Pengaplikasian *Caterpillar* RSA dan ECDSA untuk Pengamanan Pengiriman Nomor Rekening Melalui *Messaging Platform*

Winston Wijaya and 13517018
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]winstonwijaya59@gmail.com, [2]13517018@std.stei.itb.ac.id

***Abstract*—*Messaging platform* semakin berkembang dan digunakan untuk berkomunikasi sehari-hari. *Messaging platform* sendiri secara singkat merupakan peralatan yang memungkinkan pengguna untuk bertukar informasi. Saat ini sudah ada banyak *messaging platform*, diantaranya Facebook Messenger dan WhatsApp. Pesan yang dikirimkan pun beragam, mulai dari sekadar sapaan, sampai terkadang pengiriman data sensitif jika memang mendesak atau diperlukan. Beberapa *messaging platform* telah menyediakan sistem pengamanan untuk pengiriman pesan, termasuk data sensitif agar terjaga sampai tujuan, namun ada pula yang masih belum menyediakan sistem untuk pengamanan pengiriman pesan, walaupun pengiriman data sensitif sebenarnya tidak disarankan. Berangkat dari hal tersebut, algoritma untuk mengamankan pesan, serta memastikan keaslian pesan dan pengirim perlu diimplementasikan. Algoritma RSA merupakan algoritma yang dapat digunakan untuk mengenkripsi berbagai pesan , namun hasil yang diperoleh berupa angka dan panjangnya kurang lebih dua kali dibandingkan pesan aslinya, maka dilakukan modifikasi dan penyesuaian berupa algoritma *Caterpillar RSA*. Untuk memastikan keaslian pesan dan pengirim, algoritma ECDSA cocok untuk digunakan dalam menandatangani pesan yang dikirimkan. Makalah ini bertujuan untuk mengimplementasikan *Caterpillar* RSA dan ECDSA dalam pengamanan pengiriman nomor rekening melalui berbagai *messaging platform* yang sering digunakan saat ini.**

***Keywords*—*digital signing*, ECDSA, kriptografi, RSA, *messaging platform*.**

I. Pendahuluan

Perkembangan teknologi yang pesat mendukung perkembangan berbagai hal yang ada di dunia dan merubah pola dan perilaku manusia. Komunikasi merupakan salah satu hal yang tentu dilakukan manusia sebagai makhluk sosial untuk berinteraksi dan bertukar informasi. Bentuk komunikasi sendiri beragam, mulai dari yang disampaikan secara lisan maupun tertulis.

Awalnya, ketika ingin berkomunikasi jarak jauh, manusia menggunakan berbagai alat komunikasi, salah satunya yang sering digunakan menggunakan surat yang kemudian berubah menjadi pengiriman pesan melalui *email* atau *chatting* dengan lawan bicara. Saat ini sudah ada banyak *platform* yang dapat digunakan untuk bertukar pesan, diantaranya dari LINE, Whatsapp, dan Facebook Messenger. Pertukaran informasi atau pesan menggunakan berbagai *platform* bisa berupa berbagai hal, mulai dari diskusi tentang suatu topik, beramah tamah, sampai bertukaran data pribadi atau data sensitif tertentu karena satu dan lain hal.

Secara umum, pengiriman atau pertukaran data pribadi atau data sensitif melalui *chatting* atau *email* tidak disarankan karena rawan disadap atau dicuri oleh pihak tertentu demi keuntungan pribadi. Salah satu data yang lumayan rawan untuk dibagikan dan lumayan berbahaya jika dibagikan adalah data nomor rekening. Beberapa *messaging platform* sudah memasang sistem enkripsi dan dekripsi pesan yang dikirim pengguna demi menjaga kerahasiaan dan keamanan data pengguna terlepas dari beberapa kejadian atau percobaan untuk mencurangi sistem yang sudah diterapkan. Untuk beberapa *platform* yang belum dilengkapi sistem tersebut, penambahan sistem enkripsi dan dekripsi pesan merupakan suatu hal yang harus segera diimplementasikan. Ada beberapa cara untuk menjaga kerahasiaan pesan dan integritas pengirimnya, yakni mengkombinasikan algoritma RSA dengan metode tanda tangan digital yang akan dibahas pada makalah ini.

