ISSN: 2541-1004
e-ISSN: 2622-4615

Integrasi Algoritma RSA (Rivest Shamir Adleman) dan Caesar Cipher untuk Meningkatkan Keamanan Enkripsi SMS (Short Message Service)

Yosanny Ferdino Rizkyansyah[1], dan Aries Saifudin[2]

[1,2]Teknik Informatika, Universitas Pamulang, Tangerang Selatan, Indonesia
e-mail: {[1]yosanny.rizkyansyah, [2]aries.saifudin}@gmail.com

Abstract

SMS (Short Message Service) is used because it is cheap, easily accessible and a popular method to sent information by a text message using a mobile phone. However, the original SMS application on android phones are have no security features. A message sent is still in plain text form so easily be intercepted without unknown by the sender or the recipient. Therefore, proposed encryption algorithm RSA (Rivest Shamir Adleman) and Caesar Cipher to secure a text message sent. The results showed that the application can prevents the contents of SMS to intercepted or viewed by unauthorized people because the message that is sent is not in plain text, but the message has been encrypted (cipher text).

Keywords: Encription, SMS (Short Message Service), RSA (Rivest Shamir Adleman), Caesar Cipher

1. Introduction

Telepon pintar (*smartphone*) berbasis android banyak digunakan dalam berbagai aktivitas karena aplikasi dan fitur yang dimiliki terus dikembangkan menggunakan lisensi open source. SMS (*Short Message Service*) merupakan salah satu fitur dari telepon pintar yang digunakan untuk berkomunikasi dengan cara mengirimkan teks (Alvianto & Darmaji, 2015, p. 1). Fitur SMS banyak digunakan karena dapat diakses dengan mudah dengan biaya yang rendah, sehingga menjadi salah satu cara komunikasi yang populer untuk mengirim pesan dalam bentuk teks (Pereira, Santos, & Olieveira, 2013, p. 698). Hal yang paling terpenting dalam melakukan pertukaran informasi pesan teks dengan menggunakan SMS (*Short Message Service*) adalah keamanannya (Saxena & Chaudhari, 2014, p. 138). SMS asli atau bawaan telepon selular berbasis android tidak memperhitungkan fitur keamanan dalam melakukan pertukaran informasi pesan teks (Castiglione, Cattaneo, Cembalo, & Santis, 2012, p. 771). Layanan SMS bawaan telepon seluler berbasis android ini dirancang tanpa adanya keamanan pesan (Sagheer, Abdulmunem, & Abduljabbar, 2013, p. 281). Pesan SMS yang dikirimkan masih berupa pesan terbuka atau plain text. Sebelum pesan ini diterima oleh penerima, pesan akan dikirim ke pusat SMS atau dikenal sebagai SMSC (*Short Message Service Center*), pesan yang masuk ke SMSC akan disimpan di dalam pusat SMS maka pesan yang masih bersifat terbuka dapat dengan mudah dibaca oleh operator pusat SMS tersebut (Agoyi & Seral, 2010, p. 448).

Dengan demikian dibutuhkan suatu metode yang dapat menjaga kerahasiaan pesan teks yang dikirim. Salah satu metode untuk menjaga kerahasiaan/keamanan pesan adalah dengan cara melakukan enkripsi pesan teks (SMS) yang dikirimkan. Enkripsi ini bertujuan agar pertukaran informasi melalui aplikasi SMS yang terdapat pada telepon seluler berbasis android terjamin keamanannya (Patil, Sahu, & Jain, 2014, p. 2). Enkripsi ditujukan untuk menyamarkan pesan dan informasi yang terdapat di dalamnya agar tidak terbaca oleh pihak atau orang yang tidak diharapkan dapat membaca pesan tersebut. Dalam penelitian ini untuk mengenkripsi pesan SMS menggunakan suatu metode yaitu menggunakan algoritma RSA (*Rivest Shamir Adlemen*) dan algoritma *Caesar Cipher.*

Dalam penilitian ini menggunakan algoritma RSA dan *Caesar cipher.* Algoritma RSA dipilih karena keamanannya terletak pada sulitnya pemfaktoran bilangan prima, semakin besar jumlah bilangan prima dalam pembangkitan kunci maka semakin sulit untuk dipecahkan (Nagar & Alshamma, 2012, p. 639). Pemfaktoran dilakukan untuk memperoleh kunci untuk membuka pesan yang telah terenkripsi. Selama pemfaktoran bilangan prima belum ditemukan dan belum ada algoritma yang bisa memecahkan pemfaktoran bilangan prima maka keamanan algoritma RSA akan tetap terjamin (Zhao, Yang, Zhou, & Wei, 2010, p. 640). Dan algoritma *Caesar Cipher* dipilih

karena algoritma ini sederhana dan tidak membutuhkan waktu yang lama dalam penghitungan enkripsi pesan teks (Senthil, Prasanthi, & Rajaram, 2013, p. 1). Algoritma *Caesar Cipher* sederhana karena dalam enkripsi pesan dilakukan dengan cara pergeseran terhadap pesan sebanyak kunci yang diberikan (Han & Mahyuddin, 2014, p. 112). Pada penelitian ini kunci pada algoritma *Caesar Cipher* juga diperbanyak menjadi 256 kunci sehingga karakter yang terbentuk pada *cipher text* lebih beraneka ragam.

Kelemahan dari algoritma RSA (*Rivest Shamir Adleman*) apabila bilangan prima sebagai pembangkit kuncinya dapat ditemukan atau dipecahkan maka akan ada celah untuk membuka hasil enkripsi pesan teks (Dent & Mitchell, 2005, p. 58). Untuk mengurangi risiko apabila bilangan prima sebagai pembangkit kuncinya ditemukan atau dipecahkan maka diusulkan hasil dari pesan teks yang telah dienkripsi oleh algoritma RSA akan dienkripsi kembali oleh algoritma *Caesar Cipher,* jadi akan memberikan keamanan yang berlapis dari algoritma RSA dan algoritma *Caesar Cipher* untuk lebih mempersulit pemecahan *cipher text* menjadi *plain text* pada pesan teks yang dikirimkan melalui aplikasi SMS yang berada pada telepon seluler berbasis android.

