Penerapan *Digital Signature* pada Aplikasi SMS Android

Andi Kurniawan Dwi P. / 13508028
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
If18028@students.if.itb.ac.id

***Abstrak*— Perkembangan teknologi di bidang komunikasi semakin tahun semakin maju. Salah satu hasil perkembangan teknologi di bidang komunikasi adalah layanan SMS (*Short Message Service)*. Semua jenis telepon seluler (*Hand Phone)* pasti mempunyai layanan komunikasi ini. Sebagian besar orang lebih sering menggunakan layanan SMS daripada layanan telepon karena biaya nya yang tergolong murah dan mudah digunakan. Karena banyaknya informasi-informasi penting yang dikirimkan melalui SMS, maka diperlukan upaya untuk menjamin aspek keamanan informasi yang terkandung dalam SMS. Aspek keamanan dalam kriptografi ada 4 bagian utama, yaitu: kerahasiaan, otentikasi, keaslian pesan, dan anti-penyangkalan. Aspek pertama dapat diatasi dengan mengenkripsi isi pesan. Aspek 2 sampai 4 dapat diatasi dengan *digital signature* (tanda tangan digital). Pada makalah ini, akan dikembangkan aplikasi DigiSMS yang menambahkan fitur *digital signature* pada pesan sms. Aplikasi ini rencananya akan dikembangkan di platform android karena pengguna android tergolong banyak. Fungsi utama dari aplikasi ini adalah untuk menjamin aspek keamanan pada pesan sms. Terdapat tiga fitur utama dari aplikasi ini. Fitur pertama menghasilkan *digital signature* dari isi pesan dengan parameter kunci publik, kemudian *digital signature* tersebut ditambahkan pada akhir pesan. Fitur kedua yaitu untuk autentikasi isi pesan dengan menggunakan parameter kunci privat. Fitur ketiga adalah menghasilkan kunci publik dan kunci privat dari nilai p, q, dan e. Algoritma *digital signature* yang akan digunakan adalah algoritma SHA dan algoritma RSA.**

***Kata Kunci*— android, *digital signature*, RSA, SHA, SMS.**

I. PENDAHULUAN

Beberapa tahun ini perkembangan teknologi komunikasi semakin pesat. Telepon seluler merupakan salah satu hasil dari perkembangan terknologi komunikasi. Telepon seluler mempermudah orang untuk berkomunikasi satu sama lain. Dengan adanya teknologi ini dunia terasa sempit karena seseorang dapat berkomunikasi dengan orang lain yang jaraknya jauh. Di dalam telepon seluler ini ada beberapa fungsi komunikasi yang dapat digunakan antara lain telepon, *video call*, SMS, MMS, *chatting*, internet, dan lain-lain. Di antara layanan komunikasi tersebut, layanan SMS yang menjadi komunikasi favorit karena sudah dipastikan semua telepon seluler memiliki layanan ini dan yang paling penting adalah biayanya yang tergolong murah.

SMS merupakan layanan pesan singkat dari telepon seluler yang digemari masyarakat untuk saling berkomunikasi melalui pesan teks. Diperkirakan jumlah SMS yang dikirimkan melalui suatu provider komunikasi mencapai ratusan juta per hari. Angka ini bahkan melonjak drastis hingga mencapai milyaran di hari raya seperti lebaran atau natal. Dari banyaknya informasi yang terkirim melalui layanan SMS ini, muncullah potensi serangan terhadap kemanan layanan SMS dari pihak-pihak yang tidak bertanggung jawab. Ada beberapa risiko yang dapat mengancam keamanan pesan pada layanan SMS antara lain SMS *spoofing*, SMS *snooping*, dan SMS *interception*. SMS *spoofing* merupakan pengiriman sms di mana nomor pengirim yang tertera bukanlah nomer pengirim yang sebenarnya. SMS *snooping* lebih sering terjadi karena kelalaian pengguna telepon seluler. Contohnya ketika seseorang meminjamkan telepon selulernya pada orang lain untuk menggunakan telepon selulernya. Pada saat itu orang tersebut dapat dengan sengaja atau tidak membuka isi pesan yang ada pada *inbox* SMS. Pesan yang bersifat personal atau rahasia dapat dibaca dengan mudah oleh orang lain melalui cara ini. Celah keamanan terbesar pada layanan komunikasi SMS adalah pada saat SMS tersebut sedang dikirim melalui jaringan SMS tersebut. SMS bekerja pada jaringan nirkabel yang memungkinkan terjadinya pencurian isi pesan SMS ketika dalam proses transmisi dari pengirim ke penerima. Kasus ini disebut SMS *interception.* Selain itu, SMS tidak dapat dijadikan suatu barang bukti ketika terjadi suatu kasus kriminalitas karena kevalidan isi pesan dalam SMS tidak dapat dijamin.