II. Dasar Teori

A. Kriptografi

Kriptografi berasal dari bahasa Yunani yang terdiri dari dua kata, yakni cryptós (*secret*) dan gráphein (*writing*) atau secara sederhana dapat diartikan sebagai *secret writing* atau tulisan rahasia. Ada beberapa pendapat ahli terkait definisi kriptografi, diantaranya ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, serta otentikasi (Menez, 1996) dan ilmu dan seni untuk menjaga keamanan pesan. (Schneier, 1996). Dalam kriptografi, ada empat hal yang diutamakan, yakni keamanan pesan (*confidentiality / privacy / secrecy*), keaslian pesan (*data integrity*), keasilan pengirim dan penerima pesan (*authentication*), dan anti penyangkalan (*non-repudiation*).

Dalam kriptografi, ada beberapa istilah (*term*) yang harus diketahui, diantaranya[7].

1. Pesan

   Pesan atau dalam kriptografi dikenal sebagai

plaintext / *plain-image* / *plain-video* merupakan informasi yang dapat dibaca dan dimengerti maknanya. Pesan dapat dimengerti melalui persepsi secara visual maupun audio. Beberapa contoh pesan adalah teks, gambar, musik, dan video.

2. *Ciphertext*

   Ciphertext atau kriptogram merupakan pesan yang telah disandikan sehingga tidak bermakna lagi. Pesan diubah menjadi *ciphertext* agar pesan tersebut tidak dapat dibaca atau dimengerti oleh pihak yang tidak berwenang. Contoh *plaintext* dan *ciphertext* diantaranya

   Plaintext: HelloWorld
   Ciphertext: JgnnqYqtnf

3. Kunci

   Kunci merupakan parameter yang digunakan pada enkripsi dan dekripsi. Sebuah kunci dapat bersifat rahasia (hanya diketahui pengirim pesan) atau bersifat publik (diketahui umum). Kunci dapat berupa deretan angka, deretan karakter, atau kombinasi keduanya.

4. Enkripsi dan dekripsi

   Enkripsi atau *enchipering* merupakan proses penyandian pesan menjadi *ciphertext* guna merahasiakan suatu pesan. Enkripsi memetakan suatu *plaintext* (P) menjadi *ciphertext* (C). Dekripsi atau *deciphering* merupakan kebalikan dari enkripsi yang membalikkan *ciphertext* menjadi *plaintext* semula. Dekripsi memetakan suatu *ciphertext* menjadi *plaintext*.

5. *Cipher*

   Cipher merupakan algoritma enkripsi atau dekripsi. *Cipher* dapat berupa fungsi matematika untuk proses enkripsi dan dekripsi. Contoh *cipher* untuk enkripsi adalah menggeser setiap huruf empat ke kanan.

6. Penyadap

   Penyadap adalah orang atau mesin yang mencoba menangkap pesan saat pesan sedang ditransmisikan. Penyadapan dilakukan untuk mendapatkan informasi sebanyak-banyaknya. Penyadap juga dikenal sebagai *eavesdropper*, *enemy, adversary*, atau *interceptor*.

B. Messaging Platform dan *Messaging as a Platform*

Messaging platform merupakan peralatan unik yang memungkinkan pengguna internet untuk bertukar pesan untuk berkomunikasi. Beberapa contoh *messaging platform* adalah Facebook Messenger dan WhatsApp. Di lain hal, *Messaging as a Platform*[9] merupakan evolusi dari sistem *messaging* dari sistem tertutup yang merupakan sumber pesan dan berakhir di sebuah *Messaging App* pada sistem menjadi sebuah "*platform*" tempat pihak lain mampu membangun aplikasi untuk pertukaran pesan. *Messaging as a Platform* harus memuat semua fitur pada *platform* yang diharapkan pengembang aplikasi untuk *platform* tersebut, seperti API dan ekosistem.

C. Sensitive Data

Sensitive Data[12] merupakan informasi yang harus dilindungi dengan ketat dari akses pihak yang tidak berwenang. Kerusakan dan kehilangan pada data sensitif dapat mempengaruhi kehidupan seseorang secara drastis. Beberapa contoh data sensitif diantaranya data pribadi, rekaman pendidikan, dan informasi pelanggan. Secara umum, data sensitif merupakan data yang mengandung beberapa hal, diantaranya suku, agama, dan ras (SARA); data genetik; data kesehatan; data rahasia; dan data keuangan (salah satunya nomor rekening dan nomor kartu kredit).