2. Penelitian Terkait

Pada penelitian Alvianto dan Darmaji (Alvianto & Darmaji, 2015, pp. 1-6) menjelaskan bahwa SMS merupakan salah satu fitur telepon pintar yang banyak digunakan oleh masyarakat. Walaupun banyak diminati masyarakat, SMS memiliki kelemahan pada sisi keamanan/kerahasiaan. Salah satu solusi yang dapat dikembangkan untuk mengatasi masalah kemanan/kerahasiaan pesan adalah dengan enkripsi. Dengan mengirimkan pesan SMS yang telah dienkripsi, maka kerahasiaan/keamanan pesan SMS menjadi lebih baik dan aman. Pada penelitian ini dilakukan enkripsi untuk pengamanan pesan khususnya SMS menggunakan teknik kriptografi dengan algoritma RSA (*Rivest Shamir Adleman*). Pesan yang dikirimkan akan terenkripsi terlebih dahulu dengan algoritma RSA lalu dikirimkan ke penerima pesan. Dan penerima pesan untuk membukanya dengan cara dekripsi pesan. Hasil penelitian ini menunjukkan waktu dalam memproses enkripsi dan dekripsi berdasarkan jumlah karakter yang akan dikirim melalui proses enkripsi dan diterima melalui proses dekripsi menggunakan algoritma RSA. Rata-rata *response time* untuk melakukan enkripsi tiap karakter yaitu 14,925 milidetik, sedangkan untuk melakukan dekripsi per karakter rata-rata membutuhkan *response time* sebesar 4,679 milidetik. Dengan penyamarkan pesan SMS menjadi *cipher text* yang merupakan hasil enkripsi menggunakan Algoritma RSA dapat mencegah pihak lain untuk membaca isi pesan SMS sebenarnya.

Pada penelitian Fahrianto, Masruroh, dan Ando (Fahrianto, Masruroh, & Ando, 2014, pp. 31-34) menjelaskan bahwa pengunaan SMS masih banyak digunakan dalam media komunikasi, hal ini karena selain mudah digunakan, biaya juga lebih murah. Demikian pula, ponsel android, yang penggunaanya telah meningkat begitu pesat. Namun teknologi SMS masih tidak aman. Teknologinya tidak menjamin kerahasiaan SMS yang dikirim, pesan yang dikirim melalui SMS yang membentuk pesan asli (*plain text*) berbahaya jika pesan pribadi terkirim dan hanya orang-orang tertentu yang diizinkan untuk membaca pesan, karena pesan asli tersebut dapat dengan mudah terjadi penyadapan. Metode yang digunakan mengunakan 2 metode yaitu *Viegenere Cipher* dan *Caesar Cipher*. Kedua metode ini digabungkan dalam mengenkripsi pesan dan mendekripsi pesan. *Caesar Cipher* merupakan sistem pengkodean berdasarkan substitusi sederhana dengan kriptografi klasik. Enkripsi dan dekripsi di *Caesar Cipher* sistem pengkodean menggunakan operasi pergeseran. Pada dasarnya *Vigenere Cipher* hampir sama dengan *Caesar Cipher*. Perbedaan di antara keduanya adalah pada *Vigenere Cipher* setiap huruf pesan aslinya digeser sebanyak satu huruf pada kuncinya, sedangkan pada *Caesar Cipher* setiap huruf pesannya digeser sebanyak satu huruf yang sama. Algoritma *Vigenere Cipher* ini menggunakan bujur sangkar untuk melakukan enkripsi. Setiap baris di dalam bujur sangkar menyatakan huruf-huruf C*ipher text* yang diperoleh dengan *Caesar cipher*. Hasil dari penggabungan dua metode ini adalah jika rumus kombinasi dilaksanakan untuk Vigenere cipher dengan 94 karakter dan contoh *plain text* "AKU" dengan kunci "DIA", maka akan didapat sebagai berikut:

$$nCr = \frac{n!}{r!\,(n-r)!}$$

$$nCr = \left(\frac{94!}{1!\,(94-1)!}\right)^3 = \left(\frac{94!}{1!\ 93!}\right)^3 = \left(\frac{94\ x\ 93!}{2!\ 93!}\right)^3$$

$$= \left(\frac{94}{1}\right)^3$$

$$nCr = 94! = 830.584$$

Dari hasil yang telah dimodifikasi hasil kedua metode diperoleh dari penerapan rumus dalam kombinasi *Vigenere cipher* dan *Caesar Cipher* yang menggunakan 94 karakter dan contoh *plain text* "AKU" dengan kunci "DIA" adalah 830,584 peluang, karena menggunakan dua lapisan enkripsi penggabungan *Caesar Cipher* dan *Vigenere cipher*. Maka didapatkan 94 × 830,584 = 78.074.896 Jadi hasil akhir diperoleh 78.074.896 peluang untuk membongkar isi pesan teks yang telah dienkripsi.

Pada penelitian Sagheer, Abdulmunem, dan Abduljabbar (Sagheer, Abdulmunem, & Abduljabbar, 2013, pp. 281-285) menjelaskan pesan SMS dengan menerapkan skema kriptografi yang menggabungkan AES (*Advanced Encryption Standard*) dan RC4 (*Rivest's Cipher* 4) untuk enkripsi dan dekripsi dan kunci ekspansi dan generasi algoritma untuk mengamankan pesan SMS yang lebih kuat. Pada mode *default,* pesan SMS tidak terdapat jaminan dan layanan yang bisa mengirimkan pesan SMS secara aman. Keinginan para pengguna ponsel yang menggunakan komunikasi yang menginginkan privasi lebih aman dalam penggunaan sehari-hari dari ponsel mereka. Metode yang digunakan menggunakan algoritma AES (*Advanced Encryption Standard*) dan RC4 (*Ron's Code* 4). Algoritma ini diterapkan dengan panjang kunci 128-bit yang cocok untuk tujuan enkripsi pesan dan waktu proses yang kecil. Algoritma AES diimplementasikan untuk enkripsi dengan 128-bit panjang kunci yang cukup untuk perangkat telepon selular. Algoritma AES dengan ukuran kunci 128-bit ini bisa merepotkan bagi pengguna yang ingin password mereka menjadi kurang dari 16 karakter (128-bit). Oleh karena itu, sistem yang diusulkan dalam penelitian ini menggunakan algoritma kunci RC4 untuk menghasilkan kunci acak dari ukuran yang diinginkan.

3. Metode yang Diusulkan

Algoritma RSA (Rivest Shamir Adleman)

Pada bidang kriptografi, RSA merupakan sebuah algoritma enkripsi yang menggunakan *public key*. RSA adalah algoritma pertama yang sesuai dengan *digital signature* seperti halnya enkripsi, dan menjadi salah satu algoritma yang paling maju di bidang kriptografi *public key*. RSA telah digunakan secara luas pada protokol *electronic commerce*, dan dipercaya untuk mengamankan data menggunakan kunci yang cukup panjang. RSA merupakan algoritma kriptografi kunci-publik yang paling populer dari algoritma yang pernah dibuat. Algoritma RSA dikembangkan pada tahun 1976 oleh 3 orang peneliti dari MIT (*Massachussets Institute of Technology*), yaitu Ron (R)ivest, Adi (S)hamir, dan Leonard (A)dleman. Sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor bilangan prima merupakan kunci keamanan algoritma RSA. Pemfaktoran digunakan untuk mendapatkan kunci pribadi. Selama pemfaktoran bilangan dengan nilai yang besar menjadi faktor-faktor prima belum ditemukan, maka selama itu pula keamanan algoritma RSA tetap terjamin.