Jika dihubungkan dengan aspek keamanan yang ada di dalam kriptografi, aplikasi enkripsi sederhana hanya mampu menangani aspek pertama saja, aspek kerahasiaan. Aspek otentikasi, keaslian pesan, dan anti-penyangkalan tidak dapat ditangani dengan mengenkripsi isi pesan dalam SMS. *Digital signature* dapat digunakan untuk mengatasi ketiga aspek di atas. Fungsi utama dari *digital signature* adalah menambahkan signatrue (tanda tangan) pada sebuah pesan sehingga pesan tersebut dapat diketahui keaslian identitas pengirim dan integritas isi pesan. Signature ini dapat disusun dari inforamsi tertentu yang dianggap unik, di mana setiap orang akan memiliki signature yang berbeda sehingga dapat diketahui orisinilitas dari sebuah pesan yang dikirim.

II. DASAR TEORI

A. SMS (Short Message Service)

SMS (*Short Message Service*) merupakan sebuah layanan komunikasi yang ada pada telepon seluler untuk mengirim dan menerima pesan-pesan pendek. SMS pertama kali dikenalkan pada tanggal 3 Desember 1982. SMS pertama di dunia dikirimkan menggunakan jaringan GSM milik operator telepon bernama Vodafone. SMS pertama ini dikirimkan oleh ahli bernama Neil Papwort kepada Richard Jarvis menggunakan komputer.

SMS dihantarkan pada *channel signal* GSM (*Global System for Mobile Communication*) dengan spesifikasi teknis ETSI. SMS diaktifkan oleh ETSI dan dijalankan di *scope* 3GPP. SMS juga digunakan pada teknnologi GPRS dan CDMA. SMS menjamin pengiriman pesan oleh jaringan, jika terjadi kegagalan pesan akan disimpan dahulu di jaringan dan akan dikirmkan lagi ketika jaringan sudah stabil.

B. Platform Android

Android adalah sistem operasi untuk telepon seluler berbasis Linux. Android menyediakan platform terbuka bagi para pengembang untuk membangun aplikasi yang dapat dijalankan di bermacam telepon seluler. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru dalam teknologi telepon seluler. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan piranti keras, piranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Pada saat perilisan perdana Android, 5 November 2007, Android bersama Open Handset Aliance menyatakan mendukung pengembangan standar terbuka pada telepon seluler.

Fitur yang tersedia di Android antara lain:

- *Framework* aplikasi yang mendukung penggantian komponen dan *reusable*
- *Dalvik virtual machine*
- *Integrated browser*
- *Grafik* berdasarkan OpenGL
- *SQLite* untuk penyimpanan data
- *Multimedia support*
- Linkungan Development yang lengkap dan kaya termasuk perangkat emulator, tools untuk debugging, profil dan kinerja memori, dan plugin untuk IDE Eclipse.

C. Kriptografi

Kriptografi, secara umum adalah ilmu dan seni untuk menjaga kerahasiaan berita. Selain pengertian tersebut terdapat pula pengertian ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data. Tidak semua aspek keamanan informasi ditangani oleh kriptografi.

Ada empat tujuan mendasar dari ilmu kriptografi ini yang juga merupakan aspek keamanan informasi yaitu :

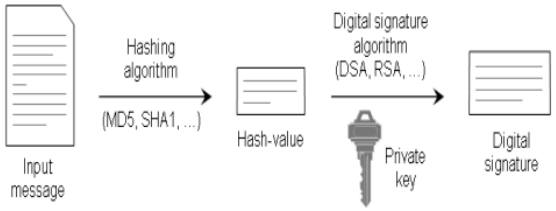
- Kerahasiaan, adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka/ mengupas informasi yang telah disandi.
- Integritas data, adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubsitusian data lain kedalam data yang sebenarnya.
- Autentikasi, adalah berhubungan dengan identifikasi/ pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus diautentikasi keaslian, isi datanya, waktu pengiriman, dan lain-lain.
- Non-repudiasi, atau nirpenyangkalan adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman/ terciptanya suatu informasi oleh yang mengirimkan/ membuat.

D. Digital signature

Digital signature (tanda tangan) sudah digunakan untuk otentikasi dokumen cetak sejak zaman dahulu. Tanda tangan memiliki karakteristik sebagai berikut:

- Tanda tangan adalah bukti otentik
- Tanda tangan tidak dapat dilupakan
- Tanda tangan tidak dapat dipindah untuk digunakan ulang
- Dokumen yang telah ditandatangi tidak dapat diubah
- Tanda tangan tidak dapat disangkal

Berikut merupakan gambaran proses penyisipan tanda tangan digital ke dalam suatu pesan:

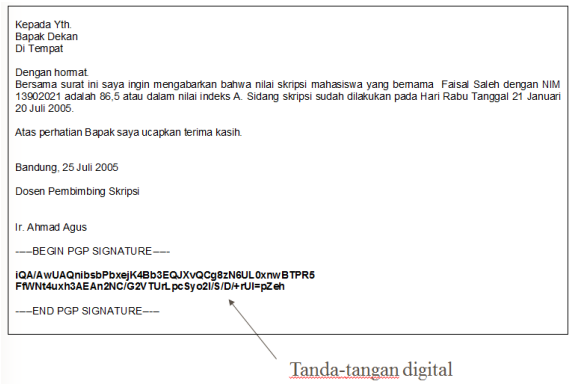


Gambar 1. Penyisipan tanda tangan digital pada pesan

Langkah pertama yang dilakukan adalah dengan mencari nilai hash dari pesan. nilai hash dapat ditentukan dengan menggunakan salah algoritma hash misalnya, MD2, MD4, MD5, SHA1, SHA-256, dan lainnya. Nliai hash untuk file yang berbeda selalu berbeda juga walaupun sebenarnya ada kemungkinan yang sangant