D. Algoritma Rivest-Shamir-Adleman (RSA)[4]

Algoritma *Rivest-Shamir-Adleman* (RSA) merupakan algoritma yang ditemukan oleh tiga orang peneliti dari MIT, yakni Ronald Rivest, Adi Shamir, dan Len Adleman, pada tahun 1976. Algoritma ini merupakan salah satu algoritma kunci publik yang masih terkenal saat ini. Keunggulan dari implementasi algoritma ini terletak pada kompleksitas pemfaktoran bilangan bulat yang besar menjadi faktor-faktor prima.

Ada beberapa komponen dari algoritma ini, diantaranya.

1. Komponen rahasia

   Komponen rahasia merupakan komponen yang tidak boleh diketahui oleh pihak luar selain pengguna. Komponen ini dapat berupa deretan angka atau kombinasi angka dan karakter. Beberapa komponen rahasia pada algoritma RSA adalah bilangan prima p dan q, fungsi Totient Euler ($\Phi(n) = (p-1)*(q-1)$) untuk menentukan berapa bilangan yang relatif prima terhadap n, kunci dekripsi d ($d \equiv e^{-1} \bmod \Phi(n)$), dan *plaintext* m yang merupakan pesan yang ingin dikirim.

2. Komponen publik

   Komponen rahasia merupakan komponen yang boleh dibagikan ke pihak luar atau umum selain pengguna. Komponen ini dapat berupa deretan angka atau kombinasi angka dan karakter. Beberapa komponen publik diantaranya nilai n ($n = p*q$), kunci enkripsi e (bilangan yang relatif prima terhadap $\Phi(n)$), dan *ciphertext* c yang merupakan pesan yang telah terenkripsi.

Ada tiga prosedur yang dilakukan pada algoritma RSA, yakni.

1. Pembangkitan Kunci

   Ada beberapa tahapan yang dilakukan untuk pembangkitan kunci RSA, yaitu.

   a. Memilih dua buah bilangan prima p dan q
   b. Menghitung nilai $n = p*q$
   c. Menghitung nilai $\Phi(n) = (p-1)*(q-1)$
   d. Memilih sebuah bilangan prima e sebagai kunci publik yang nilainya relatif prima terhadap $\Phi(n)$.
   e. Menghitung nilai kunci privat d dengan persamaan $d \equiv e^{-1} \bmod \Phi(n)$.

   Dari pengerjaan tahapan di atas, hasil yang diperoleh ada dua, yakni kunci publik (e, n) dan kunci privat (d, n).

2. Enkripsi

   Ada beberapa tahapan dalam melakukan enkripsi, yaitu.

   1. Membagi *plaintext* ke dalam beberapa blok

$m_1, m_2, .., m_n$ dengan syarat $0 \leq m_i < n - 1$.

2. Menghitung nilai blok *ciphertext* c_i dengan persamaan $c_i = m_i^e \bmod n$
3. Menggabungkan kembali blok-blok *ciphertext* menjadi *ciphertext* yang utuh

3. Dekripsi

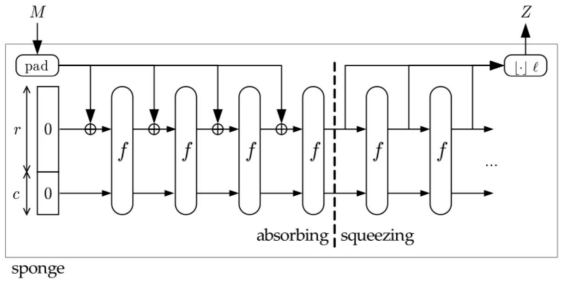
Ada beberapa tahapan dalam melakukan dekripsi, yaitu.

1. Membagi *ciphertext* ke dalam beberapa blok $c_1, c_2, .., c_n$
2. Menghitung kembali blok *plaintext* m_i dengan persamaan $m_i = c_i^d \bmod n$
3. Menggabungkan kembali blok-blok *ciphertext* menjadi *plaintext* yang utuh
4. Jika *plaintext* berupa teks (bukan sederetan angka saja), buat algoritma untuk mengembalikan nilai pada *plaintext* menjadi pesan yang sesungguhnya

E. Algoritma SHA-3 (Keccak)[6]

SHA-3 (Secure Hash Algorithm 3) adalah anggota terbaru dari standar Secure Hash Algorithm , dirilis oleh NIST pada tanggal 5 Agustus 2015. SHA-3 adalah bagian dari keluarga primitif yang lebih luas kriptografi Keccak, yang dirancang oleh Guido Bertoni , Joan Daemen , Michaël Peeters, dan Gilles Van Assche , membangun di atas RadioGatún. Meskipun merupakan bagian dari seri standar yang sama, SHA-3 adalah berbeda secara internal dari struktur mirip MD5 pada SHA-1 dan SHA-2.