Dalam algoritma RSA terdapat 3 tahapan prosesnya yaitu pembangkit kunci, enkripsi dan dekripsi. Berikut adalah 3 tahapan proses dari algoritma RSA:

1. Pembangkit kunci
   Dalam pembangkit kunci terdapat ketentuan nilai-nilai yang harus ditentukan dari algoritma RSA yaitu:
   a. Pilihlah dua buah bilangan prima secara bebas dengan nilai yang besar, misalnya p dan q. Nilai yang dipilih (p dan q) harus dirahasiakan.
   b. Hitung nilai n berdasarkan persamaan $n = p \times q$. Besaran n tidak perlu dirahasiakan.
   c. Hitung nilai m berdasarkan teorema euler menggunakan persamaan $m = (p - 1)(q - 1)$
   d. Pilih sebuah bilangan bulat sebagai kunci publik (e), yang relatif prima terhadap m. e relatif prima terhadap m artinya faktor pembagi terbesar keduanya adalah 1, secara matematis disebut *gcd (e,m)* = 1. Untuk mencarinya dapat digunakan algoritma Euclid
   e. Hitung kunci privat yang disebut d sedemikian hingga agar e x d = 1 atau d = (1 + n . m) / e. Untuk mencari nilai d yang sesuai dapat juga digunakan algoritma Extended Euclid.
2. Enkripsi
   Dalam enkripsi pesan menggunakan rumus sebagai berikut:

   $$C = M^e \ (mod\ n)$$

3. Dekripsi
   Dalam dekripsi pesan menggunakan rumus sebagai berikut:

   $$M = C^d \ (mod\ n)$$

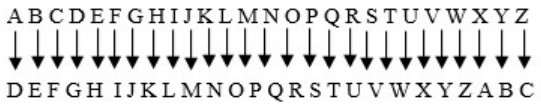
Keterangan:
C = C*ipher text*
M = Message / *Plain text*
e = kunci public
d= kunci privat
n = modulo pembagi

Algoritma Caesar Cipher

Dalam dunia persandian, substitusi *chiper* merupakan yang pertama digunakan pada waktu pemerintahan Julius Caesar, sehingga dikenal dengan *Caesar Cipher*, yaitu mengganti posisi huruf awal dari alfabet. *Caesar Cipher* dilakukan dengan cara menggeser karakter dalam pesan teks sesuai banyaknya kunci yang ditentukan. Penggeseran kunci dalam persandian dilakukan sesuai dengan keinginan pengirim pesan. Penggeseran semua karakter dari plain text menggunakan nilai yang sama merupakan inti dari algoritma kriptografi *Caesar Cipher*. Sedangkan langkah-langkah dalam membentuk *chiper text* dengan *Caesar Cipher* adalah:

1. Menetapkan besarnya nilai penggeseran karakter yang digunakan untuk membentuk *cipher text* menjadi *plain text*.
2. Menukarkan karakter dalam *plain text* menjadi *cipher text* sesuai dengan pergeseran yang telah ditentukan sebelumnya.

Di bidang kriptografi, *Caesar Cipher* merupakan salah satu dari teknik enkripsi yang paling terkenal dan sederhana. Sandi ini adalah sandi substitusi karena setiap huruf dari teks asli yang berupa *plain text* (teks terbuka) diganti menggunakan huruf lain dengan selisih posisi yang sama dalam urutannya. Sebagai contoh *Caesar cipher*, setiap huruf dari *plain text* (teks terbuka) menggunakan huruf selanjutnya ketiga dalam urutan alfabet. Maka pada kasus ini menggunakan kunci pergeseran hurufnya adalah 3. Perubahan huruf-huruf Latin ditunjukkan pada Gambar 1.



Gambar 1 Susunan Alfabet Setelah Digeser Sebanyak 3 Kali

Dalam mengenkripsi (menyandikan) pesan *plain text* (teks terbuka) cukup dengan mengganti setiap huruf dari *plain text* (teks terbuka) dengan huruf substitusinya berdasarkan nilai kunci pergeserannya sehingga diperoleh teks baru. Untuk mendekripsi sandi yang dihasilkan digunakan nilai pergeseran yang sama dengan arah sebaliknya. Contoh dalam menyandikan pesan *plain text* (teks terbuka) adalah sebagai berikut:

Teks terbuka (*plain text*):
JANGAN MENDEKATI PAMULANG GEDUNG A

Teks tersandi (*cipher text*):
MDQJDQ PHQGHNDWL SDPXODQJ JHGXQJ D

Pengkodean setiap huruf Latin (alfabet) berdasarkan nilai integer 'A'= 0, 'B'= 1, 'C'=2 …, 'Z'= 25 dengan pergeseran 3 huruf secara matematis setara dengan dilakukan operasi modulo terhadap *plain text* P sehingga menjadi *cipher text* C dengan persamaan:

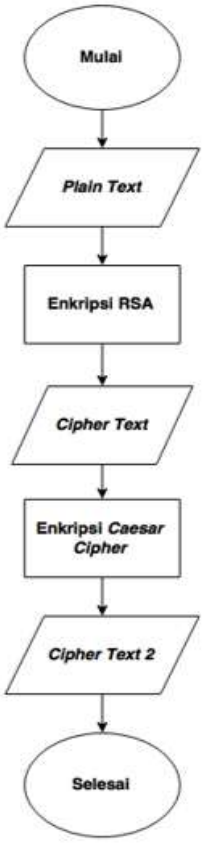
$$\boldsymbol{C = E\,(P) = (P + 3)\ mod\ 26}$$

Keterangan:

- C = *Cipher Text*
- E = Enkripsi
- P = *Plain Text*
- 3 = pergeseran kunci
- mod 26 = Huruf alfabet yang terdiri dari 26 huruf

Dalam analisa untuk pembuatan aplikasi enkripsi dan dekripsi SMS tidak membutuhkan metode yang rumit karena termasuk aplikasi yang sederhana. Secara umum aplikasi yang dibuat adalah aplikasi pengirim pesan yang dapat mengenkripsi *plain text* (teks terbuka) yang akan dikirim lewat layanan SMS. Aplikasi juga harus dapat mendekripsi pesan yang diterima agar dapat dibaca karena pesan yang masuk dalam keadaan terenkripsi. Sehingga pada ponsel receiver (penerima) harus juga terdapat aplikasi enkripsi dan dekripsi SMS. Dalam mengenkripsi dan dekripsi pesan dalam penelitian ini menggunakan suatu metode yaitu dengan menggunakan algoritma RSA dan algoritma *Caesar Cipher.* Untuk penerapan algoritma tersebut pesan yang dikirimkan di enkripsi menggunakan algoritma RSA dan hasil enkripsi algoritma RSA yang berupa *cipher text* untuk meningkatkan keamanannya dienkripsi lagi dengan menggunakan algoritma

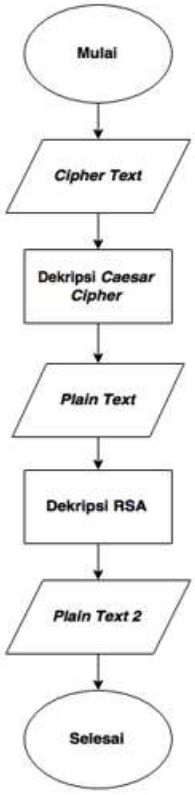
Caesar Cipher begitu juga sebaliknya dalam membuka pesan yang terenkripsi dengan cara dekripsi pesan yaitu dengan algoritma *Caesar Cipher* setelah didekripsi maka akan didekripsi lagi dengan algoritma RSA agar pesan asli atau *plain text* dapat terbuka. Pada intinya dalam metode ini yaitu dengan enkripsi dan dekripsi pesan.