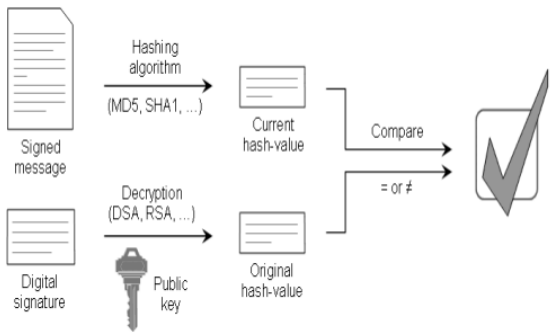
kecil bahwa dapat menghasilkan nilai hash yang sama.

Setelah mendapatkan nilai hash dari pesan tersebut, maka nilai tersebut dienkripsi dengan menggunakan algoritma asimetri untuk *digital signature*. Algoritma yang paling sering digunakan aalah RSA (berdasarkan teori bilangan), DSA (berdasarakan teori logaritma diskrit), dan ECDSA (berdasarkan teori kurva melengkung). Dalam kasus ini, enkripsi menggunakan kunci private sedangkan kunci publik digunakan untuk proses diekripsi. Sehingga kunci private hanya dimiliki oleh pengirim pesan, namun pengirim pesan memberikan kunci publiknya ke orang lain dengan tujuan agar pesan tersebut nantinya dapat diotentikasi. Setelah dienkripsi, hasil enkripsinya ditambahkan ke dalam pesan tersebut dengan mekanisme tertentu. Contoh penambahan *digital signature* adalah pada surat email di bawah ini:



Gambar 2. Contoh Letak *Digital signature* pada Email

Untuk proses otentikasi suatu pesan, tahapan yang dilakukan dapat dilihat pada diagram di bawah ini:



Gambar 3. Proses Otentikasi Pesan dengan *Digital signature*

Langkah pertama yang dilakukan adalah melakukan hash pada pesan aslinya. Kemudian, nilai hash dari pesan asli nantinya akan dibandingkan dengan nilai hash hasil dekripsi *digital signature* yang diletakkan dalam pesan. langkah kedua, adalah melakukan dekripsi dari *digital signature* yang ditambahkan ke dalam pesan. kali ini, digunakan kunci publik untuk mendekripsi *digital signature* ini. Setelah dilakukan proses dekripsi, maka didapatlah nilai hash asli dari pesan. kemudian, kedua nilai hash ini dibandingkan, jika sama, pesan yang diterima adalah otentik. Jika tidak, maka telah terjadi pengubahan pada pesan yang diterima.

Terdapat tiga kemungkinan jika hasil otentikasi tanda tangan digital tidak valid, yaitu:

- *Digital signature* nilainya telah diubah, sehingga ketika didekripsi didapat nilai hash yang berbeda
- Jika pesan telah diubah setelah ditambahkan *digital signature*, maka nilai hash pada langkah pertama akan berbeda dengan nilai hash hasil dekripsi *digital signature*
- Kemungkinan kesalahan ketiga adalah kunci publik yang digunakan untuk dekripsi dan kunci private yang digunakan untuk enkripsi tidak bersesuaian

Ketidak validan hasil otentikasi tidak semuanya dikarenakan *digital signature* / isi pesan telah diubah. Terkadang orang yang melakukan otentikasi salah menggunakan kunci publik dari kunci private yang digunakan untuk enkripsi. Kesalahan ini dapat ditangani dengan cara menggunakan sertifikat yang berisi informasi dari kunci publik seseorang. Setiap kunci publik yang dimiliki seseorang memiliki sertifikat sendiri-sendiri. Sertifikat inilah yang dikirim juga bersama dengan pesan tersebut dengan cara yang lebih aman.

E. SHA

SHA adalah fungsi hash satu arah yang dibuat NIST dan digunakan bersama DSS (*digital signature standard*). Algoritma ini menerima masukan berupa pesan dengan ukuran maksimum 2^{64} bit (2.147.483.648 gigabyte) dan menghasilkan messsage digest yang panjangnya 160 bit, lebih panjang dari message digest yang dihasilkan oleh MD5. Terdapat enam varian SHA, antara lain: SHA-0, SHA-1, SHA-224, SHA-256, SHA-384, dan SHA-512.