Keccak didasarkan pada pendekatan baru yang disebut konstruksi spons. Konstruksi spons didasarkan pada fungsi acak yang luas atau permutasi acak , dan memungkinkan memasukkan ("menyerap" dalam terminologi spons) sejumlah data, dan mengeluarkan ("memeras") sejumlah data, sambil bertindak sebagai fungsi *pseudo random* berkenaan dengan semua masukan sebelumnya. Ini mengarah pada fleksibilitas yang tinggi.



Gambar 1. Konstruksi Spons

Ada dua fase pada algoritma Keccak, yakni.

1. Fase penyerapan (*absorbing*)
   a. Untuk setiap blok masukan *Pi* berukuran *r*-bit, XOR-kan dengan *r*-bit pertama dari *state S*, kemudian hasilnya dimasukkan ke dalam fungsi permutasi *f* untuk menghasilkan *state* baru *S*.
   b. Bila semua blok masukan selesai diproses, konstruksi spons beralih ke fase pemerasan (*squeezing*).
2. Fase pemerasan (*squeezing*)
   a. *Message digest* akan disimpan di dalam *Z*.
   b. Inisialisasi *Z* dengan string kosong (*null string*).
   c. Selagi panjang *Z* belum sama dengan *d*, *r*-bit pertama dari *state S* disambungkan (*append*) ke *Z*.
   d. Jika panjang *Z* masih belum sama dengan *d*, masukkan ke dalam fungsi permutasi *f* menghasilkan *state* baru *S*.

F. Algoritma Elliptic Curve Cryptography (ECC)[5]

Elliptical Curve Cryptography merupakan pendekatan untuk kriptografi kunci publik didasarkan pada aljabar struktur dari kurva eliptik lebih bidang terbatas . ECC memungkinkan kunci yang lebih kecil dibandingkan dengan kriptografi non-EC (berdasarkan bidang Galois biasa) untuk memberikan keamanan yang setara. Kurva eliptik adalah kurva dengan bentuk umum persamaan: $y_2 = x_3 + ax + b$ dengan syarat $4a_3 + 27b_2 \neq 0$. Tiap nilai *a* dan *b* berbeda memberikan kurva eliptik yang berbeda. Kurva eliptik terdefinisi untuk $x,y \in R$. Didefinisikan sebuah titik bernama titik $O(x, \infty)$, yaitu titik pada *infinity*. Titik-titik $P(x, y)$ pada kurva eliptik bersama operasi + membentuk sebuah grup. Himpunan grup: semua titik $P(x, y)$ pada kurva eliptik & Operasi biner : +.

G. Algoritma Elliptic Curve Digital Signature Algorithm (ECDSA)[11]

Dalam protokol ECDSA, pihak yang akan melakukan tanda tangan digital, mempunyai parameter domain kurva eliptik berupa $D = \{q, FR, a, b, G, n, h\}$ dan pasangan kunci kunci rahasia *dA* dan kunci publik *QA*. Kemudian pihak yang akan melakukan verifikasi terhadap tanda tangan, memiliki salinan dokumen D yang otentik dan kunci publik *QA*. Proses-proses yang terjadi adalah sebagai berikut :

1. Pembangkitan Kunci

   Ada beberapa tahapan dalam pembangkitan kunci dengan algoritma ECDSA, yakni.
   1. Memilih sebuah bilangan bulat random *dA*, yang nilainya diantara $[1, n-1]$
   2. Menghitung $QA = dA \bullet G = (x_1, y_1)$
   3. Kunci rahasia = *dA*, dan kunci publik = *QA*.
2. *Signing*

   Dalam proses menandatangani suatu data digital, terdapat beberapa tahap, diantaranya.
   1. Memilih sebuah bilangan bulat random k, yang nilainya diantara [1,n-1].
   2. Menghitung $QA = k \bullet G = (x_1,y_1)$ dan $r = x_1 \bmod n$. Jika $r = 0$, maka kembali ke langkah pertama.
   3. Menghitung $k\text{-}1 \bmod n$
   4. Menghitung $e = Hash(m)$
   5. Menghitung $s = k\text{-}1 \{e+dA \bullet r\} \bmod n$.

