Studi dan Aplikasi *Verifiable Ring Signature* dengan Menggunakan Fungsi *Hash* SHA-256

Kaisar Siregar - 13506072

Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jl. Ganeca No. 10, Bandung, Jawa Barat
e-mail: kaisar.siregar@gmail.com

ABSTRACT

Tanda tangan digital (*Digital Signature*) adalah sebuah metode dalam dunia kriptografi yang berguna untuk menjaga keotentikan, integritas, dan nirpenyangkalan suatu dokumen digital, sehingga mencegah adanya pemalsuan atau modifikasi dari pihak luar yang tidak berwenang. Salah satu metode tanda tangan digital yang ada adalah. *Ring Signature*.

***Ring Signature* yang dapat digunakan oleh salah satu anggota kelompok pengguna yang berada dalam sebuah lingkar kerahasiaan. Untuk menandatagani sebuah dokumen dengan *Ring Signature* diperlukan kunci privat dari individu yang mengirim dan semua kunci publik dari anggota lingkar kerahasiaan tersebut.**

Salah satu karakteristik unik dari *Ring Signature* adalah pihak penerima mengetahui bahwa pesan tersebut dikirim oleh sekumpulan orang yang tergabung dalam lingkar kerahasiaan tersebut, namun tidak bisa mengetahui siapakah individu yang mengirimkan pesan tersebut. Hal ini kadang dapat digunakan secara elegan bagi seseorang untuk membocorkan suatu rahasia grup tersebut.

Untuk menghadapi penyalahgunaan tersebut, pada tahun 2003, Jiqiang LV dan Xinwei Wang mengajukan sebuah metode penandatanganan digital, yaitu *Verifiable Ring Signature*. Dalam aplikasinya *Verifiable Ring Signature* membutuhkan sebuah fungsi *hash* yang bebas dari *collision*. Dalam aplikasi kali ini, penulis memilih untuk mengunakan fungsi *hash* SHA-256, yang merupakan varian dari SHA2, karena sampai sekarang belum ditemukan adanya *collision* pada fungsi tersebut.

Kata Kunci: *Ring Signature*, SHA-256, Verifiable *Ring Signature*

A. PENDAHULUAN

Sesuai dengan namanya, sebuah tanda tangan digital memiliki karakteristik sebuah tanda tangan pada umumnya, yaitu:

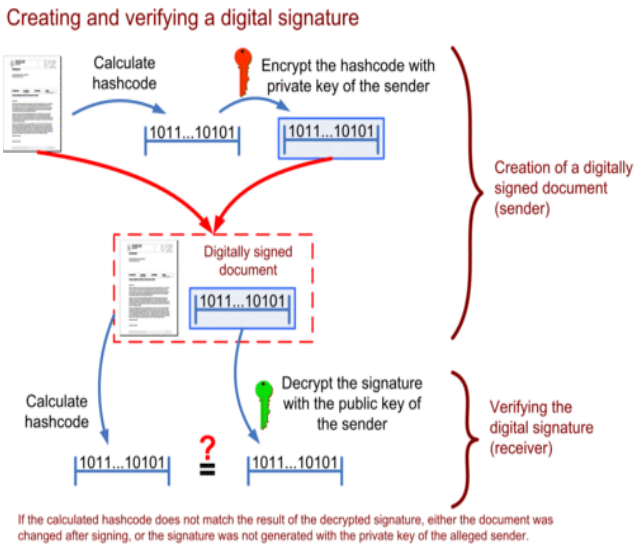
- Tanda-tangan adalah bukti yang otentik.
- Tanda tangan tidak dapat dilupakan.
- Tanda-tangan tidak dapat dipindah untuk digunakan ulang.
- Tanda-tangan tidak dapat disangkal(*repudiation*).

Namun di luar karakteristik tersebut, tanda tangan digital memiliki sejumlah keunikan dibandng tanda tangan pada dokumen biasa, karakteristik unik yang ada antara lain:

- Tanda-tangan digital digunakan khusus untuk data digital
- Tanda-tangan digital bukanlah tulisan tanda-tangan yang di-digitisasi (di-*scan*).
- Tanda-tangan digital adalah nilai kriptografis yang bergantung pada isi pesan dan kunci.
- Tanda-tangan pada dokumen cetak selalu sama, apa pun isi dokumennya.
- Tanda-tangan digital selalu berbeda-beda antara satu isi dokumen dengan dokumen lain.