Gambar 2 Flowchart Enkripsi

Penerapan algoritma enkripsi dilakukan untuk mengubah pesan yang berupa *plain text* (teks terbuka) menjadi pesan terenkripsi (tersandi) dan tidak dapat dibaca secara langsung. Untuk membacanya dibutuhkan proses dekripsi menggunakan kuncinya. Dengan algoritma yang terdapat di dalamnya dalam hal ini yaitu algoritma RSA dan *Caesar Cipher* menambahkan data rahasia yang disebut kunci untuk mencegah pesan dapat dibaca/disadap oleh orang yang tidak diijinkan untuk membaca pesan tersebut. Pada Gambar menunjukkan proses alur enkripsi dengan menggunakan algoritma RSA dan algoritma *Caesar Cipher.*

Untuk membuka pesan yang terenkripsi dibutuhkan suatu yang bisa membuka pesan tersebut yaitu dengan cara dekripsi pesan tersebut. Dekripsi kebalikan dari enkripsi. Dekripsi mengubah *cipher text* atau naskah acak menjadi *plain text* atau pesan asli yang mudah dimengerti. Pada Gambar menunjukkan proses alur dari dekripsi pesan dengan menggunakan algoritma RSA dan algoritma *Caesar Cipher.*



Gambar 3 Flowchart Dekripsi

4. Hasil Eksperimen

Dalam enkripsi pesan secara penghitungan manual yaitu mengambil *plain text* yang telah ditentukan untuk dikirimkan ke penerima pesan. *Plain text* akan dienkripsi menggunakan algoritma RSA dan hasil enkripsi tersebut akan dienkripsi lagi dengan menggunakan algoritma *Caesar Cipher*. Berikut adalah pengujian penghitungan secara manual dalam enkripsi pesan menggunakan algoritma RSA dan algoritma *Caesar Cipher.* Untuk enkripsi pesan menggunakan algoritma RSA, dalam penerapan algoritma RSA terdapat penentuan nilai-nilai yang harus ditentukan yaitu adalah sebagai berikut:

1. Menentukan dua buah bilangan prima yang berbeda yaitu dengan notasi bilangan pirma p dan q, kedua bilangan prima ini tidak boleh sama nilainya.

p = 13 dan q = 19

2. Menetapkan nilai modulus yang digunakan sebagai pasangan kunci publik dengan kunci rahasia. Nilai modulus disimbolkan dengan variabel n. Nilai n diperoleh dari perkalian dua bilangan prima yang telah ditetapkan dilangkah sebelumnya, yang disimbolkan dengan variabel p dan variabel q.

n = p x q
n = 13 x 19 = 247

3. Menentukan nilai m, di mana m = (p – 1) x (q – 1)

m = (13 – 1) x (19 – 1)
m = 12 – 18
m = 216

4. Langkah berikutnya dilakukan dengan menentukan *enciphering exponent* yang akan digunakan bersama modulus sebagai pasangan kunci publik. *Enciphering exponent* disimbolkan oleh variabel e.
Persamaan untuk menentukan nilai e adalah sebagai berikut:

gcd (e, m) = 1

Dengan syarat nilai e antara 1 sampai m (1 < e < m) dan nilai e merupakan bilangan prima. gcd adalah kependekan dari kata-kata *greatest common divisor* atau biasa disebut faktor persekutuan terbesar (FPB). Jika gcd atau FPB dari dua buah bilangan memiliki nilai = 1, misalnya gcd(e,m)=1, maka nilai e adalah bilangan yang *relative.*
Algoritma Euclid digunakan untuk mencari gcd dua bilangan sebagai berikut:

- e = 2 ⟶ gcd (e, 216) = gcd (2, 216) = 2
- e = 3 ⟶ gcd (e, 216) = gcd (3, 216) = 3
- e = 4 ⟶ gcd (e, 216) = gcd (4, 216) = 4
- **e = 5 ⟶ gcd (e, 216) = gcd (5, 216) = 1**
- e = 6 ⟶ gcd (e, 216) = gcd (6, 216) = 6
- e = 7 ⟶ gcd (e, 216) = gcd (7, 216) = 7
- e = 8 ⟶ gcd (e, 216) = gcd (8, 216) = 8
- e = 9 ⟶ gcd (e, 216) = gcd (9, 216) = 9
- e = 10 ⟶ gcd (e, 216) = gcd (10, 216) = 10

Dari bilangan gcd atau FPB dari dua buah bilangan harus bernilai 1 (satu) maka nilai e adalah 5.

5. Menghitung nilai d integer sehingga e x d = 1 atau d = (1 + n . m) / e, atau dapat digunakan algoritma *extended* Euclid sebagai berikut:

- n = 0 maka ⟶ $d = \frac{(1+0*216)}{5} = \frac{1}{5} = 0{,}2$
- n = 1 maka ⟶ $d = \frac{(1+1*216)}{5} = \frac{217}{5} = 43{,}4$
- n = 2 maka ⟶ $d = \frac{(1+2*216)}{5} = \frac{433}{5} = 86{,}6$
- n = 3 maka ⟶ $d = \frac{(1+3*216)}{5} = \frac{649}{5} = 129{,}8$
- **n = 4 maka ⟶ $d = \frac{(1+4*216)}{5} = \frac{865}{5} = 173$**
- n = 5 maka ⟶ $d = \frac{(1+5*216)}{5} = \frac{1081}{5} = 216{,}2$
- n = 6 maka ⟶ $d = \frac{(1+6*216)}{5} = \frac{2594}{5} = 294{,}5$

Untuk mendapatkan nilai d maka hasil dari nilai d haruslah bilangan bulat positif atau bilangan asli. Dari hasil di atas hasil yang bilangan bulat positif adalah d = 173 di mana n = 4.

Setelah semua penentuan nilai-nilai algoritma RSA telah ditentukan maka langkah selanjutnya yaitu mengenkripsi pesan. Isi pesan atau *plain text* yang akan dienkripsi yaitu sebagai berikut:

Plain text: besok kita bertemu untuk menyusun strategi

Setelah pesan *plain text* ditentukan maka langkah selanjutnya yaitu mengubah atau mengkonversikan setiap karakter pesan ke dalam bilangan desimal sesuai dalam karakter ASCII.