   Tanda tangan Alice untuk message m adalah (*r, s*)
3. *Verifying*

   Dalam melakukan verifikasi terhadap data digital yang diberikan, ada beberapa tahapan yang harus dilakukan, diantaranya.
   1. Memverifikasi bahwa *r* dan *s* adalah bilangan bulat yang antara [*1, n-1*]
   2. Menghitung $e = Hash\ (m)$

3. Menghitung $w = s^{-1} \bmod n$
4. Menghitung $u_1 = e*w \bmod n$ dan $u_2 = r*w \bmod n$
5. Menghitung $u_1 \bullet G + u_2 \bullet QA = (x_1, y_1)$
6. Menghitung $v = x_1 \bmod n$
7. Menerima tanda tangan jika dan hanya jika $v = r$

H. Tanda tangan Digital[8]

Tanda-tangan, jika dikaitkan dengan hal penting dalam kriptografi, memenuhi tiga aspek, yakni *data integrity*, *authentication*, dan *non-repudiation*. Tanda-tangan merupakan bukti otentik yang tidak dapat dilupakan, dipindah untuk digunakan ulang, dan tidak dapat disangkal. Dokumen yang telah ditandatangani tidak dapat diubah. Tanda-tangan digital sendiri merupakan tanda-tangan yang digunakan untuk data digital, bukan tulisan tanda-tangan yang di-digitalisasi dengan cara dipindai atau difoto.

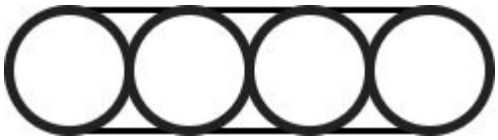
I. Circular Shift[1]

Circular shifting merupakan operasi menyusun ulang seluruh bagian pada suatu *tuple*. *Circular shifting* bisa dengan menggeser semua bagian kecuali bagian pertama ke kiri kemudian memasang bagian pertama di bagian terakhir atau dengan berbagai susunan lainnya, selama menggeser seluruh bagian dalam bentuk melingkar atau *circular*. Dalam pemrograman, *circular shifting* dapat dilakukan untuk menggeser bit-bit atau karakter-karakter pada suatu *string*.

III. Implementasi

A. Implementasi Caterpillar RSA

Algoritma ini berdasarkan ide dari bentuk dan cara gerak sebuah *caterpillar* (roda *tank*). Pada saat bergerak, akan terjadi perputaran dari rantai bagian luar dan bagian dalam. Berikut adalah gambaran kasar dari roda *tank* yang dimaksud.



Gambar 2. Roda *Tank*

Implementasi dari ide tersebut dilakukan pada proses enkripsi dan dekripsi pesan dengan algoritma RSA. Pada saat pengerjaan, ukuran blok yang digunakan adalah 4. Pada saat enkripsi, ketika sebuah blok sudah dihitung nilainya, bila ada hasil perhitungan blok *ciphertext* yang nilainya dalam *integer* < 1000 maka akan dilakukan *padding* angka "0" sampai terdapat 4 digit. Setiap dua karakter dalam blok dilakukan *shifting*, ketika sudah dilakukan *shifting* per dua karakter, selanjutnya dilakukan *circular shifting* sebanyak dua karakter ke kiri untuk setiap karakter di dalam blok. Berikut adalah fungsi yang dibuat.

```
def shift_chars(chars):
    res = ""
    last = len(chars)-2
    for j in range(last):
        res += chars[j+2]
    res += chars[:2]
    return res

def rotate_digits(digit):
    return (digit[1] + digit[0])

def rsa_encrypt(text, key=None, n=None):
    res = ""
    itr = len(text)
    for i in range(0, itr, 4):
        temp = int(text[ i : i+4 ])
        ci_temp = str(count_rsa_enc(temp,
key, n))
        ci_temp = "0"*(4-len(ci_temp)) +
ci_temp
        print(ci_temp)

        res +=
shift_chars(rotate_digits(ci_temp[:2]) +
rotate_digits(ci_temp[2:]))
    return(res)
```

Untuk proses dekripsi dapat menggunakan tahapan yang kurang lebih mirip dengan tahap enkripsi. Pertama, c*iphertext* dibagi ke dalam beberapa blok. Pada setiap blok, pertama dilakukan *circular shifting* sebanyak dua karakter ke kanan. Selanjutnya setiap dua karakter di dalam blok dilakukan *shifting*. Ketika sudah dilakukan *shifting* untuk setiap blok selanjutnya dilakukan ke tahapan dekripsi RSA seperti biasa untuk mendapatkan nilai *plaintext* semula. Berikut merupakan algoritma untuk tahapan dekripsi.

```
def rsa_decrypt(text, key=None, n=None):
    block_list = div_str_to_bits(text, 4)
    print(block_list)
    res = ""
    blocks = []

    for char in block_list:
        shifted_block = shift_chars(char)

blocks.append(rotate_digits(shifted_block
[:2]) + rotate_digits(shifted_block[2:]))
    for char in blocks:
        res += str(count_rsa_dec(char,
key, n))
    return(res)
```