Pembangkitan tanda tangan digital dapat dilakukan dengan dua cara yaitu dengan enkripsi pesan atau Menggunakan fungsi *hash* (*hash function*) dan kriptografi kunci-publik Cara pertama sudah tidak umum digunakan karena kurang efektif. Namun cara kedua selain praktis digunakan, keaslian dokumen yang bersangkutan akan lebih terjamin pula.

Berikut adalah skema pengunaan fungsi *hash* (*hash function*) dan kriptografi kunci-publik dalam membangkitan sebuah tanda tangan digital



Gambar 1 1Skema Alur Penggunaan Fungsi Hash dan Kriptografi Kunci-Publik dalam Memangkitkan Sebuah Tanda Tangan Digital

Keotentikan ini dijelaskan sebagai berikut:

Apabila pesan *M* yang diterima sudah berubah, maka *MD'* yang dihasilkan dari fungsi *hash* berbeda dengan *MD* semula. Ini berarti pesan tidak asli lagi.

Apabila pesan *M* tidak berasal dari orang yang sebenarnya, maka *message digest MD* yang dihasilkan dari persamaan 3 berbeda dengan *message digest MD'* yang dihasilkan pada proses verifikasi (hal ini karena kunci publik yang digunakan oleh penerima pesan tidak berkoresponden dengan kunci privat pengirim).

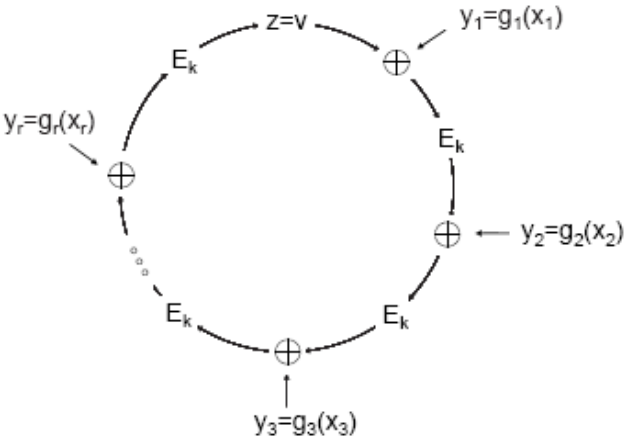
Bila *MD = MD'*, ini berarti pesan yang diterima adalah pesan yang asli (*message authentication*) dan orang yang mengirim adalah orang yang sebenarnya (*user authentication*).

Dengan banyaknya kombinasi fungsi hash dan algoritma kunci publik yang ada, maka banyak sekali metode-metode penandatanganan digital yang mungkin dilakukan. Tanda tangan digital itu sendiri dapat berlaku untuk sebuah individu, seperti *DSA, Poitcheval-Stern Signature Algorithm*, dan *Rabin Signature Algorithm* atau berlaku untuk sebuah kelompok orang, seperti *Group Signature* dan *Ring Signature*

B. RING SIGNATURE

Dalam dunia *digital signature*, *ring signature* adalah tanda tangan digital yang bisa dibangkitkan oleh salah satu anggota dari sebuah kelompok orang (lingkar kerahasiaan) yang masing memiliki sebuah kunci privat dan sebuah kunci publik yang diketahui oleh seluruh anggota lingkar kerahaiaan tersebut. Pesan yang ditandatangani oleh *ring signature* tersebut akan mengotentikasi bahwa pesan tersebut benar-benar dari lingkar kerahasiaan tersebut.

Ring Signature berbeda dengan metode lain yaitu *Group Signaure*, di mana *Ring Signature* tidak memerlukan adanya *group manager*, koordinasi, ataupun persiapan yang panjang. Oleh karena itu dapat dikatakan bahwa *Ring Signature* bersifat lebih cepat, efisien, dan spontan. *Ring Signature* ini sendiri dikemukakan oleh Ron Rivest, Adi Shamir, dan Yael Tauman pada tahun 2001 dan dipublikasikan pertama kali pada ASIACRYPT (sebuah konferensi mengenai riset di bidang kriptografi). Nama *Ring Signature* itu sendiri berasal dari struktur algoritmanya yang berbentuk seperti cincin.