Langkah-langkah pembuatan message digest dengan SHA-1:

1. Penambahan bit-bit pengganjal (padding bits)
2. Penambahan nilai panjang pesan semula
3. Inisialisasi penyangga (buffer) MD
4. Pengolahan pesan dalam blok berukuran 512 bit

SHA membutuhkan 5 buah penyangga yang masing-masing panjangnya 32 bit. Total panjang penyangga adalah 5x32= 160 bit. Kelima penyangga ini diberi nama A, B, C, D, dan E. setiap penyangga diinisialsasi dengan nilai-nilai dalam notasi HEX. Proses H_{SHA} terdiri dari 80 buah putaran dengan menggunakan bilangan penambah K_t untuk tiap putaran.

F. RSA

Merupakan algoritma kunci-publik yang paling terkenal dan paling banyak aplikasinya. Algoritma ini ditemukan oleh tiga peneliti dari MIT yaitu, Ron Rivest, Adi Shamir, dan Len Adleman pada tahun 1976. Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima.

Berikut merupakan langkah yang diperlukan untuk membangkitkan sepasang kunci pada algoritma RSA:

1. Pilih dua bilangan prima, p dan q (rahasia)
2. Hitung n=pq
3. Hitung $\phi(n) = (p-1)(q-1)$
4. Pilih sebuah bilangan bulat e untuk kunci publik, e relatif prima terhadap $\phi(n)$
5. Hitung kunci dekripsi d dengan persamaan $ed \equiv 1 \pmod{\phi(n)}$ atau $d \equiv e^{-1} \bmod (\phi(n))$

hasil dari algoritma di atas adalah kunci publik yang merupakan pasangan (e, n) dan kunci privat yang merupakan pasangan (d, n).

Proses enkripsi dimulai dengan membagi byte plainteks menjadi blok-blok plainteks: m1, m, m3, … dengan syarat $0 < m_i < n-1$. Kemudian hitung blok cipherteks c_i untuk blok plainteks p_i dengan persamaan $c_i = m_i^e \bmod n$. Dalam hal ini e adalah kunci publik. Sedangkan untuk proses dekripsi dilakukan dengan menggunakan persamaan $m_i = c_i^d \bmod n$. dalam hal ini d adalah kunci privat.

Kekuatan algoritma RSA terletak pada tingkat kesulitan dalam memfaktorkan bilangan menjadi faktor-faktor prima, yang dalam hal ini n=ab. Sekali n berhasil difaktorkan menjadi a dan b, maka $\phi(n) = (a-1)(b-1)$ dapat dihitung. Selanjutnya, karena kunci enkripsi e diumumkan (tidak rahasia), maka kunci dekripsi d dapat dihitung dengan persamaan $ed \equiv 1 \pmod{\phi(n)}$.

III. Analisis dan Implementasi *Digital Signature* pada Aplikasi SMS Android

A. Analisis Kebutuhan

Aplikasi DigiSMS ini digunakan untuk mengirim dan menerima pesan yang telah ditambahkan *digital signature* melalui layanan SMS. Pesan yang akan dikirimkan melalui SMS terlebih dahulu dihitung *digital signature* nya dengan menggunakan kunci publik kemudian *digital signature* ini disisipkan di akhir pesan SMS. Untuk dapat melakukan pengecekan otentikasi pesan tersebut penerima harus memasukan pasangan kunci private yang bersesuaian.

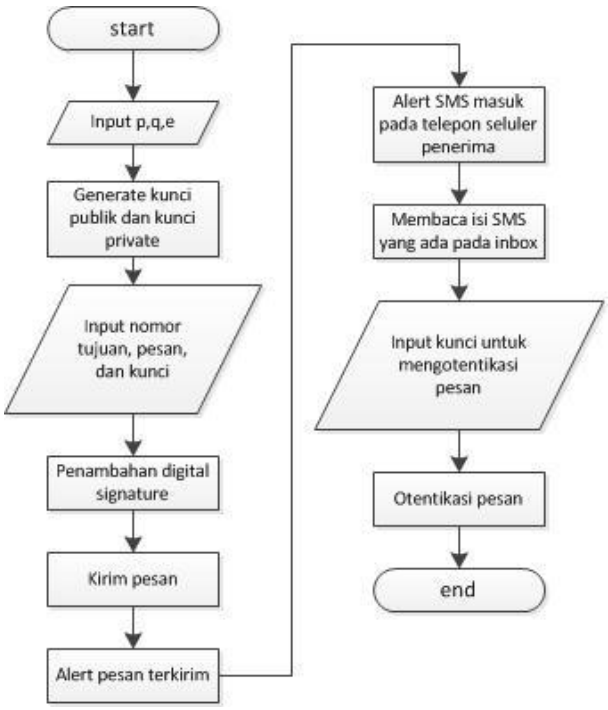
Dalam membangun aplikasi DigiSMS ini diperlukan batasan yang jelas agar aplikasi yang dibangun tidak keluar dari rencana awal. Beberapa kebutuhan sistem yang akan didefinisikan antara lain:

- Memiliki kemampuan untuk mengirimkan dan menerima pesan melalui SMS
- Memiliki kemampuan untuk membuat *digital signature* kemudian menambahkannya di akhir isi pesan
- Memiliki kemampuan untuk melakukan otentikasi isi pesan
- Memiliki kemampuan untuk membuat pasangan kunci publik dan kunci privat
- Memiliki kemampuan untuk membaca pesan masuk yang ada di dalam *inbox*