B. Implementasi ECDSA

Implementasi ECDSA dilakukan dengan membuat sebuah kelas ECDSA yang terdiri dari sebuah kelas *ECC*, nilai *g*, dan nilai *n*. Kelas *ECC* sendiri terdiri dari *a* dan *b* sebagai parameter formula kurva, dan sebuah bilangan prima *q*. Inisiasi untuk kelas ECDSA dimulai dengan inisiasi untuk kelas ECC terlebih dahulu.

Pada algoritma yang disusun, parameter kurva yang digunakan adalah a = -1, b = 188, dan q = 751. Selanjutnya, seluruh koordinat yang terdapat pada kurva disimpan pada sebuah variabel. Selanjutnya dilakukan inisiasi kelas ECDSA.

Pertama tentukan sebuah poin(g) pada kurva eliptik yang akan digunakan. Kemudian inisiasi kelas ECDSA dengan kelas ECC sebelumnya dan poin(g) yang ditentukan sebelumnya. Pada implementasi ini poin yang digunakan adalah (581, 20). Pesan yang digunakan pada implementasi ini adalah hasil enkripsi dari *plaintext* dengan algoritma *Caterpillar RSA*.

Terdapat dua tahapan yang bisa dilakukan, yakni *signing* atau *verifying*.

1. *Signing*

   Untuk tahap *signing*, ada beberapa langkah yang akan dilakukan, yakni.

   a. Mengubah pesan(m) ke dalam bentuk heksadesimal
   b. Menghitung nilai fungsi hash dari pesan(m) dengan algoritma SHA-3 (Keccak)
   c. Mempersiapkan akhir dari pesan berupa *tag* "<ds>public_key_tuple</ds>"
   d. Menghitung nilai r dan s dari pesan(m) dengan kunci privat yang dimiliki
   e. Menambahkan nilai r dan s di tengah *tag* yang dibuat sebelumnya sebagai bagian dari pesan(m)

2. *Verifying*

   Untuk tahapan *verifying*, ada beberapa langkah yang dilakukan, yakni.

   a. Mengubah pesan(m) ke dalam bentuk heksadesimal
   b. Menghitung nilai fungsi hash dari pesan(m) dengan algoritma SHA-3 (Keccak)
   c. Melakukan verifikasi dengan fungsi *verify* yang dimiliki dengan parameter *input* r, s, pesan(m), dan kunci publik yang dimiliki
   d. Jika dikembalikan hasil true maka pesan(m) asli, sedangkan jika tidak, maka pesan(m) palsu.

C. Implementasi Caterpillar RSA dan ECDSA pada Pengiriman Pesan Nomor Rekening Berupa JSON

Berikut adalah contoh implementasi kedua algoritma pada pengiriman pesan nomor rekening berupa JSON dengan spesifikasi sebagai berikut.

a. *Caterpillar* RSA
   1. Nilai p = 1553 dan nilai q =1427
   2. Pasangan kunci privat (6845,9989)
   3. Pasangan kunci publik (5,9989)
   4. Nomor rekening = 1192339918

b. ECDSA
   1. Nilai kunci privat = 123
   2. Pasangan kunci publik (30, 236)

1. Pesan sebelum ditambahkan tanda tangan digital namun sudah dienkripsi


```
{
  "senderId": 1,
  "receiverId": 2,
  "sentDate": "20-12-2020",
  "modifiedDate": "20-12-2020",
  "message": "379346933708"
}
```


2. Pesan yang sudah ditandatangani secara digital


```
{
  "senderId": 1,
  "receiverId": 2,
  "sentDate": "20-12-2020",
  "modifiedDate": "20-12-2020",
  "message":
"379346933708\n<ds>354,322</ds>"
}
```


IV. Eksperimen dan Hasil Analisis

Berikut adalah eksperimen dari algoritma yang telah dibuat terhadap beberapa aspek dalam kriptografi, yakni *authentication* dan *data integrity*. Berikut adalah data yang digunakan untuk pengujian

Data setelah enkripsi namun belum ditandatangani


```
{
  "senderId": 1,
  "receiverId": 2,
  "sentDate": "20-12-2020",
  "modifiedDate": "20-12-2020",
  "message": "379346933708"
}
```


Data setelah enkripsi dan tanda-tangan


```
{
  "senderId": 1,
  "receiverId": 2,
  "sentDate": "20-12-2020",
  "modifiedDate": "20-12-2020",
  "message":
"379346933708\n<ds>354,322</ds>"
}
```


Ada beberapa pengujian yang dilakukan yakni.