Gambar 2 1 Ring Signature

Skema dari pembangkitan *Ring Signature* oleh anggota *s* dengan pesan *m* dan sekumpulan kunci publik anggota P_1, P_2, ..., P_r (tiap Pi adalah kunci publik RSA,sehingga $P_i = (n_i, e_i)$) serta sebuah kunci privat S_s (yang diperlukan untuk mengkomputasi g_s^{-1}) dengan banyak anggota *r* adalah sebagai berikut:

1. Menentukan Kunci Simetris dari Pesan:
Penanda tangan mengkomputasi kunci simetris k dari pesan m dengan sebuah fungsi hash *h.*

$$k = h(m) \qquad \textbf{(2.1)}$$

2. Memilih sebuah *glue value* acak:
Penanda tangan mengkomputasi nilai inisialisasi (atau *glue*) v secara acak:

$$v = \{0,1\}^b \qquad \textbf{(2.2)}$$

dimana b adalah nilai perputaran permutasi (besar block dalam bit) pada algoritma kunci simetri yang akan dipakai nanti.

3. Memilih nila X_i acak:
Penanda tangan memilih nilai X_i untuk setiap anggota lainnyadengan $1 \leq i \leq r$ di mana $i \neq s$, yang bernilai acak dari $\{0, 1\}^b$, kemudian mengkomputasi:

$$y_i = g_i(x_i) \qquad \textbf{(2.3)}$$

g(m) adalah sebuah fungsi *trapdoor permutation* yang didefinisikan sebagai berikut:

$$g(m) = \begin{cases} qn + (p^{e_i} \bmod (n_i)) & ,(q+1)n \leq 2^b \\ m & ,(q+1)n > 2^b \end{cases}$$

(2.4)

q dan p adalah bilangan bulat positif sedemikian rupa sehingga m = qn + p dan $0 \leq p \leq n$.

4. Menemukan nila y_s:
Penanda tangan mengkalkulasikan fungsi berikut untuk menemukan y_s.

$$C_{k,v}(y_1, y_2, \ldots, y_r) = v \qquad \textbf{(2.5)}$$

Fungsi kombinasi C*k,v* haruslah memenuhisyarat di mana hanya akan terdapat sebuah nilai y_s yang memenuhi persamaan di atas dan dapat dikomputasi secara efisien. Salah satu fungsi yang memenuhi adalah

$$C_{k,v}(y_1, y_2, \ldots, y_r) = E_k(y_r \oplus E_k(y_{r-1} \oplus E_k(y_{r-2} \oplus E_k(\ldots \oplus E_k(y_1 \oplus v)\ldots))))$$
(2.6)

Di mana E_k adalah fungsi enkripsi kunci simetris dengan kunci K.

5. Menginvers fungsi *trapdoor permutation* untuk mendapatkan X_s:
Penanda tangan mendapatkan nilai X_s nya dengan menginvers fungsi *trapdoor permutation*:

$$X_s = g_s^{-1}(y_s) \qquad \textbf{(2.7)}$$

6. Membangkitkan *Ring Signature*:
Penanta tangan kemudian membangkitan *Ring Signature* dari nilai-nilai yang telah didapat. Sehingga dihasilkan tupple sebanyak (2r + 1) yang berisi

$$(P_1, P_2, \ldots, P_r;\ v;\ X_1, X_2, \ldots, X_r) \qquad \textbf{(2.8)}$$

Dalam proses tersebut terlihat bahwa diperlukan suatu metode enkripsi kunic simetris (biasanya digunakan AES atau Rijndael) dan sebuah fungsi *hash* untuk membangkitkan sebuah tanda tangan digital *Ring Signature*. Sedangkan untuk proses memverifikasi sebuah dokumen yang telah dibubuhi *Ring Signature* $(P_1, P_2, \ldots, P_r;\ v;\ X_1, X_2, \ldots, X_r)$ maka proses yang dilakukan adalah sebagai berikut:

1. Melakukan *trapdoor permutation*:
Langkah pertama yang harus dilakukan pihak penerima dokumen adalah menghitung nilai y_i sesuai dengan fungsi *trapdoor permutation* (formula 2.3) untuk tiap nilai $1 \leq I \leq r$.