Karena aplikasi ini tergolong aplikasi sederhana maka tidak dibutuhkan suatu kondisi yang rumit. Berikut adalah gambaran sistem secara umum:

- Pengirim akan mengirim pesan melalui layanan SMS
- Pengirim meng-*generate* kunci publik dan kunci privat yang akan digunakan untuk membuat *digital signature* dan melakukan otentikasi isi pesan
- Pesan ini akan ditambahkan *digital signature* terlebih dahulu menggunakan kunci publik sebelum dikirim
- Pesan ini akan dikirim berupa pesan teks (SMS)
- Pesan ini nantinya akan diterima oleh penerima pesan
- Untuk melakukan otentikasi isi pesan dibutuhkan kunci privat yang bersesuaian

Cara kerja sistem ini akan dibagi ke dalam beberapa proses utama. Proses ini dibagi menjadi empat tahapan yaitu enkripsi pesan, pengiriman pesan, pembacaan pesan, dan dekripsi pesan. Untuk lebih jelasnya dapat dilihat pada gambar 4.



Gambar 4. Alur Proses

B. Komponen yang Digunakan

Implementasi aplikasi DigiSMS pada makalah ini dilakukan pada perangkat lunak Eclipse Helios dengan bahasa Java dan dibangun pada platform Android 2.2.

C. Implementasi

Implementasi aplikasi DigiSMS ini dibagi menjadi 9 kelas yaitu kelas Global, DigiSMS, DigiSign, MySHA, RSA, GenerateKey, SMSSender, SMSReader, dan SMSAuthenticate.

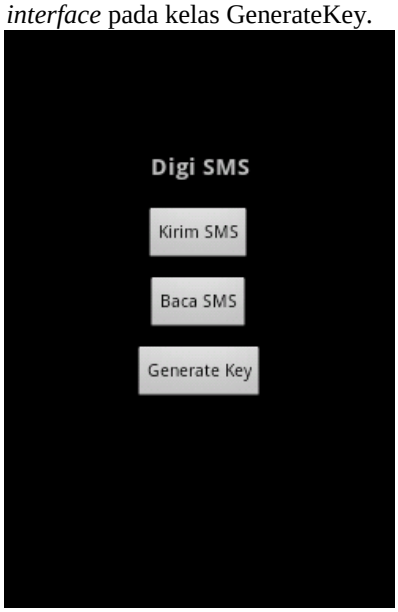
1. Kelas Global

Kelas ini hanya berfungsi untuk menyimpan variabel global. Terdiri dari 2 variabel static yang

bertipe String. Variabel static ini menyimpan String nomor pengirim SMS dan isi pesan yang akan dihasilkan pada SMSReader untuk diproses pada SMSAuthenticate.

2. Kelas DigiSMS

Pada kelas ini diimplementasikan *interface* untuk tampilan menu utama. *Interface* pada kelas ini dapat dilihat pada gambar 5. Pada menu utama ini terdapat 3 pilihan menu yaitu Kirim SMS, Baca SMS, dan Generate Key. Ketika tombol Kirim SMS ditekan maka akan memanggil *interface* yang ada pada kelas SMSSender dan apabila tombol Baca SMS ditekan maka akan memanggil *interface* yang ada pada kelas SMSReader. Tombol generate key berfungsi untuk memanggil *interface* pada kelas GenerateKey.



Gambar 5. *Interface* Menu Utama

3. Kelas DigiSign

Di dalam kelas ini terdapat 2 prosedur utama yaitu generateDS dan authenticate. Kedua prosedur ini dibuat dengan memanfaatkan algoritma SHA dan RSA. Berikut adalah psudocode untuk kedua prosedur tersebut:

```
    public String generateDS(String text, BigInteger e,
BigInteger n){
    MySHA sha = new MySHA();
    RSA rsa = new RSA();
    byte[] plain = text.getBytes();
    byte[] hashed = sha.messageDigester(plain);
    String hashString = sha.byteToHex(hashed);
    rsa.setE(e);
    rsa.setN(n);
    String ds=text+"\n-----
"+rsa.enkripsiRSA(hashString.getBytes());
    return ds;
}
```

Gambar 6. Pseudocode generateDS

```
    public boolean autenticate(String text, BigInteger d,
BigInteger n){
    String[] temp = text.split("\n-----");
    MySHA sha = new MySHA();
    RSA rsa = new RSA();
    byte[] plain = temp[0].getBytes();
    byte[] hashed = sha.messageDigester(plain);
    String hashString = sha.byteToHex(hashed);
    rsa.setD(d);
    rsa.setN(n);
    String ds =
rsa.ArrayByteToString(rsa.dekripsiRSA(temp[1]));
    return ds.equalsIgnoreCase(hashString);
}
```

