Aplikasi Pengaman Pertukaran SMS pada Perangkat Android dengan Metode RSA

Hendra[1], Sukiman[2]
Teknik Informatika, STMIK IBBI
Jl. Sei Deli No. 18 Medan, Indonesia
hendra.soewarno@gmail.com[1], sukiman_liu@yahoo.com[2]

Abstrak

Pengiriman pesan dengan menggunakan Short Message Service (SMS) merupakan suatu layanan yang sangat populer dikalangan pemakaian perangkat bergerak di Indonesia. Pengiriman SMS dari satu perangkat ke perangkat lain membutuhkan perantara SMS Center (SMSC) untuk menyimpan dan menyampaikan SMS ke tujuan ketika perangkat tujuan tersedia. Teknologi SMS sendirinya memiliki kelemahan yaitu yaitu enkripsi hanya dilakukan antara Mobile Station (MS) dan Base Transceiver Station (BTS) sedangkan pada bagian lain terbuka sama sekali, sehingga memungkinkan serangan berupa penyadapan maupun modifikasi. Metode RSA memiliki keunggulan yaitu tidak membutuhkan pertukaran secret key dan dapat menjaga keaslian dari pengirim SMS. Aplikasi metode RSA dengan ukuran key 688 bit dapat diaplikasikan agar pertukaran kunci publik dapat dilakukan melalui sms, dan membawa konsekuensi berkurangnya jumlah karakter yang dapat dikirim per-sms menjadi 86 karakter.

1. Pendahuluan

SMS merupakan salah satu layanan yang populer pada telepon bergerak, dan digunakan oleh berbagai kalangan baik hanya sekedar untuk membuat pesan pribadi maupun oleh institusi bisnis untuk melakukan transaksi seperti pemesanan barang, konfirmasi pengiriman, sampai kepada konfirmasi pembayaran. Layanan SMS juga digunakan pada transaksi perbankan yaitu digunakan untuk pengiriman informasi saldo dan PIN untuk transaksi e-Banking.

SMS sendirinya memiliki berbagai kelemahan yaitu SMS dibangun dengan sistem dan program yang sama, dan SMS sendiri bisa melakukan roaming jaringan setempat hingga ke jaringan asing sehingga dimungkinkan SMS spoofing dalam bentuk penyamaran atau manipulasi informasi seperi alamat atau data lainnya yang menyerupai user pada umumnya. Kelemahan dari SMS lainnya adalah isi SMS yang dikirim terbuka di sistim penyedia jasa maupun pegawainya.

Penelitian ini akan mengaplikasi metode enkripsi dengan algoritma RSA untuk meningkatkan aspek *confidentiality*, *integrity, authenticity* dan *non-repudiation* pada perangkat mobile berbasis Android.

Secara umum, tujuan dari penelitian ini adalah mendapatkan suatu rancangan aplikasi pengirim dan penerima SMS yang meningkatkan kesulitan bagi cracker untuk membaca, mengubah maupun membuat SMS spoofing pada lintasan jalur pengiriman sms antara dua perangkat berbasis Android.

Adapun pembatasan masalah yang dilakukan pada rancangan aplikasi adalah aplikasi hanya beroperasi pada Android 2.2 keatas.

2. Metode

Android

Android adalah sebuah tumpukan software untuk peralatan bergerak yang terdiri dari sistim operasi, middleware, dan aplikasi kunci lainnya. Platform Android merupakan produk dari Open Handset Alliance (OHA) yang merupakan suatu kelompok organisasi yang berkolaborasi untuk membangun mobile phone yang lebih baik. Kelompok ini dipimpin oleh Google, operator mobile, pabrikan, pabrikan komponen, dan software provider, serta perusahaan marketing[1]

Arsitektur SMS

Suatu pesan sms dikirim melalui kanal sinyal GSM antara *Mobile Station* (MS) dan *Base Transceiver Station* (BTS). Pesan ini mengalir seperti panggilan normalnya, tetapi mereka diarahkan dari MSC ke suatu *Short Message Service Center* (SMSC). SMSC akan menyimpan pesan tersebut sampai dapat dikirim kepada penerima ataupun sampai kepada validitas waktu dari pesan telah terlewati. Penerima dari pesan ini dapat berupa pemakai MS maupun SMS gateway. SMS gateway adalah suatu server yang terkoneksi kepada satu atau lebih SMSC untuk menyediakan aplikasi SMS kepada pemakai MS.