2. Menghitung nilai *hash* dari pesan yang diterima:
Penerima dokumen kemudian menghitung nilai *hash* dari dokumen yang diterima sesuai dengan fungsi *hash* yang diaplikasikan pada saat pembangkitan tanda tangan digital (formula 2.1)

3. Memverifikasi keabsahan dokumen:
Penerima dokumen pada akhirnya memverifikasi keabsahan dokumen yang diterima dengan menghitung persamaan dasar *Ring Signature* (formula 2.5). Apabila persamaan tersebut terpenuhi, maka dokumen tersebut terbukti keabsahannya.

Dari proses tersebut terlihat bahwa karakteristik unik dari Ring Signature adalah pihak penerima mengetahui bahwa pesan tersebut dikirim oleh sekumpulan orang yang tergabung dalam lingkar kerahasiaan tersebut, namun tidak bisa mengetahui siapakah individu yang mengirimkan pesan tersebut. Hal ini kadang dapat digunakan secara elegan bagi seseorang untuk membocorkan suatu rahasia grup tersebut. Misal seseorang dari KPU membocorkan informasi penting terkait dengan pemilu kepada seseorang. Sang penerima akan yakin bahwa informasi tersebut berasal dari KPU karena ditandatangani oleh Ring Signature KPU. Namun apabila dokumen tersebut ditemukan pihak berwenang, maka dokumen tersebut tidak dapat dijadikan barang bukti karena identitas sang penerima tidak bisa diketahui secara langsung.

Oleh karena itu ada baiknya apabila pihak verifikasi juga dapat mengidentifikasi sang pengirim secara langsung dari tanda tangan digital yang diimbuhkan. Hal ini pada tahun 2003 diterapkan pada sebuah metode pengembangan *Ring Signature* yang bersama *Verifiable Ring Signature*.

C. VERIFIABLE RING SIGNATURE

Pada tahun 2003, Jiqiang LV dan Xinwei Wang dari Universitas Xidian di Cina mengajukan sebuah metode penandatanganan digital, yaitu *Verifiable Ring Signature*. Metode ini hampir sama dengan *Ring Signature* biasa

namun identitas dari pengirim dapat diverikasi dengan menggunakan metode pembangkitan kunci khusus. Metode ini sebenarnya didasari oleh teori yang dikemukakan oleh Ron Rivest dkk. pada saat mengembangkan *Ring Signature.*

Rivest dkk. menyatakan bahwa apabila nilai X_i dibangkitkan tidak dengan acak, melainkan acak semu (*pseudo-random*) sedemikian sehingga mungkin untuk memverifikasi individu yang mengirimkan *Ring Signature* tersebut. Wang dan LV kemudian menambahkan karakteristik ini ke dalam *Verifiable Ring Signature*, sehingga selain memiliki karakteristik dari *Ring Signature*, *Verifiable Ring Signature* juga memiliki karakteristik unik, yaitu pihak penerima dapat mengetahui siapakah pengirim dokumen tersebut.

Proses pembangkitan tanda tangan digital pada *Verifiable Ring Signature*serupa dengan proses pembangkitan tanda tangan digital pada *Ring Signature*, namun terdapat sedikit perbedaan khususnya pada saat menentukan nilai X_i Proses pembangkitan tanda tangan digital pada *Verifiable Ring Signature* oleh anggota As dengan pesan *m* dengan banyak anggota *r* adalah sebagai berikut:

1. Tahap inisialisasi

Tahap inisialisasi ini hanya dilakukan satu kali pada saat pembangkitan kunci untuk masing-masing anggota. Setiap anggota, A_i melakukan hal sebagai berikut:

Pilih sebuah bilangan prima besar p_i yang sulit dikomputasi oleh logaritma diskrit dalam fungsi $GF(p_i)$, sebuah bilangan q_i yang merupakan bilangan prima pembagi $p_i - 1$, sebuah bilangan o_i yang merupakan bilangan prima pembagi q_i -1 dan g_i adalah basis dari $GF(p_i)$ yang pangkatnya adalah q_i. Kunci privat A_i adalah X_{Ai} yang memenuhi $X_{Ai} < q_i$ dan kunci publik A_i adalah (y_{Ai}, p_i, q_i, g_i) di mana:

$$y_{Ai} = g_i^{X_{Ai}} \; m \quad \textbf{(3.1)}$$

2. Menentukan Kunci Simetris dari Pesan:

Penanda tangan mengkomputasi kunci simetris k dari pesan m dengan sebuah fungsi hash *h.*

$$k = h(m) \quad \textit{(3.2)}$$

3. Memilih sebuah *glue value* acak:

Penanda tangan mengkomputasi nilai inisialisasi (atau *glue*) v secara acak:

$$v = \{0,1\}^b \quad \textit{(3.3)}$$

dimana b adalah nilai perputaran permutasi (besar block dalam bit) pada algoritma kunci simetri yang akan dipakai nanti.