Setelah diubah ke dalam bentuk bilangan desimal sesuai dengan yang terdapat dalam karakter ASCII maka langkah selanjutnya adalah mengenkripsi pesan tersebut dengan rumus enkripsi algoritma RSA sebagai berikut:

$$C = M^e \ (mod\ n)$$

Keterangan:
C = *Cipher Text*
M = Karakter yang telah diubah ke dalam bilangan decimal sesuai ASCII
e = *enciphering exponent* / kunci publik
n = modulo pembagi

Sesuai rumus enkripsi di atas maka setiap karakter dari isi pesan atau *plain text* tersebut dienkripsi menjadi *cipher text*. Maka dari hasil

enkripsi setiap karakter isi pesan atau *plain text* adalah sebagai berikut:

- b = 98 → 98^5 (*mod* 247) = 9039207968 (*mod* 247) = 167
- e = 101 → 101^5 (*mod* 247) = 10510100501 (*mod* 247) = 43
- s = 115 → 115^5 (*mod* 247) = 20113571875 (*mod* 247) = 20
- o = 111 → 111^5 (*mod* 247) = 16850581551 (*mod* 247) = 232
- k = 107 → 107^5 (*mod* 247) = 14025517307 (*mod* 247) = 217

- spasi = 32 → 32^5 (*mod* 247) = 33554432 (*mod* 247) = 223

- k = 107 → 107^5 (*mod* 247) = 9039207968 (*mod* 247) = 217
- i = 105 → 105^5 (*mod* 247) = 10510100501 (*mod* 247) = 79
- t = 116 → 116^5 (*mod* 247) = 20113571875 (*mod* 247) = 51
- a = 97 → 97^5 (*mod* 247) = 16850581551 (*mod* 247) = 184

- spasi= 32 → 32^5 (*mod* 247) = 33554432 (*mod* 247) = 223

- b = 98 → 98^5 (*mod* 247) = 9039207968 (*mod* 247) = 167
- e = 101 → 101^5 (*mod* 247) = 10510100501 (*mod* 247) = 43
- r = 114 → 114^5 (*mod* 247) = 19254145824 (*mod* 247) = 95
- t = 116 → 116^5 (*mod* 247) = 21003416576 (*mod* 247) = 51
- e = 101 → 101^5 (*mod* 247) = 10510100501 (*mod* 247) = 43
- m = 109 → 109^5 (*mod* 247) = 15386239549 (*mod* 247) = 200
- u = 117 → 117^5 (*mod* 247) = 21924480357 (*mod* 247) = 91

- spasi= 32 → 32^5 (*mod* 247) = 33554432 (*mod* 247) = 223

- u = 117 → 117^5 (*mod* 247) = 21924480357 (*mod* 247) = 91
- n = 110 → 110^5 (*mod* 247) = 16105100000 (*mod* 247) = 2
- t = 116 → 116^5 (*mod* 247) = 21003416576 (*mod* 247) = 51
- u = 117 → 117^5 (*mod* 247) = 21924480357 (*mod* 247) = 91
- k = 107 → 107^5 (*mod* 247) = 14025517307 (*mod* 247) = 217

- spasi= 32 → 32^5 (*mod* 247) = 33554432 (*mod* 247) = 223

- m = 109 → 109^5 (*mod* 247) = 15386239549 (*mod* 247) = 200
- e = 101 → 101^5 (*mod* 247) = 10510100501 (*mod* 247) = 43
- n = 110 → 110^5 (*mod* 247) = 16105100000 (*mod* 247) = 2
- y = 121 → 121^5 (*mod* 247) = 25937424601 (*mod* 247) = 49
- u = 117 → 117^5 (*mod* 247) = 21924480357 (*mod* 247) = 91
- s = 115 → 115^5 (*mod* 247) = 20113571875 (*mod* 247) = 20
- u = 117 → 117^5 (*mod* 247) = 21924480357 (*mod* 247) = 91
- n = 110 → 110^5 (*mod* 247) = 16105100000 (*mod* 247) = 2

- spasi= 32 → 32^5 (*mod* 247) = 33554432 (*mod* 247) = 223

- s = 115 → 115^5 (*mod* 247) = 20113571875 (*mod* 247) = 20
- t = 116 → 116^5 (*mod* 247) = 21003416576 (*mod* 247) = 51
- r = 114 → 114^5 (*mod* 247) = 19254145824 (*mod* 247) = 95
- a = 97 → 97^5 (*mod* 247) = 8587340257 (*mod* 247) = 184
- t = 116 → 116^5 (*mod* 247) = 21003416576 (*mod* 247) = 51
- e = 101 → 101^5 (*mod* 247) = 10510100501 (*mod* 247) = 43
- g = 103 → 103^5 (*mod* 247) = 11592740743 (*mod* 247) = 12
- i = 105 → 105^5 (*mod* 247) = 12762815625 (*mod* 247) = 79

Setelah semua karakter isi pesan telah dienkripsi maka dari hasil enkripsi tersebut yang berupa bilangan desimal akan dikonversi lagi sesuai karakternya dengan ASCII.

Cipher text menggunakan algoritma RSA:
§+DC4èÙßÙO3¸ß§+_3+È[ß[STX3[UßE+STX1[DC4[STXßDC43_.3+FFO

Setelah melakukan penghitungan manual menggunakan algoritma RSA maka sesuai dari penerapan penelitian ini hasil dari enkripsi menggunakan algoritma RSA akan dienkripsi kembali dengan menggunakan algoritma *Caesar cipher.* Berikut adalah pengujian penghitungan manual penggunaan algoritma *Caesar Cipher* dengan besar pergeseran sebesar kunci yang diberikan yaitu 249. Dengan nilai pergeseran tersebut, didapat rumus untuk menghitung enkripsi pesan menggunakan algoritma Caesar Cipher sebagai berikut:

$$C = (p + k) \bmod 256$$

Keterangan:
C = *Cipher text*
P = *Plain text*
K = *K*unci

Di bawah ini adalah *cipher text* yang telah dienkripsikan dengan algoritma RSA dan akan dienkripsi kembali dengan mengunakan algoritma *Caesar Cipher.*

Hasil *Cipher text* menggunakan algoritma RSA:
§+DC4èÙßÙO3¸ß§+_3+È[ß[STX3[UßE+STX1[DC4[STXßDC43_.3+FFO
Pergeseran kunci = 249

Dari isi pesan enkripsi tersebut lalu dikonversikan atau diubah ke dalam bilangan desimal sesuai dalam karakter ASCII. Sesuai dengan rumus enkripsi *Caesar Chiper* yang diberikan di atas berikut hasil penghitungan enkripsi setiap karakter isi pesan menggunakan algoritma *Caesar Chiper.*

$$C = (p + k) \bmod$$

- 167 = § → 167 + 249(*mod* 256) = 416 (*mod* 256) = 160
- 43 = + → 167 + 249(*mod* 256) = 292 (*mod* 256) = 36
- 20 = **DC4** → 20 + 249(*mod* 256) = 269 (*mod* 256) = 13
- 232 = **è** → 232 + 249(*mod* 256) = 481 (*mod* 256) = 225
- 217 = **Ù** → 217 +249(*mod* 256) = 466 (*mod* 256) = 210