Keamanan SMS

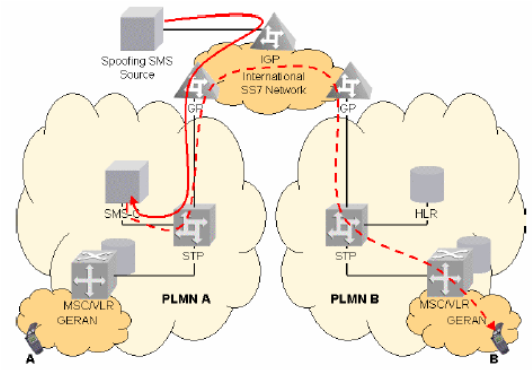
Dua aspek penting dari suatu entitas yang menggunakan teknologi SMS untuk pertukaran data pribadi maupun bisnis adalah sebagai berikut:

1. SMS bukanlah suatu lingkungan yang aman.
2. Pelanggaran keamanan sering terjadi lebih mudah pada konsentrasi terhadap manusia dari pada teknologi.[3]

Isi dari pesan SMS adalah tidak terenkripsi pada sistim operator, sehingga karyawan maupun penyusup pada sistim operator dapat dengan gampang melakukan penyadapan maupun perubahan, sehingga SMS bukankah suatu teknologi yang cocok untuk komunikasi yang membutuhkan keamanan. Kebanyakan pemakai tidak menyadari bahwa begitu mudahnya pembacaan isi SMS oleh karyawan operator.

Serangan pada SMS

Koneksi antara SMSC dan SMS gateway adalah bukan merupakan bagian dari standard GSM, dimana dimungkinkan pemakaian protokol seperti TCP/IP dan X.25 untuk komunikasi antara SMSC dan gateway, kemudian koneksi tersebut juga bukanlah merupakan bagian dari jaringan GSM, tetapi merupakan jaringan antara operator dengan penyedia konten, atau bahkan dengan internet. Keamanan pada jaringan ini sangat longgar proteksinya, sehingga dapat dengan gampang menjadi sasaran cracker karena isi dari pesan dikirim dalam bentuk plaintext, walaupun pada gateway telah memiliki authentikasi, tetapi kebanyakan menggunakan protokol dengan plaintext yang mengandung informasi nama login dan password. Serangan *man-in-the-middle attack* dapat saja dilakukan walaupun tidak gampang dilakukan karena biasanya gateway dan SMSC berada dibelakang firewall. Permasalahan lainnya yang perlu diketahui adalah dimungkinkan penyerang untuk membuat suatu gateway dimana dia berpura-pura menjadi gateway yang sebenarnya. Gateway palsu ini kemudian dapat mengirim segala jenis sms kepada pemakai MS melalui SMSC tersebut dengan memanipulasi informasi pengirim pesan pada field tertentu pada SMSC protokol, dengan teknik spoofing ini dimungkinkan untuk memanipulasi Mobile Station International ISDN Number (MSISDN) sehingga seakan-akan pesan tersebut datang dari telepon bergerak tertentu. Keberhasilan teknik ini tergantung kepada implementasi pada SMSC, karena SMSC dapat juga cukup cerdas untuk memeriksa keabsahan pengirim dan memblokir pesan tersebut, karena berasal dari suatu gateway yang bukan merupakan suatu telepon bergerak, dan hal ini penyerang harus membuat suatu SMSC simulator yang berpura-pura menjadi suatu SMSC yang sebenarnya sebagaimana ditunjukan Gambar 2.



Gambar 2, Teknik Spoofing SMS[6]

Pada gambar 2 dapat dijelaskan bahwa sms dikirim kepada SMS-C dengan memanipulasi MSISDN seakan-akan berasal dari pemakai akhir dari PLMN A, dan mengirim SMS kepada pemakai akhir pada PLMN B.

Algoritma RSA

Algoritma RSA mengambil nama dari Ron Rivest, Adi Shamir dan Len Adleman yang menciptakan metode ini pada tahuin 1977. Teknik dasarnya ditemukan pertama kali pada tahun 1973 oleh Clifford Cock dari CESG (bagian dari British GCHQ) tetapi dirahasiakan sampai tahun 1977. Paten dimiliki oleh RSA Labs dan telah expired[4].