Gambar 7. Pseudocode authenticate

4. Kelas MySHA

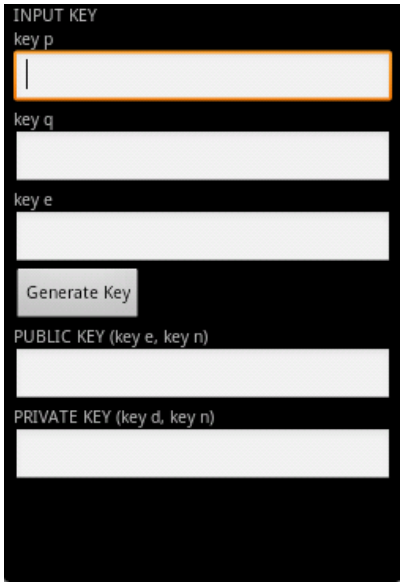
Kelas ini merupakan implementasi dari algoritma SHA yang digunakan untuk menghitung nilai hash dari isi pesan SMS.

5. Kelas RSA

Kelas ini merupakan implementasi dari algoritam RSA yang digunakan untuk melakukan enkripsi dan dekripsi dari message digest hasil fungsi hash pada isi pesan SMS.

6. Kelas GenerateKey

Kelas ini berfungsi untuk menghasilkan sepasang kunci publik dan kunci privat. Untuk dapat menghasilkan kunci publik dan privat dibutuhkan 3 angka yaitu, p, q, dan e. Nilai p dan q yang dimasukkan harus bilangan prima. Interface yang digunakan pada kelas ini dapat dilihat pada gambar 8.



Gambar 8. *Interface* Generate Key

7. Kelas SMSSender

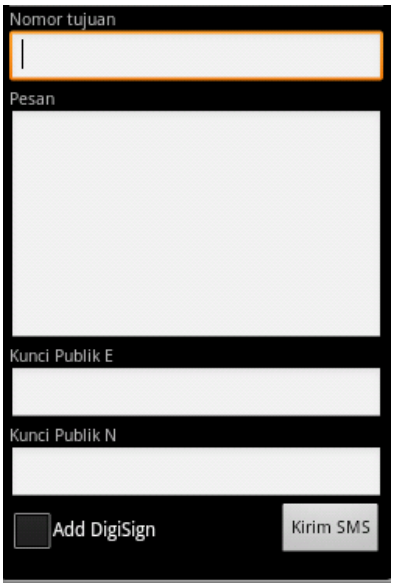
Kelas ini berfungsi untuk menampung input user berupa nomor tujuan, isi pesan, dan kunci. Kemudian dilakukan pembuatan *digital signature* dan penambahan *digital signature* tersebut di akhir pesan. Terakhir pesan dikirimkan melalui SMS. Pada kelas

ini terdapat prosedur utama untuk mengirim isi pesan melalui layanan SMS yaitu prosedur sendSMS. Pseudocode untuk prosedur ini dapat dilihat pada gambar 9.

```
    private void sendSMS(String phoneNumber, String
message)
    {
     String SENT = "SMS_SENT";
     String DELIVERED = "SMS_DELIVERED";
     PendingIntent sentPI =
PendingIntent.getBroadcast(this, 0,
            new Intent(SENT), 0);
     PendingIntent deliveredPI =
PendingIntent.getBroadcast(this, 0,
            new Intent(DELIVERED), 0);

     //---when the SMS has been sent---
     registerReceiver(new BroadcastReceiver(){
            @Override
            public void onReceive(Context arg0, Intent
arg1) {
                        switch (getResultCode())
                        {
                                    case
Activity.RESULT_OK:

     Toast.makeText(getBaseContext(), "SMS sent",
Toast.LENGTH_SHORT).show();
                                                break;
                                    case
SmsManager.RESULT_ERROR_GENERIC_FAILURE:

     Toast.makeText(getBaseContext(), "Generic failure",
Toast.LENGTH_SHORT).show();
                                                break;
                                    case
SmsManager.RESULT_ERROR_NO_SERVICE:

     Toast.makeText(getBaseContext(), "No service",
Toast.LENGTH_SHORT).show();
                                                break;
                                    case
SmsManager.RESULT_ERROR_NULL_PDU:

     Toast.makeText(getBaseContext(), "Null PDU",
Toast.LENGTH_SHORT).show();
                                                break;
                                    case
SmsManager.RESULT_ERROR_RADIO_OFF:

     Toast.makeText(getBaseContext(), "Radio off",
Toast.LENGTH_SHORT).show();
                                                break;
                        }
            }
     }, new IntentFilter(SENT));

     //---when the SMS has been delivered---
     registerReceiver(new BroadcastReceiver(){
            @Override
            public void onReceive(Context arg0,Intent
arg1) {
                        switch (getResultCode())
                        {
                                    case
Activity.RESULT_OK:

     Toast.makeText(getBaseContext(), "SMS delivered",
Toast.LENGTH_SHORT).show();
                                                break;
                                    case
Activity.RESULT_CANCELED:

     Toast.makeText(getBaseContext(), "SMS not
delivered", Toast.LENGTH_SHORT).show();
                                                break;
                        }
            }
     }, new IntentFilter(DELIVERED));
     SmsManager sms = SmsManager.getDefault();
     sms.sendTextMessage(phoneNumber, null, message,
sentPI, deliveredPI);
    }
```