- 223 = **ß** → 223 + 249(*mod* 256) = 472 (*mod* 256) = 216

- 217 = **Ù** → 217 + 249(*mod* 256) = 466 (*mod* 256) = 210
- 79 = **O** → 79 + 249(*mod* 256) = 328 (*mod* 256) = 72
- 51 = **3** → 51 +249(*mod* 256) = 300 (*mod* 256) = 44
- 184 = ¸ → 184 + 249(*mod* 256) = 433 (*mod* 256) = 177

- 223 = **ß** → 223 + 249(*mod* 256) = 472 (*mod* 256) = 216
- 167 = § → 167 + 249(*mod* 256) = 416 (*mod* 256) = 160
- 43 = + → 43 + 249(*mod* 256) = 292 (*mod* 256) = 36
- 95 = _ → 95 + 249(*mod* 256) = 344 (*mod* 256) = 88
- 51 = **3** → 51 + 249(*mod* 256) = 300 (*mod* 256) = 44
- 43 = + → 43 + 249(*mod* 256) = 292 (*mod* 256) = 36
- 200 = **È** → 200 + 249(*mod* 256) = 449 (*mod* 256) = 193
- 91 = **[** → 91 + 249(*mod* 256) = 340 (*mod* 256) = 84

- 223 = **ß** → 223 + 249(*mod* 256) = 472 (*mod* 256) = 216

- 91 = **[** → 91 + 249(*mod* 256) = 340 (*mod* 256) = 84
- 2 = **STX** → 2 + 249(*mod* 256) = 251 (*mod* 256) = 251
- 51 = **3** → 51 + 249(*mod* 256) = 300 (*mod* 256) = 44
- 91 = **[** → 91 + 249(*mod* 256) = 340 (*mod* 256) = 84
- 217 = **Ú** → 217 + 249(*mod* 256) = 466 (*mod* 256) = 210

- 223 = **ß** → 223 + 249(*mod* 256) = 472 (*mod* 256) = 216

- 200 = **È** → 200 + 249(*mod* 256) = 449 (*mod* 256) = 193
- 43 = **+** → 43 + 249(*mod* 256) = 292 (*mod* 256) = 36
- 2 = **STX** → 2 + 249(*mod* 256) = 251 (*mod* 256) = 251
- 49 = **1** → 49 + 249(*mod* 256) = 298 (*mod* 256) = 42
- 91 = **[** → 91 + 249(*mod* 256) = 340 (*mod* 256) = 84
- 20 = **DC4** → 20 + 249(*mod* 256) = 269 (*mod* 256) = 13
- 91 = **[** → 91 + 249(*mod* 256) = 340 (*mod* 256) = 84
- 2 = **STX** → 2 + 249(*mod* 256) = 251 (*mod* 256) = 251

- 223 = **ß** → 223 + 249(*mod* 256) = 472 (*mod* 256) = 216

- 20 = **DC4** → 20 + 249(*mod* 256) = 269 (*mod* 256) = 13
- 51 = **3** → 51 + 249(*mod* 256) = 300 (*mod* 256) = 44
- 95 = **_** → 95 + 249(*mod* 256) = 344 (*mod* 256) = 88
- 184 = **¸** → 184 + 249(*mod* 256) = 433 (*mod* 256) = 177
- 51 = **3** → 51 + 249(*mod* 256) = 300 (*mod* 256)= 44
- 43 = **+** → 43 + 249(*mod* 256) = 292 (*mod* 256)= 36
- 12 = **FF** → 12 + 249(*mod* 256) = 261 (*mod* 256)= 5
- 79 = **O** → 79 + 249(*mod* 256) = 328 (*mod* 256)= 72

Setelah semua karakter isi pesan telah dienkripsi kembali dengan algoritma *Caesar Cipher* maka dari hasil enkripsi tersebut akan dikonversi lagi sesuai karakternya.

Berikut adalah hasil pesan yang telah dienkripsi oleh algoritma RSA dan dienkripsi kembali oleh algoritma *Caesar Cipher*. Dan hasil ini yang akan dikirimkan kepenerima pesan.

Hasil *cipher text 2* yang akan dikirimkan:
$ áÒØÒH,±Ø $X,$ÂTØTû,TÒØÁ$û*T TûØ ,X±,$ ▪▪

Pada Gambar 4 (a) menunjukkan halaman tulis pesan untuk mengenkripsi pesan yang telah dimasukkan nomer tujuan penerima pesan, kunci pesan dan isi pesan. Selanjutnya pada Gambar 4 (b) menunjukkan hasil dari enkripsi pesan apabila telah menekan tombol enkripsi yang berada di samping isi pesan. Hasil enkripsi sesuai dengan hasil penghitungan manual yang telah dihitung yaitu **$ áÒØÒH,±Ø $X,$ÂTØTû,TÒØÁ$û*T TûØ ,X±,$ H.** Dari hasil enkripsi tersebut didapat *cipher text* yang akan dikirimkan ke nomer tujuan penerima pesan dengan menekan tombol kirim yang ada di bawah hasil enkripsi maka pesan akan terkirim ke penerima pesan. Setelah mengrim pesan selesai yaitu tinggal penerima pesan dan membuka pesan tersebut. Untuk membuka pesan apabila telah masuk pesan dari pengirim maka buka aplikasi enkripsi SMS setelah terbuka maka pilih menu kotak masuk maka akan terbuka tampilan daftar SMS yang masuk

Gambar 4 Antarmuka Aplikasi Enkripsi SMS

Setelah Enkripsi pesan maka selanjutnya yaitu pengujian dekripsi. Pesan yang dikirimkan ke penerima pesan untuk membukannya haruslah didekripsi terlebih dahulu. Dalam pengujian dekripsi pesan algoritma *Caesar Cipher* digunakan terlebih dahulu setelah didekripsi pesan dengan algoritma *Caesar Cipher* maka hasil dari dekripsi tersebut belum bisa terbaca karena masih terenkripsi oleh algoritma RSA maka untuk membacanya perlu didekripsi kembali dengan menggunakan algoritma RSA. Berikut adalah dekripsi pesan dengan menggunakan algoritma *Caesar Cipher* terlebih dahulu.