RSA digunakan untuk enkripsi suatu berita tanpa membutuhkan pertukaran *secret key*. Alis dapat mengirim pesan tersandi kepada Bob tanpa perlu pertukaran *secret key* sebelumnya. Alis cukup menggunakan kunci publik milik Bob untuk proses enkripsi berita menjadi *ciphertext*, dan Bob menggunakan kunci privat yang hanya diketahuinya untuk proses dekripsi *ciphertext* tersebut menjadi *plaintext*. RSA juga dapat digunakan untuk suatu tandatangan digital pada berita, dimana Alis menandatangani pesan dengan menggunakan kunci privat-nya dan Bob dapat memeriksanya dengan menggunakan kunci publik milik Alis[4].

Berikut ini adalah algoritma untuk pembuatan key:

1. Hasilkan dua bilangan prima besar secara random, p dan q, dengan ukuran kira-kira sama seperti bahwa produk mereka n = pq adalah panjang bit yang diperlukan, misalnya 1024 bit.
2. Hitung n = pq dan (phi) φ = (p-1)(q-1).
3. Pilih sebuah bilangan bulat e, 1 < e < phi, dimana gcd(e, phi) = 1.
4. Hitung eksponen rahasia d, 1 < d < phi, sehingga ed ≡ 1 (mod phi).
5. Kunci publik adalah (n, e) dan kunci private (d, p, q). Simpan semua nilai dari d, p, q dan phi secara rahasia, n dikenal sebagai modulus, e dikenal sebagai eksponen public, dan d dikenal sebagai eksponen rahasia atau eksponen dekripsi.[5]

Enkripsi

Misalnya Alis ingin mengirim sebuah pesan m kepada Bob. Alis membuat ciphertext c dengan eksponensialisasi: c = me mod n, dimana e dan n adalah kunci publik dari Bob. Dia mengirim c kepada Bob. Untuk melakukan dekripsi, Bob juga melakukan eksponensialisasi: m = cd mod n; keterkaitan antara e dan d adalah memastikan bahwa Bob dengan benar melakukan recovery m. Karena hanya Bob mengetahui d, maka hanya Bob yang dapat melakukan dekripsi terhadap pesan tersebut[5].

Tanda Tangan Digital

Misalnya Alis ingin mengirim sebuah pesan m kepada Bob dengan cara dimana Bob bisa diyakinkan bahwa pesan tersebut adalah asli dalam keadaan tidak mengalami perubahan, dan benar adalah dari Alis. Alis membuat suatu tandatangan digital s dengan eksponensialisasi: s = md mod n, dimana d dan n adalah kunci privat dari Alis. Dia mengirim m dan s kepada Bob. Untuk menverifikasi tandatangan ini, Bob melakukan eksponensialisasi dan memeriksa bahwa pesan m berhasil direcover: m = se mod n, dimana e dan n adalah kunci publik milik Alis[5].

Keamanan RSA

Pada tahun 2003, RSA Security melakukan klaim bahwa ukuran key 1024-bit adalah setara dengan kunci simetris 80-bit, dan diperkirakan dapat dipecahkan pada antara waktu tahun 2006 dan 2010, dan ukuran key 2048 adalah cukup sampai tahun 2030. Arjen Lesntra dan Eric Verheul's memproyeksikan bahwa pada tahun 2009 suatu mesin dengan harga $250 juta dapat melakukan faktorisasi 1024-bit kunci RSA dalam waktu satu hari, sehingga suatu mesin dengan harga $10 juta dapat melakukan hal yang sama dalam waktu kurang dari satu bulan[6].

Base64-encoding

Permasalahan penyimpan kunci publik dan kunci privat maupun pesan hasil enkripsi sebagai data biner dan harus berurusan dengan masalah *trailing zero* maupun karakter ASCII yang tidak dapat dicetak, sehingga anda perlu mengkonversikannya ke teks string [6]. Teks string adalah lebih mudah ditangani jika anda perlu mengirim ataupun menyimpan informasi tersebut dalam format teks. Suatu base64-encoding dapat digunakan untuk memecahkan masalah *trailing zero* maupun karakter ASCII yang tidak dapat dicetak[7]. Panjang dari suatu string hasil encoding base64 dapat dihitung dengan formula 1.