4. Memilih nilai (α_i, β_i) acak:

Penanda tangan A_s memilih nilai (α_i, β_i) untuk setiap anggota lainnyadengan $1 \leq i \leq r$ di mana $i \neq s$, yang masing-masing bernilai acak dari $\{0, 1\}^b$, kemudian mengkomputasi:

$$y_i = g_i(\alpha_i, \beta_i) \quad \textit{(3.4)}$$

$g_i(m)$ adalah sebuah fungsi *trapdoor permutation* yang didefinisikan sebagai berikut:

$$g_i(\alpha, \beta) = \alpha \, . \, y_{A_i}^{\alpha^*} . \, g_i^{\beta} \; mod \; p_i \quad \textit{(3.5)}$$

5. Menemukan nila y_s:

Penanda tangan mengkalkulasikan fungsi berikut untuk menemukan y_s.

$$C_{k,v}(y_1, y_2, \ldots, y_r) = v \quad \textbf{(3.6)}$$

Fungsi kombinasi C*k,v* haruslah memenuhi syarat di mana hanya akan terdapat sebuah nilai y_s yang memenuhi persamaan di atas dan dapat dikomputasi secara efisien. Salah satu fungsi yang memenuhi adalah formula 2.6.

6. Menginvers fungsi *trapdoor permutation* untuk mendapatkan (α_s , β_s):

Penanda tangan mendapatkan nilai X_s nya dengan menginvers fungsi *trapdoor permutation* $^{-1}(y) = (\alpha, \beta)$ di mana::

$$\alpha \equiv y . g_i^{-K . g_i^K} \bmod p_i \quad \textbf{(3.7)}$$

$$\alpha^* = \alpha \, m \quad \textbf{(3.8)}$$

$$\beta = x_{A_i} \; \alpha - K . g_i^K \, m \quad \textbf{(3.9)}$$

K adalah sebuah bilangan bulat acak yang memenuhi K < O_i

7. Membangkitkan *Ring Signature*:

Penanta tangan kemudian membangkitan *Ring Signature* dari nilai-nilai yang telah didapat. Sehingga dihasilkan tupple sebanyak (2r + 1) yang berisi

$$(P_1, P_2, \ldots, P_r; \; v; \; (\alpha_1, \beta_1), (\alpha_2, \beta_2) \ldots . \; (\alpha_s, \beta_s))$$

(3.10)

Untuk proses verifikasi *Verifiable Ring Signature*, pada dasarnya sama dengan *Ring Signature*, hanya saja pada saat menghitung nilai y_i kita mnggunakan formula 3.4, dan bukan 2.3. Untuk kasus *Verifiable Ring Signature* terdapat sebuahmetode tambahan yaitu *Signature Verification* yan berfungsi untuk memvalidasi individu pengirim dokumen tersebut. Adapun prosesnya adalah seperti berikut:

1. Mendapatkan nilai g_s^K

Penandatangan mengirimkan secara rahasia sebuah bilangan g_s^K ke pihak penerima.

2. Memvalidasi nilai g_s^K yang telah didapat

Dari nilai g_s^K yang telah didapat, pihak penerima kemudian mengecek apakah nilai tersebut memenuhi persamaan berikut:

$$\alpha_s \equiv y.(g_s{}^K)^{-g_s{}^K} m \qquad \textbf{(3.11)}$$

Apabila persamaan tersebut terpenuhi, maka validasi pengirim dokumen tersebut berhasil.

4. FUNGSI *HASH* SHA-256

Seperti yang telah dijabarkan pada bagian C, pengaplikasian teknik penandatanganan digital *Ring Signature (*dan juga *Verifiable Ring Signature* tentunya*)* memerlukan suatu fungsi *hash* yang bebas dari *collision*. Salah satu fungsi *hash* memenuhi syarat tersebut adalah fungsi *hash* SHA-256.

