi kunci ciphering (disebut juga *cipher key*).

Dengan 16 *byte*, maka baik blok data dan kunci yang berukuran 128-bit dapat disimpan di dalam ketiga array tersebut (128 = 16 x 8).

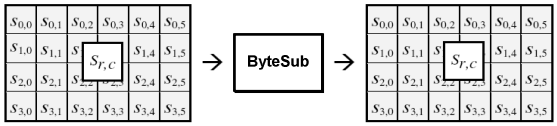
Selama kalkulasi plainteks menjadi cipherteks, status sekarang dari data disimpan di dalam *array of byte* dua dimensi, `state`, yang berukuran `NROWS x NCOLS`. Elemen *array* `state` diacu sebagai *S*[*r*,*c*], dengan $0 \le r < 4$ dan $0 \le c < Nc$ (*Nc* adalah panjang blok dibagi 32). Pada *AES*, *Nc* = 128/32 = 4.



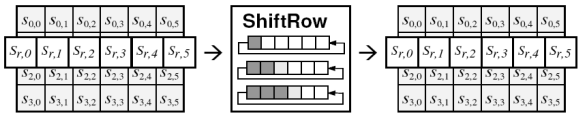
Gambar 4 Diagram Proses Enkripsi *AES*

Tabel 1 Tabel *S-box* yang digunakan dalam transformasi *ByteSub() AES*

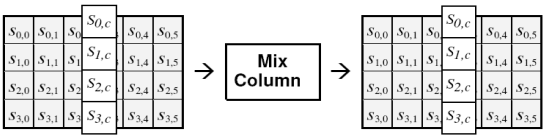
hex	y 0	1	2	3	4	5	6	7	8	9	a	b	c	d	e
x 0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb



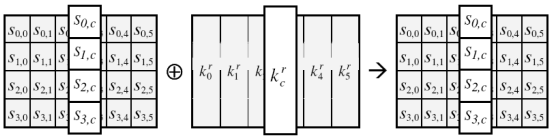
Gambar 5 Ilustrasi Transformasi *ByteSub() AES*



Gambar 4 Ilustrasi Transformasi *ShiftRow() AES*



Gambar 7 Ilustrasi Transformasi *MixColumn() AES*



Gambar 8 Ilustrasi Transformasi *AddRoundKey() AES*

3.2 Cipher Kebalikan (*Inverse Cipher*)

Cipher kebalikan merupakan algoritma kriptografi *AES* yang digunakan untuk melakukan proses dekripsi cipherteks menjadi plainteksnya. Secara garis besar, *cipher* kebalikan yang beroperasi blok 128-bit dengan kunci 128-bit adalah sebagai berikut:

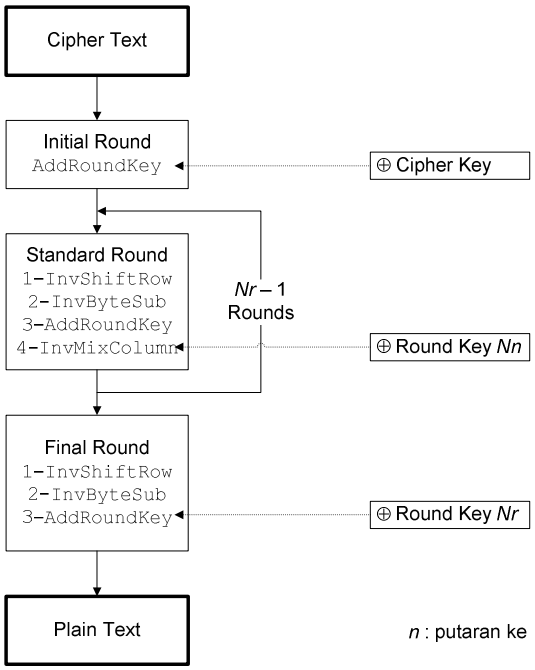
1. *AddRoundKey*: melakukan *XOR* antara *state* awal (cipherteks) dengan *cipher key*. Tahap ini disebut juga *initial round*.
2. Putaran sebanyak *Nr* – 1 kali. Proses yang dilakukan pada setiap putaran adalah:
   a. *InvShiftRow:* pergeseran baris-baris *array state* secara *wrapping*.
   b. *InvByteSub*: substitusi byte dengan menggunakan tabel substitusi kebalikan (*inverse S-box*). Tabel substitusi dapat dilihat pada tabel 3.
   c. *AddRoundKey:* melakukan *XOR* antara *state* sekarang dengan *round key*.
   d. *InvMixColumn*: mengacak data di masing-masing kolom *array state*.

3. *Final round*: proses untuk putaran terakhir:
    a. *InvShiftRow*.
    b. *InvByteSub*.
    c. *AddRoundKey*.

Tabel 2 Tabel *S-box* yang digunakan dalam transformasi *InvByteSub() AES*

hex	y 0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x 0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Diagram proses dekripsi *AES* dapat dilihat pada Gambar 9.



Gambar 5 Diagram Proses Dekripsi *AES*

4. *One Time Password* (*OTP*)

Kekuatan algoritma enkripsi kunci simetri terletak pada algoritma penghasil kunci enkripsi tersebut. Apabila kunci yang dihasilkan dapat diprediksi, maka algoritma tersebut akan tergolong lemah. Kunci yang dihasilkan juga haruslah berupa karakter-karakter yang dapat diinputkan oleh para pengguna dengan menggunakan telepon selulernya. Oleh karena itu, algoritma penghasil kunci yang digunakan harus mampu menghasilkan karakter acak yang sesuai dan tidak dapat diprediksi (*non-predictable*).