Base64 = (Bytes + 2 - ((Bytes + 2) MOD 3)) / 3 * 4 ….....1)

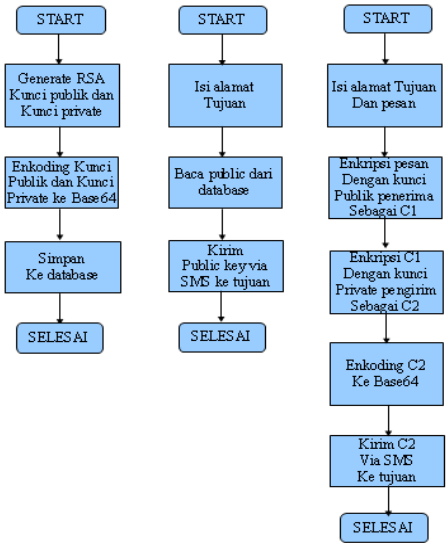
3. Diskusi

Algoritma

Secara umum aplikasi yang dirancang terdiri dari 5 komponen utama yaitu:

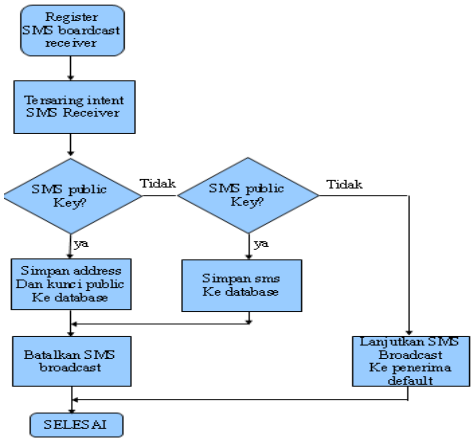
1. Activity untuk generate kunci publik dan kunci privat pemakai dengan algoritma RSA.
2. Activity untuk mengirim kunci publik pemakai kepada alamat target tujuan dengan menggunakan sms.

3. Activity untuk mengirim sms terenkripsi ke alamat tujuan.
4. Broadcast receiver yang berfungsi melakukan penerimaan kunci publik maupun sms terenkripsi dan menyimpan ke database
5. Activity yang berfungsi menampilkan plaintext dari sms terenkripsi.



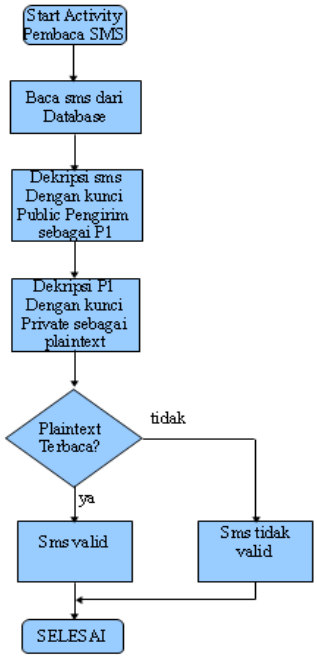
Gambar 3, Algoritma generate key dan pertukaran key

Awalnya pemakai perlu men-generate sepasang kunci menurut algoritma RSA yaitu kunci publik dan kunci privat. Kedua kunci hasil generate akan disimpan pada suatu tabel. Kemudian pemakai perlu melakukan pertukaran kunci publik dengan pihak dimana pertukaran sms rahasia perlu dilakukan, pertukaran kunci ini hanya perlu dilakukan sekali saja. Setelah pertukaran kunci publik berhasil dilakukan, maka pengiriman sms rahasia dapat dilakukan, sms plaintext akan dienkripsi dengan kunci publik penerima sebagai C1, kemudian C1 akan dienkripsi sekali lagi dengan menggunakan kunci private pengirim sms menjadi C2, dan akhirnya C2 inilah yang akan dikirim kepenerima. Algoritma dari ketiga proses tersebut ditunjukan pada Gambar 3.



Gambar 4 Algoritma penerimaan sms kunci publik dan sms rahasia

Penerimaan sms pada Android dilakukan melalui suatu Broadcast Receiver yang dilengkapi dengan intent filter untuk menyaring broadcast yang dipancarkan oleh sistim dalam bentuk intent. Broadcast receiver tersebut akan memfilter intent dengan action android.provider.Telephony.SMS_RECEIVED dan membaca data sms, serta melakukan pemeriksaan apakah sms tersebut adalah suatu sms yang berisi kunci publik dengan kriteria sms diawali dengan tanda @ dan memiliki panjang 160 (159 + 1) Karakter, jika tidak maka akan diperiksa apakah merupakan sms rahasia berdasarkan alamat pengirim yang dicocokan dengan tabel TheirKey yang berisi kunci publik dan alamat pemiliknya, jika ada maka sms tersebut akan dikategorikan sebagai sms rahasia, dan disimpan ke tabel Sms dan broadcast tersebut akan dibatalkan, dan jika bukan sms rahasia maka broadcast tersebut akan dilanjutkan ke broadcast receiver aplikasi sms bawaan dari perangkat android. Algoritma dari proses ini ditunjukan pada Gambar 4.