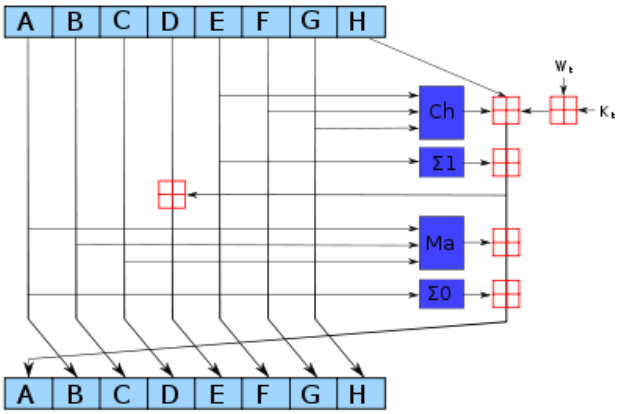
Algoritma Hash SHA merupakan salah satu dari 5 algoritma hash dalam kriptografi yang diciptakan oleh National Security Agency (NSA) dan dipublikasikan oleh NIST sebagai Amerika Serikat Federal Information Processing Standard. SHA merupakan singkatan dari Secure Hash Algorithm. Algoritma hash melakukan proses terhadap sebuah masukan dengan panjang berapapun dan menghasilkan sebuah keluaran dengan panjan yang tetap tanpa terpengaruh dari panjang masukan yang diterima. Keluaran dalam fungsi hash serting disebut message digest. Perubahan yang sangat kecil pada pesan masukan akan menghasilkan message digest yang berbeda. Suatu algoritma hash dikatakan aman jika:

1. Untuk menemukan pesan masukan dari suatu message digest membutuhkan waktu yang sangat lama dari segi komputasi.
2. Untuk menemukan 2 pesan masukan yang berbeda yang menghasilkan message digest yang sama membutuhkan waktu yang sangat lama dari segi komputasi.

Algorithm and variant		Output size (bits)	Rounds	Operations	Collisions found
SHA-0		160	80	+,and,or,xor,rot	Yes
SHA-1		160	80	+,and,or,xor,rot	None (2^{63} attack)
SHA-2	*SHA-256/224*	256/224	64	+,and,or,xor,shr,rot	None
	SHA-512/384	512/384	80	+,and,or,xor,shr,rot	None

Gambar 4 1 Perbandingan berbagai algoritma SHA

SHA-256, yang merupakan salah satu varian dari SHA 2 menghasilkan message digest dengan panjang 256 bit. Hal ini menyebabkan algoritma hash SHA-256 lebih aman dari *collision* dibandingkan SHA 1 Berikut ini skema round yang dipakai pada SHA-256.



Gambar 4 2 Fungsi Hash SHA-256

SHA-256 terdiri atas 64 kali putaran dan menggunakan 8 buah 32 bit register berbeda dengan SHA-1 yang menggunakan 5 buah 32 bit register dan terdiri atas 80 kali puaran. Pemrosesan pesan dilakukan dalam blok yang panjangnya 512 bit. Jadi pesan yang bertugas sebagai masukan dibagi ke dalam blok – blok sepanjang 512 bit. Selain itu operasi yang dilakukan pada SHA-256 ini lebih banyak dan lebih kompleks dibandingkan dengan operasi yang dilakukan di SHA-1. Penambahan konstanta juga dilakukan dalam SHA-256.

D. APLIKASI VERIFIABLE RING SIGNATURE

Berikut adalah beberapa fungsi untuk mengimplementasikan tiap tahap *Verifiable Ring Signature* yang dibuat oleh penulis.