Selanjutnya untuk membuka hasil pesan yang telah dienkripsi yaitu dengan cara melakukan dekripsi pesan yaitu dengan rumus dekripsi pesan dengan algoritma *Caesar Cipher* sebagai berikut:

$$P = (c - k) \, mod \, 256$$

Keterangan:
c = *Cipher Text*
k = kunci
P = *Plain Text*

Sesuai rumus dekripsi di atas maka setiap karakter dari isi pesan atau *plain text* tersebut dienkripsi menjadi *cipher text*. Maka dari hasil dekripsi setiap karakter isi pesan atau *cipher text* diubah dengan penghitungan manual sebagai berikut:

- 160 = **Null** → 160 − 249 (*mod* 256) = − 89 (*mod* 256) = 167
- 36 = **$** → 36 − 249 (*mod* 256) = − 213 (*mod* 256) = 43
- 13 = **CR** → 13 − 249 (*mod* 256) = − 236 (*mod* 256) = 20
- 225 = **á** → 225 − 249 (*mod* 256) = − 24 (*mod* 256) = 232
- 210 = **Ò** → 210 − 249 (*mod* 256) = − 39 (*mod* 256) = 217

- 216 = **Ø** → 216 − 249 (*mod* 256) = − 33 (*mod* 256) = 223

- 210 = **Ò** → 210 − 249 (*mod* 256) = − 39 (*mod* 256) = 217
- 72 = **H** → 72 − 249 (*mod* 256) = − 177(*mod* 256) = 79
- 44 = **,** → 44 − 249 (*mod* 256) = − 205 (*mod* 256) = 51
- 177 = **±** → 177 − 249 (*mod* 256) = − 72 (*mod* 256) = 184

- 216 = **Ø** → 216 − 249 (*mod* 256) = − 33 (*mod* 256) = 223

- 160 = **Null** → 160 − 249 (*mod* 256) = − 89 (*mod* 256) = 167
- 36 = **$** → 36 − 249 (*mod* 256) = − 213 (*mod* 256) = 43
- 88 = **X** → 88 − 249 (*mod* 256) = − 161 (*mod* 256) = 95
- 44 = **,** → 44 − 249 (*mod* 256) = − 205 (*mod* 256) = 51
- 36 = **$** → 36 − 249 (*mod* 256) = − 213 (*mod* 256) = 43
- 193 = **Â** → 193 − 249 (*mod* 256) = − 56 (*mod* 256) = 200
- 84 = **T** → 84 − 249 (*mod* 256) = − 165 (*mod* 256) = 91

- 216 = **Ø** → 216 − 249 (*mod* 256) = − 33 (*mod* 256) = 223
- 84 = **T** → 84 − 249 (*mod* 256) = − 165 (*mod* 256) = 91
- 251 = **û** → 251 − 249 (*mod* 256) = 2 (*mod* 256) = 2
- 44 = **,** → 44 − 249 (*mod* 256) = − 205 (*mod* 256) = 51
- 84 = **T** → 84 − 249 (*mod* 256) = − 165 (*mod* 256) = 91
- 210 = **Ò** → 210 − 249 (*mod* 256) = − 39 (*mod* 256) = 217

- 216 = **Ø** → 216 − 249 (*mod* 256) = − 33 (*mod* 256) = 223

- 193 = **Á** → 193 − 249 (*mod* 256) = − 56 (*mod* 256) = 200
- 36 = **$** → 36 − 249 (*mod* 256) = − 213 (*mod* 256) = 43
- 251 = **û** → 251 − 249 (*mod* 256) = 2 (*mod* 256) = 2
- 42 = ***** → 42 − 249 (*mod* 256) = − 207 (*mod* 256) = 49
- 84 = **T** → 84 − 249 (*mod* 256) = − 165 (*mod* 256) = 91
- 13 = **CR** → 13 − 249 (*mod* 256) = − 236 (*mod* 256)= 20
- 84 = **T** → 84 − 249 (*mod* 256) = − 165 (*mod* 256) = 91
- 251 = **û** → 251 − 249 (*mod* 256) = 2 (*mod* 256) = 2

- 216 = **Ø** → 216 − 249 (*mod* 256) = − 33 (*mod* 256) = 223

- 13 = **CR** → 13 − 249 (*mod* 256) = − 236 (*mod* 256)= 20
- 44 = **,** → 44 − 249 (*mod* 256) = − 205 (*mod* 256) = 51
- 88 = **X** → 88 − 249 (*mod* 256) = − 161 (*mod* 256) = 95
- 177 = **±** → 177 − 249 (*mod* 256) = − 72 (*mod* 256) = 184
- 44 = **,** → 44 − 249 (*mod* 256) = − 205 (*mod* 256) = 51
- 36 = **$** → 36 − 249 (*mod* 256) = − 213 (*mod* 256) = 43
- 5 = **ENQ** → 5 − 249 (*mod* 256) = − 244 (*mod* 256) = 12
- 72 = **H** → 72 − 249 (*mod* 256) = − 177 (*mod* 256) = 79

Setelah semua karakter isi pesan *cipher text 2* telah didekripsi maka dari hasil dekripsi tersebut akan dikonversi lagi sesuai karakternya pada tabel ASCII.

Cipher text menggunakan dekripsi algoritma *Caesar Cipher*:
§+DC4èÙßÙO3¸ß§+_3+È[ß[STX3[UßE+STX 1[DC4[STXßDC43 .3+FFO

Setelah melakukan penghitungan manual menggunakan algoritma *Caesar Cipher* maka sesuai dari penerapan penelitian ini hasil dari dekripsi menggunakan algoritma *Caesar Cipher* akan dienkripsi kembali dengan menggunakan algoritma RSA. Berikut adalah pengujian penghitungan manual penggunaan algoritma RSA. rumus untuk menghitung dekripsi pesan menggunakan algoritma RSA sebagai berikut:

$$P = M^d \ (mod \ n)$$

Keterangan:

P = *Plain Text*

M = Karakter yang telah diubah ke dalam bilangan decimal sesuai ASCII / *plain text*

d = *enciphering exponent /* kunci privat

n = modulo pembagi

Sesuai rumus dekripsi algoritma RSA di atas, maka dari hasil penghitungan manual dekripsi algoritma *Caesar Cipher* diubah menjadi *plain text* atau pesan asli adalah sebagai berikut penghitungannya:

- 167 = § → $167^{173} \ (mod \ 247) = 98$ → **b**
- 43 = + → $43^{173} \ (mod \ 247) = 101$ → **e**
- 20 = **DC4** → $20^{173} \ (mod \ 247) = 115$ → **s**
- 232 = **è** → $232^{173} \ (mod \ 247) = 111$ → **o**
- 217 = **Ù** → $217^{173} \ (mod \ 247) = 107$ → **k**

- 223 = **ß** → $223^{173} \ (mod \ 247) = 32$ → **Spasi**

- 217 = **Ù** → $217^{173} \ (mod \ 247) = 107$ → **k**
- 79 = **O** → $79^{173} \ (mod \ 247) = 105$ → **i**
- 51 = **3** → $51^{173} \ (mod \ 247) = 116$ → **t**
- 184 = **¸** → $184^{173} \ (mod \ 247) = 97$ → **a**

- 223 = **ß** → $223^{173} \ (mod \ 247) = 32$ → **Spasi**

- 167 = § → $167^{173} \ (mod \ 247) = 98$ → **b**
- 43 = + → $43^{173} \ (mod \ 247) = 101$ → **e**
- 95 = **_** → $95^{173} \ (mod \ 247) = 114$ → **r**
- 51 = **3** → $51^{173} \ (mod \ 247) = 116$ → **t**
- 43 = + → $43^{173} \ (mod \ 247) = 101$ → **e**
- 200 = **È** → $200^{173} \ (mod \ 247) = 109$ → **m**
- 91 = **[** → $91^{173} \ (mod \ 247) = 117$ → **u**