Gambar 9. Pseudocode SendSMS

Interface yang ada pada kelas ini dapat dilihat pada gambar 10. Dapat dilihat terdapat 4 buah *textbox* yang menampung input pengguna. Input yang diterima antara lain nomor tujuan, isi pesan, dan kunci publik yang terdiri dari kunci e dan kunci n. Kunci publik ini akan digunakan untuk membuat *digital signature*. Secara *default* pesan belum ditambahkan *digital signature*, namun ketika *checkbox* add digisign dipilih maka secara otomatis *digital signature* akan ditambahkan di akhir pesan. Pesan akan dikirim melalui layanan SMS apabila tombol Kirim SMS ditekan. Sistem akan memberikan notifikasi ketika sms terkirim dan diterima.



Gambar 10. Interface untuk menulis pesan

8. Kelas SMSReader

Kelas ini berfungsi membaca semua pesan yang disimpan di dalam inbox telepon seluler kemudian menampilkannya di dalam aplikasi DigiSMS.

Pertama kali masuk pada tampilan inbox DigiSMS hanya terdapat tombol update SMS list. Ketika tombol ditekan maka sistem akan menampilkan pesan yang disimpan di inbox telepon seluler. Apabila salah satu isi pesan ditekan maka sistem akan memanggil *interface* untuk melakukan otentikasi isi pesan.

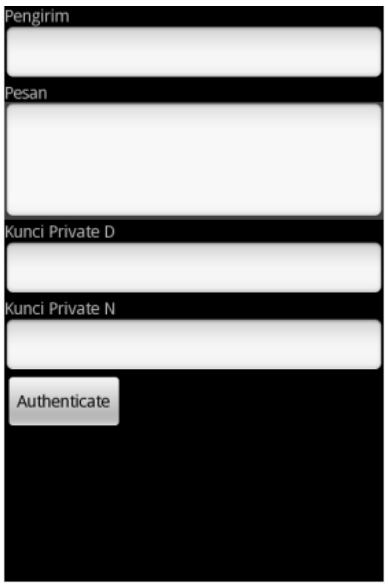
Di dalam kelas ini terdapat prosedur utama yang berfungsi untuk menampilkan list pesan yang ada di dalam inbox. Prosedur ini dapat dilihat pada gambar 11.

```
public void onClick( View v )
{
 ContentResolver contentResolver =
getContentResolver();
 Cursor cursor = contentResolver.query( Uri.parse(
"content://sms/inbox" ), null, null, null, null);
 int indexBody = cursor.getColumnIndex( "body");
 int indexAddr = cursor.getColumnIndex( "address" );
 if ( indexBody < 0 || !cursor.moveToFirst() ) return;
 smsList.clear();
 do
 {
        String str = "Sender: " + cursor.getString(
indexAddr ) + "\n" + cursor.getString( indexBody );
        smsList.add( str );
 }
 while( cursor.moveToNext() );
 ListView smsListView = (ListView) findViewById(
R.id.SMSList );
 smsListView.setAdapter( new ArrayAdapter<String>(
this, android.R.layout.simple_list_item_1, smsList) );
 smsListView.setOnItemClickListener( this );
}
```

Gambar 11. Pseudocode untuk menampilkan list pesan

9. Kelas SMSAuthenticate

Kelas ini berfungsi untuk melakukan otentikasi isi pesan yang ada di dalam inbox. Pertama-tama sistem meminta pengguna untuk memasukkan kunci private. Kemudian ketika pengguna menekan tombol authenticate maka akan dilakukan pengecekan apakah isi pesan otentik/ tidak ada perubahan kemudian akan muncul pesan pop up yang menunjukan apakah isi pesan otentik atau tidak. *Interface* pada kelas ini dapat dilihat pada gambar 12.



Gambar 12. Interface Dekripsi pesan

IV. PENGUJIAN

Pengujian dilakukan pada telepon seluler Android Samsung Galaxy GT-i9003 di dalamnya menggunakan firmware Android 2.3. Pengujian terdiri dari 5 bagian utama yaitu pengujian pembuatan kunci, penambahan *digital signature* dari isi pesan, pengiriman SMS, pembacaan SMS, dan otentikasi pesan. Pengujian ini dilakukan dengan mengirimkan pesan pada nomor sendiri, sehingga dapat dibaca kembali isi pesan yang diterima.

Selama dilakukan pengujian dengan kombinasi input berbeda-beda, tidak ditemukan masalah. Semua fungsi dapat berjalan dengan baik.