Program yang digunakan untuk menghasilkan kunci enkripsi tersebut akan diinisialisasi dengan suatu himpunan karakter yang sesuai. Selanjutnya, program akan membangkitkan sebuah bilangan acak yang digunakan sebagai indeks untuk memilih karakter dari himpunan tersebut dan menjadikannya sebagai salah satu karakter pembentuk *password.*

Pemilihan karakter menjadi sebuah *password* membutuhkan suatu pengacak bilangan yang aman. Dalam hal ini, algoritma SHA1 akan digunakan sebagai algoritma penghasil bilangan acak. Hal ini disebabkan sifat algoritma SHA1 yang hanya *one-way-hashing* sehingga bilangan acak yang dihasilkan tidak dapat diprediksi. Selain itu, efisiensi algoritma SHA1 dalam melakukan kalkulasi *message digest* pada lingkungan telepon seluler juga menjadi alasan lainnya pemilihan algoritma ini.

5. Implementasi

Tabel 3 menunjukkan implementasi algoritma *Advanced Encryption Standard* (*AES*) dalam bahasa Java dengan menggunakan Bouncy Castle library pada crypto package.

```
public static byte[] encodeMessage (byte[]
content, String password, String cipherAlg)
throws Exception {
        byte key[] = password.getBytes();

        BufferedBlockCipher cipherEngine =
new
PaddedBufferedBlockCipher(createEngine(ciph
erAlg));
        cipherEngine.init(true,          new
KeyParameter(key));

        //Encrypt message
        byte[]     cipherText     =     new
byte[cipherEngine.getOutputSize(content.len
gth)];
        int        cipherTextLength        =
cipherEngine.processBytes(content,        0,
content.length, cipherText, 0);
        cipherEngine.doFinal(cipherText,
cipherTextLength);

        ByteArrayOutputStream   out   =   new
ByteArrayOutputStream();
        DataOutputStream    dout    =    new
```

```
DataOutputStrream();

        dout.writeShort(cipherText.length);
        dout.write(cipherText);
        return out.toByteArray();

}
```

Tabel 3 *AES* dalam Java

Selain itu, berikut adalah *screenshot* dari aplikasi SMS Banking yang diimplementasikan dalam Java™ Wireless Toolkit emulator.

Gambar 10 User Interface SMS Banking Application

6. Pengujian Performansi

Kekuatan proteksi SMS Banking akan bergantung pada kekuatan algoritma enkripsi dan algoritma penghasil kunci enkripsi.

Apabila seorang *attacker* melakukan *brute force attack* untuk mendapatkan isi dari suatu pesan yang dikirimkan, maka penjelasan kalkulasi berikut akan menggambarkan waktu yang dibutuhkan oleh *attacker* untuk mendapatkan isi pesan tersebut.

Karakter-karakter yang sesuai dengan lingkungan telepon seluler adalah :
[a-z] = 26; [A-Z] = 26; [0-9] = 10
Oleh karena itu, terdapat 26 + 26 + 10 = 62 kemungkinan karakter.

Diasumsikan *attacker* menggunakan komputer yang cukup canggih untuk melakukan serangan yang dapat melakukan 10^6 dekripsi per detik.

Panjang kunci yang mungkin adalah 8 sampai 16 karakter. Oleh karena itu, *attacker* akan melakukan percobaan sebanyak $62^8 + 62^9 + 62^{10} + 62^{11} + 62^{12} + 62^{13} + 62^{14} + 62^{15} + 62^{16} = 4.845 \times 10^{28}$. Dengan mengasumsikan bahwa rata-rata *attacker* harus melakukan percobaan sebanyak setengah kali dari seluruh percobaan yang dilakukan, maka *attacker* harus melakukan 2.422×10^{28} percobaan sebelum *attacker* tersebut dapat membaca isi pesan yang diinginkan.

Waktu yang dibutuhkan oleh *attacker* untuk memecahkan enkripsi pesan tersebut adalah $2.422 \times 10^{28} / 10^6 \times 60 \times 60 \times 24 \times 365.25$ tahun $= 7.677 \times 10^{14}$ tahun.

Penggunaan *One Time Password* akan memberikan tingkat keamanan yang lebih tinggi bagi protokol yang digunakan. Apabila seorang *attacker* telah mengetahui PIN seorang nasabah dan *sequence number* seorang *user*, *attacker* tersebut tetap akan membutuhkan *One Time Password* untuk mengenkripsi pesan supaya server bank yang dituju dapat menjalankan transaksi perbankan yang diminta.

7. Kesimpulan

Advanced Encryption Standard (*AES*) dengan *One Time Password* merupakan salah satu metode alternatif yang dapat digunakan untuk meningkatkan keamanan layanan SMS Banking. Hal ini disebabkan algoritma enkripsi *AES* yang cukup tangguh dan penggunaan *One Time Password* yang sukar dipecahkan yang mengakibatkan seorang *attacker* akan membutuhkan waktu yang sangat lama untuk mengetahui isi suatu pesan. Selain itu, metode alternatif ini juga memungkinkan terjadinya enkripsi pesan antara telepon seluler dengan server bank yang dituju (*end-to-end communication*) sehingga tingkat keamanan dalam melakukan transaksi perbankan via SMS Banking lebih terjaga.

Daftar Pustaka

[1] Munir, Rinaldi. (2004). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[2] Public Key Cryptography Standards (PKCS), No.1, RSA Encryption standard, http://www.rsasecurity.com/rsalabs/.

[3] Margrave, D. GSM Security and Encryption. http://www.hackcanada.com/blackcrawl/cell/gsm/gsm-secur/gsm-secur.html.

[4] Schneier, Bruce. (1996). Applied Cryptography 2nd. John Wiley & Sons.