- 223 = **ß** → $223^{173} \ (mod \ 247) = 32$ → **Spasi**

- 91 = **[** → $91^{173} \ (mod \ 247) = 117$ → **u**
- 2 = **STX** → $2^{173} \ (mod \ 247) = 110$ → **n**
- 51 = **3** → $51^{173} \ (mod \ 247) = 116$ → **t**
- 91 = **[** → $91^{173} \ (mod \ 247) = 117$ → **u**
- 217 = **Ù** → $217^{173} \ (mod \ 247) = 107$ → **k**

- 223 = **ß** → $223^{173} \ (mod \ 247) = 32$ → **Spasi**

- 200 = **È** → $200^{173} \ (mod \ 247) = 109$ → **m**
- 43 = + → $43^{173} \ (mod \ 247) = 101$ → **e**
- 2 = **STX** → $2^{173} \ (mod \ 247) = 110$ → **n**
- 49 = **1** → $49^{173} \ (mod \ 247) = 121$ → **y**
- 91 = **[** → $91^{173} \ (mod \ 247) = 117$ → **u**
- 20 = **DC4** → $20^{173} \ (mod \ 247) = 115$ → **s**
- 91 = **[** → $91^{173} \ (mod \ 247) = 117$ → **u**
- 2 = **STX** → $2^{173} \ (mod \ 247) = 110$ → **n**

- 223 = **ß** → $223^{173} \ (mod \ 247) = 32$ → **Spasi**

- 20 = **DC4** → $20^{173} \ (mod \ 247) = 115$ → **s**
- 51 = **3** → $51^{173} \ (mod \ 247) = 116$ → **t**
- 95 = **_** → $95^{173} \ (mod \ 247) = 114$ → **r**
- 184 = **¸** → $184^{173} \ (mod \ 247) = 97$ → **a**
- 51 = **3** → $51^{173} \ (mod \ 247) = 116$ → **t**
- 43 = + → $43^{173} \ (mod \ 247) = 101$ → **e**
- 12 = **FF** → $12^{173} \ (mod \ 247) = 103$ → **g**
- 79 = **O** → $79^{173} \ (mod \ 247) = 105$ → **i**

Setelah semua karakter isi pesan *cipher text* telah didekripsi maka dari hasil penghitungan dekripsi tersebut akan dikonversi lagi sesuai karakternya pada tabel ASCII dan hasil dekripsi tersebut sudah bisa dibaca oleh penerima pesan. Berikut adalah hasil pesan yang telah didekripsi oleh algoritma *Caesar Cipher* dan didekripsi kembali oleh algoritma RSA. Dan hasil ini sudah bisa terbaca oleh penerima pesan dengan mudah karena telah kembali menjadi *plain text* atau pesan asli.

Plain text: besok kita bertemu untuk menyusun strategi

Pada Gambar 5 (a) menunjukkan halaman terima pesan untuk didekripsi. Setelah itu masukan kunci yang sama seperti pengirim pesan kiriman untuk mengenkripsi pesan, maka pesan yang terenkripsi akan terbuka sesuai dengan pesan aslinya. Pada Gambar 5 (b) menunjukkan membuka pesan dengan memasukkan kunci dan menekan tombol dekripsi dan setelah itu muncul hasil dekripsi pesan yaitu pesan asli. Hasil dari dekripsi pesan juga sama dengan hasil dari penghitungan manual yaitu sama dengan pesan asli yang sesuai pengirim pesan berikan.

Gambar 5 Antarmuka Aplikasi Enkripsi SMS

5. Kesimpulan

Dari hasil implementasi dan analisa aplikasi enkripsi dan dekripsi pesan SMS yang dibuat dapat disimpulkan sebagai berikut:

1. Integrasi algoritma enkripsi RSA (*Rivest Shamir Adleman*) dengan algoritma enkripsi *Caesar Cipher* dapat meningkatkan kerumitan proses dekripsi pesan yang dikirim.
2. Aplikasi SMS yang menerapkan integrasi algoritma enkripsi RSA (*Rivest Shamir Adleman*) dengan algoritma enkripsi *Caesar Cipher* dapat meningkatkan kerahasiaan untuk mengamankan pesan teks sehingga dapat mencegah pihak yang tidak berhak dapat membacanya.

Daftar Pustaka

Agoyi, M., & Seral, D. (2010). SMS Security: An Asymmetric Encryption Approach. *Sixth International Conference on Wireless and Mobile Communications*, 448-452 .

Alvianto, A. R., & Darmaji. (2015). Pengaman Pengiriman Pesan Via SMS dengan Algoritma RSA Berbasis Android. *Jurnal Sains dan Seni ITS Vol. 4, No.1,*, 1-6.

Castiglione, A., Cattaneo, G., Cembalo, M., & Santis, A. D. (2012). Engineering a Secure Mobile Messaging Framework. *Computers & Security*, 771-781.

Dent, A. W., & Mitchell, C. J. (2005). *User's Guide to Criptography and Standards.* London: Artech House.

Fahrianto, F., Masruroh, S. U., & Ando, N. Z. (2014). Encrypted SMS application on Android with Combination of Caesar Cipher and Vigenere Algorithm. *International Conference on Cyber and IT Service Management (CITSM) 2014*, 31-33.

Han, L. C., & Mahyuddin, N. M. (2014). An Implementation of Caesar Cipher and XOR Encryption Technique in a Secure Wireless Communication. *2nd International Conference on Electronic Design (ICED)*, 111-116.

Nagar, S. A., & Alshamma, S. (2012). High Speed Implementation of RSA Algorithm with Modified Keys Exchange. *6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*, 639-642.

Patil, M., Sahu, V., & Jain, A. (2014). SMS Text Compression and Encryption on Android O.S. *International Conference on Computer Communication and Informatics*, 1-6.

Pereira, G. C., Santos, M. A., & Olieveira, B. T. (2013). SMSCrypto: A lightweight cryptographic framework for secure SMS transmission. *The Journal of Systems and Software*, 698-706.

Sagheer, A. M., Abdulmunem, A., & Abduljabbar, M. A. (2013). SMS Security for Smartphone. *Sixth International Conference on Developments in eSystems Engineering*, 281-285.

Saxena, N., & Chaudhari, N. S. (2014). Secure SMS: A secure SMS protocol for VAS and other applications. *The Journal of Systems and Software*, 138-150.

Senthil, K., Prasanthi, K., & Rajaram, R. (2013). A Modern Avatar of Julius Ceasar and Vigenere

Cipher. *International Conference on Computational Intelligence and Computing Research*, 1-3.

Zhao, G., Yang, X., Zhou, B., & Wei, W. (2010). RSA-Based Digital Image Encryption Algorithm In Wireless Sensor Networks. *2nd International Conference on Signal Processing Systems (ICSPS)*, 640-643.