1. Pengubahan karakter dalam pesan (dekripsi tanpa adanya *signing*)

   a. Penghapusan karakter pada pesan

      Penghapusan karakter dilakukan pada karakter "8" pada *ciphertext*

Data setelah penghapusan karakter pada pesan


```
{
  "senderId": 1,
  "receiverId": 2,
  "sentDate": "20-12-2020",
```

```
  "modifiedDate": "20-12-2020",
  "message": "3793469337088"
}
```


Hasil dekripsi	11923399
Pesan asli	1192339918

b. Penambahan karakter pada pesan
Penambahan karakter dilakukan adalah penambahan *string* "1234" di akhir *ciphertext*

Data setelah penambahan karakter pada pesan


```
{
  "senderId": 1,
  "receiverId": 2,
  "sentDate": "20-12-2020",
  "modifiedDate": "20-12-2020",
  "message": "37934693370881234"
}
```


Hasil dekripsi	11923399181334
Pesan asli	1192339918

2. Penggunaan kunci privat yang tidak sesuai (dekripsi tanpa adanya *signing*)
Kunci privat sebenarnya: (5,9989)
Kunci privat yang digunakan: (70,9989)

Hasil dekripsi	514273242046
Pesan asli	1192339918

3. Verifikasi dengan pengubahan karakter dalam pesan
   a. Penghapusan karakter pada pesan
   Penghapusan karakter dilakukan pada dua karakter akhir *ciphertext*, "08".

Data setelah penghapusan karakter pada pesan


```
{
  "senderId": 1,
  "receiverId": 2,
  "sentDate": "20-12-2020",
  "modifiedDate": "20-12-2020",
  "message":
"3793469337708\n<ds>354,322</ds>"
}
```


Hasil verifikasi	Verified message = 3793469337 Pesan berbeda

b. Penambahan karakter pada pesan
Data setelah penambahan karakter pada pesan


```
{
  "senderId": 1,
  "receiverId": 2,
  "sentDate": "20-12-2020",
  "modifiedDate": "20-12-2020",
  "message":
"37934693370881234\n<ds>354,322</ds>"
}
```


Hasil verifikasi	Verified message = 37934693370812 34 Pesan berbeda

4. Verifikasi dengan pengubahan karakter dalam *digital sign*
Karakter dalam *digital sign* yang diubah dari (354,322) menjadi (354,347)
Data setelah penambahan karakter dalam *digital sign*


```
{
  "senderId": 1,
  "receiverId": 2,
  "sentDate": "20-12-2020",
  "modifiedDate": "20-12-2020",
  "message":
"379346933708\n<ds>354,347</ds>"
}
```


Hasil verifikasi	Verified message = 379346933708 Pesan berbeda

5. Verifikasi dengan penggunaan kunci privat yang tidak sesuai
Kunci privat sebenarnya: (30, 236)
Kunci privat yang digunakan: (70, 236)

Hasil verifikasi	Verified message = 379346933708 Pesan berbeda

6. Verifikasi dengan penghapusan *digital sign*
Data setelah penghapusan *digital sign*


```
{
  "senderId": 1,
  "receiverId": 2,
  "sentDate": "20-12-2020",
  "modifiedDate": "20-12-2020",
  "message": "379346933708"
}
```


Hasil verifikasi	Tanda tangan digital tidak ditemukan

V. Kesimpulan dan Saran Pengembangan

Berdasarkan implementasi dan eksperimen, hasil yang diperoleh menunjukkan bahwa implementasi dengan kedua algoritma yang dibahas sebelumnya dapat memenuhi dua aspek pada kriptografi, yakni *authentication* dan *data integrity*. Algoritma tersebut dapat digunakan untuk mengenkripsi pesan berupa nomor rekening, kemudian menandatanganinya untuk memastikan keaslian pengirim. Ketika ada perubahan pada pesan yang dikirim, algoritma ini dapat mendeteksi perubahan yang dilakukan pada pesan dan atau *digital sign* kemudian setelah dipastikan keaslian pengirim pesan, nomor rekening sebenarnya hanya dapat diketahui bila memiliki kunci yang sesuai. Dengan adanya hal tersebut, keamanan pengiriman data sensitif berupa nomor rekening dapat ditingkatkan dengan lebih baik saat dikirim melalui *messaging platform* terlepas dari sistem pengamanan yang telah disediakan.