Gambar 5 Algoritma tampil sms rahasia

Struktur Database

Penyimpanan data kunci publik dan privat, kunci publik dari hasil pertukaran key, maupun sms terenkripsi menggunakan database SQLite3 yang sudah tersedia pada platform Android, adapun struktur data masing-masing tabel adalah disajikan pada Tabel 1, Tabel 2 dan Tabel 3.

Tabel 1, Struktur tabel MyKey

Nama field	Tipe Data	Keterangan
id	Integer	PK, autoincr
PublicKey	Text	
PrivateKey	Text	

Tabel 2, Struktur tabel TheirKey

Nama field	Tipe Data	Keterangan
id	Integer	PK, autoincr
Nomor	Text	
PublicKey	Text	

Tabel 3, Struktur tabel SMS

Nama field	Tipe Data	Keterangan
id	Integer	PK, autoincr
Jenis	Integer	0=sent, 1=received
Nomor	Text	
Waktu	Integer	long timestamp
Pesan	Text	
Baca	Integer	0=unread, 1 = read

Class dan Metoda

Class MyCrypto merupakan class yang berisi beberapa konstanta dan static method yang nantinya akan digunakan untuk mengenerate kunci publik dan kunci private RSA, mengembalikan kunci tersebut dalam format base64-encoded, melakukan proses enkripsi dan dekripsi berdasarkan kunci privat dan kunci publik yang dapat ditentukan oleh pemakai.

```
public class MyCrypto {
 private static final String CIPHER_ALGORITHM = "RSA";
 public static final int RANDOM_KEY_SIZE = 688;
 private static KeyPair keyPair;
 private static Cipher cipher;
 private static PrivateKey privateKey;
 private static PublicKey publicKey;
public static void generateKey() throws Exception {
 KeyPairGenerator kpg =
KeyPairGenerator.getInstance(CIPHER_ALGORITHM);
                    kpg.initialize(RANDOM_KEY_SIZE);
 cipher = Cipher.getInstance(CIPHER_ALGORITHM);
             keyPair = kpg.generateKeyPair();
             privateKey = keyPair.getPrivate();
             publicKey = keyPair.getPublic();
 }
public static String encrypt(String plaintext) throws Exception {
 cipher = Cipher.getInstance(CIPHER_ALGORITHM);
 cipher.init(Cipher.ENCRYPT_MODE,publicKey);
 byte[] bytes = plaintext.getBytes();
 byte[] encrypted1 = cipher.doFinal(bytes);
 cipher.init(Cipher.ENCRYPT_MODE,privateKey);
 byte[] encrypted2 = cipher.doFinal(encrypted1);
             String encryptedString =  Base64.encodeToString(encryp-
ted2,Base64.DEFAULT);
return encryptedString;
}
public static String decrypt(String encrypted)  throws Exception {
 cipher = Cipher.getInstance(CIPHER_ALGORITHM);
 cipher.init(Cipher.DECRYPT_MODE, publicKey);
 byte[] bytes = Base64.decode(encrypted, Base64.DEFAULT);
 byte[] decrypted1 = cipher.doFinal(bytes);
 cipher.init(Cipher.DECRYPT_MODE, privateKey);
 byte[] decrypted2 = cipher.doFinal(decrypted1);
 return new String(decrypted2);
 }
public static void setPrivateKeyFromString(String key) throws
Exception  {
  KeyFactory keyFactory;
    keyFactory = KeyFactory.getInstance(CIPHER_ALGORITHM);
    EncodedKeySpec privateKeySpec = new
PKCS8EncodedKeySpec(Base64.decode(key.getBytes(),Base64.DE-
FAULT));
    privateKey = keyFactory.generatePrivate(privateKeySpec);
 }
public static void setPublicKeyFromString(String key)  hrows Ex-
ception {
  KeyFactory keyFactory;
```

```
  keyFactory = KeyFactory.getInstance(CIPHER_ALGORITHM);
      EncodedKeySpec publicKeySpec = new
X509EncodedKeySpec(Base64.decode(key.getBytes(),Base64.DE-
FAULT));
  publicKey = keyFactory.generatePublic(publicKeySpec);

 }
Public static String getPrivateKey() throws UnsupportedEn-
codingException {
  byte[] encoded = Base64.encode(privateKey.getEncoded(),
Base64.DEFAULT);
  String string = new String(encoded,"UTF-8");
  return string;
 }
public static String getPublicKey() throws UnsupportedEn-
codingException {
  byte[] encoded = Base64.encode(publicKey.getEncoded(),
Base64.DEFAULT);
  String string = new String(encoded,"UTF-8");
  return string;
  }
}
```