Berikut ini merupakan beberapa contoh pengujian yang dilakukan

Kombinasi kunci: p 223 q 137 e 115 Kunci publik (115, 30551) Kunci privat (26779, 30551)
Pesan + *digital signature*: besok ada rapat di gedung x pukul 13.00 WIB ----- 45ef 743e 16d9 6da7 2d77 16d9 64ca 582 9b3 16d9 3aed 3db5 691e 743e 64ca 582 465d 9b3 6910 382c 3db5 2d77 691e 2baa 691e 2d77 9b3 6da7 64ca 2baa 2e16 45ef 382c 382c 2e16 9b3 691e 743e 691e 2e16 5e84 2e16 16d9 743e

Untuk otentikasi, ketika dilakukan dengan memasukkan kunci privat yang tidak sesuai, maka diperoleh bahwa isi pesan tidak otentik. Hal yang sama berlaku ketika dilakukan perubahan pada isi pesan saja atau pada *digital signature* saja.

Kombinasi kunci:
p 17 q 7 e 5 Kunci publik (5, 119) Kunci privat (77, 119)
Pesan + *digital signature*: besok ada rapat di gedung x pukul 13.00 WIB ----- 45 61 10 e 37 10 49 64 2 10 2e 31 5c 61 49 64 66 2 11 67 31 37 5c 46 5c 37 2 e 49 46 32 45 67 67 32 2 5c 61 5c 32 13 32 10 61

Kunci publik yang berbeda akan menghasilkan *digital signature* yang berbeda walaupun isi pesannya sama. Dapat dilihat pada kombinasi kunci pertama digunakan kunci publik dan kunci privat yang relatif besar, hal ini berdampak *digital signature* yang diperoleh bertambah panjang. Jika dibandingkan dengan kombinasi kunci kedua yang memiliki nilai kunci lebih kecil, nampak *digital signature* yang lebih pendek. Hal ini disebabkan karena besarnya nilai n (hasil perkalian p dan q) membatasi nilai dari setiap blok-blok byte dari isi pesan. Semakin besar nilai p dan q, maka semakin besar pula nilai n sehingga nilai di setiap blok juga bertambah besar. Nilai di dalam blok ini kemudian akan dituliskan di dalam heksadesimal dan kemudian digabungkan dengan blok-blok lainnya untuk membentuk *digital signature*.

Dari beberapa pengujian, dapat dilihat bahwa *digital signature* yang dihasilkan pada DigiSign masih terhitung panjang, bahkan melebihi panjang isi pesannya. Hal ini menjadi kelemahan dari aplikasi ini karena semakin banyak karakter yang dituliskan di dalam pesan SMS semakin mahal pula biayanya.

V. Simpulan dan Saran

A. Simpulan

Digital signature dapat digunakan untuk mengatasi tiga aspek keamanan di dalam kriptografi yaitu: aspek otentikasi, keaslian pesan, dan anti-penyangkalan. Fungsi utama dari *digital signature* adalah menambahkan signatrue (tanda tangan) pada sebuah pesan sehingga pesan tersebut dapat diketahui keaslian identitas pengirim dan integritas isi pesan.

DigiSMS adalah aplikasi SMS yang dibangun untuk menangani aspkek kemanan kriptografi pada layanan SMS. Aplikasi ini dikembangkan dengan memanfaatkan algoritma SHA dan RSA. Aplikasi ini mampu berjalan dengan baik pada platform Android. Aplikasi ini mempunyai fungsi membuat kunci publik dan privat, menulis pesan, menambahkan *digital signature* pada akhir pesan, mengirim pesan melalui SMS, membaca pesan yang ada pada inbox telepon seluler, dan otentikasi isi pesan. DigiSMS memiliki kelemahan yaitu *digital signature* yang dihasilkan masih relatif panjang.

B. Saran

Untuk perbaikan dan pengembangan aplikasi DigiSMS lebih lanjut disarankan sebagai berikut:

- Menggunakan algoritma *digital signature* yang lebih efektif daripada kombinasi antara algoritma SHA+RSA, misalnya algoritma ECDSA (Eliptic Curve *Digital signature* Algorithm). Algoritma ini mampu menghasilkan *digital signature* yang lebih singkat
- Memanfaatkan algoritma Hoffman untuk mengkompresi *digital signature* sehingga *digital signature* yang dihasilkan lebih singkat
- Aplikasi dapat menyimpan SMS terkirim dan SMS yang belum dikirim
- Aplikasi menampilkan isi SMS masuk dan keluar dalam bentuk *thread* seperti aplikasi SMS bawaan Android
- Aplikasi didesign dengan interface yang menarik

Daftar Pustaka

[1] Rinaldi Munir, Situs Perkuliahan Kriptografi, http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2010-2011/kripto10-11.htm

[2] Nazruddin Safaat, "Android, Pemrograman Aplikasi Mobile Smartphone dan tablet PC Berbasis Android", Bandung: Penerbit Informatika, 2009.

[3] http://mobiforge.com/developing/story/sms-messaging-android

[4] http://www.apriorit.com/our-company/dev-blog/227-handle-sms-on-android

[5] http://gabohong.blogspot.com/2012/01/inilah-sejarah-sms-pertamakali-di-dunia.html

[6] http://octianaeni.blogspot.com/2011/11/pengenalan-android.html

[7] http://id.wikipedia.org/wiki/Kriptografi

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, 14 Mei 2012

Andi Kurniawan Dwi P.
13508028