Ada beberapa saran pengembangan kedepannya terlepas dari keterbatasan perangkat yang digunakan penulis. Saran untuk bagian enkripsi dengan algoritma *Caterpillar RSA* kedepannya adalah penggunaan nilai *p* dan *q* yang jauh lebih besar sehingga bisa dihasilkan kunci yang lebih panjang dan kompleks dibandingkan yang digunakan saat pengujian , perubahan dari ukuran blok yang digunakan, dan penggunaan algoritma *shifting* yang lebih kompleks. Saran untuk bagian *signing* adalah pembangkitan kunci yang dilakukan dapat menggunakan kunci yang lebih panjang dan dilengkapi dengan algoritma enkripsi lain, misal Elgamal untuk mengenkripsi kunci yang akan dipakai.

VI. Ucapan Terima Kasih

Pertama-tama, penulis mengucapkan syukur kepada Tuhan Yang Maha Esa atas berkat dan anugerah-Nya, penulis dapat menempuh pendidikan di Institut Teknologi Bandung, khususnya di Program Studi Teknik Informatika dan dapat menyelesaikan penulisan makalah IF4020 Kriptografi ini secara tepat waktu.

Penulis juga berterima kasih kepada kedua orang tua dan keluarga penulis yang senantiasa memberikan doa dan bantuan sampai saat ini. Penulis juga ingin mengucapkan terima kasih yang sebesar-besarnya kepada Dr. Ir. Rinaldi Munir, M.T. selaku dosen mata kuliah IF4020 Kriptografi yang telah memberikan banyak ilmu dan bimbingan dalam pembelajaran kriptogragi.

Tidak lupa pula penulis ingin mengucapkan banyak terima kasih kepada teman-teman UNIX 2017, khususnya teman-teman dari K3 pada grup “OTW Seminar TA Gan": Irfan, Jofiandy, Abda, Juniardi, Faiz, Asif, Hanif, dan Hamzah, ; teman-teman pada grup tanpa nama: Vincent, Josep, Suhailie, Vijjasena, Nixon, dan Nano; dan adik tingkat penulis, Ricky, yang senantiasa memberikan semangat dan motivasi bagi penulis, terutama saat penyusunan makalah ini.

Referensi

[1] A. N. Maslov, "Cyclic shift operation for languages", Problems of Information Transmission 9:333–338, 1973

[2] Dworkin, Morris J. (August 4, 2015). "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions". *Federal Inf. Process. STDS. (NIST FIPS) – 202*.

[3] Menezes, A. J., Van, O. P. C., & Vanstone, S. A. (1997). Handbook of applied cryptography. Boca Raton: CRC Press.

[4] Munir, R. (2020). Algoritma RSA. Retrieved from http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Algoritma-RSA-2020.pdf

[5] Munir, R. (2020). Elliptical Curve Cryptography (ECC) (Bagian 2). Retrieved from http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/ECC-2020-Bagian2.pdf

[6] Munir, R. (2020). Fungsi hash SHA-3 (Keccak). Retrieved from http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/SHA-3-2020

[7] Munir, R. (2020). Pengantar Kriptografi (2020). Retrieved from http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Pengantar-Kriptografi-(2020).pdf

[8] Munir, R. (2020). Tanda-tangan Digital. Retrieved from http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Tanda-tangan-digital-2020.pdf

[9] Myer, M. (2016). Messaging as a Platform. Retrieved from https://medium.com/quiq-blog/messaging-as-a-platform-bb5b1071cc7f

[10] Schneier, Bruce, 1963-. (1996). Applied cryptography : protocols, algorithms, and source code in C. New York :Wiley,

[11] Triwinarko, A. Elliptic Curve Digital Signature Algorithm (ECDSA) Departemen Teknik Informatika ITB. *Institut Teknologi Bandung*.

[12] Tunggal A. T., (2020). What is Sensitive Data?. Retrieved from https://www.upguard.com/blog/sensitive-data

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.

Jambi, 21 Desember 2020

Winston Wijaya 13517018