4. Hasil

Ukuran Kunci RSA dan ukuran SMS

Pada rancangan ini pertukaran kunci publik akan dilakukan melalui sms, maka perlu memperhatikan keterbatasan jumlah teks yang dapat dikirim suatu sms adalah 160 byte, sehingga perlu dipilih ukuran kunci RSA yang menghasilkan kunci public dalam format base64-encoded yang mendekati batasan tersebut. Pengujian ukuran karakter kunci publik dan kunci privat dalam format base64-encoded berdasarkan berbagai ukuran kunci RSA dengan hasil yang disajikan pada Tabel 4.

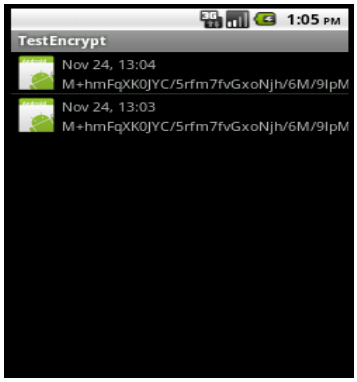
Tabel 4, Ukuran kunci publik dan privat dari berbagai ukuran key RSA

Ukuran kunci	Base64-encoded	
	Kunci publik	Kunci private
512	130	471
640	150	564
672	154	588
688	**159**	**600**
768	171	661
1024	219	860

Berdasarkan penyajian tersebut diatas, maka panjang key RSA yang dipilih adalah 688 bit, dan karena RSA bekerja sebagai block cipher, maka panjang sms yang dapat dienkripsi per-blok (688-bit/8) adalah 86 karakter

Pembacaan SMS Rahasia

Proses pembacaan SMS Rahasia ditunjukan pada Gambar 11 dan Gambar 12.



Gambar 11. Daftar SMS rahasia

Gambar 12. Plaintext SMS rahasia

5. Kesimpulan

Berdasarkan hasil pengujian yang dilakukan maka diekspektasikan aplikasi hasil rancangan dapat meningkatkan keamanan sms dari aspek authentikasi, integritas dan non-repudiation, karena plaintext sms sebelum dikirim akan dienkripsi dengan kunci publik penerima sebagai ciphertext1, dan ciphertext1 akan dienkripsi sekali lagi dengan kunci privat pengirim menjadi ciphertext2. Sesuai dengan kemampuan dari metode RSA, jika seorang cracker melakukan penyadapan dan ingin membaca ataupun melakukan perubahan isi pesan, maka dia perlu memecahkan kunci privat penerima sms rahasia tersebut. Jika seorang cracker ingin mengirim SMS spoofing, maka dia perlu memecahkan kunci private pengirim sms rahasia tersebut.

6. Saran

Aplikasi hasil rancangan dibatasi pada RSA dengan ukuran kunci 688 bit untuk memenuhi batasan pengiriman kunci publik dalam format base64-encoded sebesar 160 karakter per-sms, untuk pemakaian yang membutuhkan tingkat keamanan yang lebih tinggi (pada tahun 2010 RSA Laboratories menyarankan ukuran key minimum adalah 1024, dan 2048 untuk tahun 2030), maka aplikasi perlu dipertimbangkan dan pengembangan aplikasi yang memungkinkan untuk mengirim kunci publik dan sms rahasia yang menggunakan lebih dari satu sms.

Daftar Pustaka

[1] Developer Android, What is Android, http://developer.android.com/guide/basics/what-is-android.html, November 2011.

[2] Network Security Solution (NSS), SMS Vulnerabilities – XMS Technology White Paper, February 2006.

[3] Nick Jones, Don't Use SMS for Confidential Communication, Gartner, Inc., 2002.

[4] David Ireland, RSA Algorithm, http://www.di-mgt.com.au/rsa_alg.html, DI Management.

[5]. R. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, 21 (2), pp. 120-126, February 1978.

[6]. Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology* 14(4):255–293, 2001.

[7] Obiviex, How to Calculate the Size of Encrypted Data?, http://www.obviex.com/Articles/CiphertextSize.aspx