1. Tahap Inisialisasi

```
public Key() //Konstruktor sekaligus
pembangkit key
 {
    Random rd = new
Random(System.DateTime.Now.Millisecond)
;
    BigInt iter,oValue;
    BigInt zero = new BigInt(0);
    Boolean done = false;
    Boolean qdone = false;

    while (!done)
    {
      do
      {
        pValue = new
BigInt((ulong)rd.Next(100000,
1000000));

       } while
(!pValue.isProbablePrime());
       qdone = false
       iter = new BigInt(pValue / 2);
       for (; ; )
       {
         if (iter < 2)
           break;
         else
         {
          if ( (((pValue-1) % iter) ==
zero) && (iter.isProbablePrime())) //
apabila habis membagi (p-1) dan prima
          {
            qValue = new BigInt(iter);
            qdone = true;
            break;
          }
         }
         iter--;
       }
       if (qdone)
       {
         iter = new BigInt(qValue /
2);
          for (; ; )
          {
           if (iter < 2)
             break;
           else
           {
            if ((((qValue - 1) % iter)
== zero) && (iter.isProbablePrime()))
// apabila habis membagi (p-1) dan
prima
              {
                oValue = new
BigInt(iter);

                done = true;
                break;
              }
             }
              iter--;
            }
           }
         }
         //Setelah nilai p dan q didapat
         xValue = new
BigInt((ulong)rd.Next(1, oValue));
//Mendapatkan kunci privat
        gValue = new BigInt(GF(pValue,
qValue)); //Mendapatkan nilai g
         yValue =
Math.Pow(gValue,xValue) mod pValue; //
Menghitung nilai y

   }
```

Fungsi di atas akan menghasilkan sebuah set kunci publik dan kunci privat untuk satu orang yang nantinya akan digunakan untuk membangkitkan tanda tangan digital.

2. Menentukan Kunci Simetris dari Pesan:

```
public byte[] hashing(byte[] data)
{
  SHA256 sha256 = new SHA256Managed();
  return sha256.ComputeHash(data);
}
```

Dengan menggunakan fungsi hash SHA-256 yang telah diintegrasi dalam library System.Security.Cryptography milik .NetFramework maka nilai *hash* dari dokumen dapat dikomputasi dengan mudah.

3. Memilih Nilai Glue Secara Acak

```
public byte[] genGlue()
{
  Random rd = new
Random(System.DateTime.Now.Millisecond)
;
  ulong glue = rd.Next(0,(int)
Math.Pow(2,256));
   return toByteArray(glue);
```

```
}
```

Dari fungsi tersebut didapatkan nilai inisialisasi v, yang dibutuhkan dalam pembangkitan tanda tangan digital.

4. Memilih nilai (α_i, β_i) acak dan menghitung nilai y:

```
public void genYValue(BigInt alpha,
BigInt beta, BigInt alphastar)
 {
  yValue = alpha * Math.Pow(yValue,
alphastar) * Math.Pow(gValue, beta) %
pValue;
  }
```

5. M enemukan nila y_s:

```
public BigInt genYsValue(Key[] keys,
byte[] data, byte[] v, int s)
 {
   Rijndael aes = Rijndael.Create();
   BigInt iter = new BigInt(0);
   aes.GenerateIV();
   aes.BlockSize = 256;
   while (true)
   {
     for (int i = 0; i < keys.Length;
i++)
     {
       FileStream fs =
File.Open("result.txt",
FileMode.OpenOrCreate);
        if(i==s)
        {
          aes.Key = toByteArray(iter);
          iter++;
         }
         else
           aes.Key = keys[i].yValue;
         CryptoStream cs = new
CryptoStream(fs, aes.CreateEncryptor(),
CryptoStreamMode.Write);
         StreamWriter sw = new
StreamWriter(cs);
          sw.WriteLine(data);
          sw.close(); cs.Close();
fs.Close();
          fs = File.Open("result.txt",
FileMode.Open);
          StreamReader sr = new
StreamReader(fs);
          data =
toByteArray(sr.ReadToEnd());
          sr.Close(); fs.Close();
        }
       if (data.Equals(v))
           return iter;
      }
 }
```

6. Menginvers fungsi *trapdoor permutation* untuk mendapatkan (α_s , β_s):

```
public BigInt genAlphaS()
{
  Random        rd        =        new
Random(System.DateTime.Now.Millisecond)
;
   k          =          rd.Next(0,(int)
Math.Pow(2,oValue));
   alpha   =   new   BigInt((yValue   *
Math.Pow(gValue,          (-k          *
Math.Pow(gValue,k))) % pValue ));
    return alpha;
}

 public   BigInt   genAlphaStarS(BigInt
alpha)
 {
   return new BigInt(alpha & pValue);
 }

  public BigInt genBetaS(BigInt alpha,
BigInt k)
  {
    return new BigInt(xValue * alpha -
k * Math.Pow(gValue, k) % qValue);
   }
```

Ketiga fungsi tersebut akan menghasilkan kunci privat (α_s , β_s) dan juga nilai α_s^*. Setelah semua fungsi yang ada dijalankan, makaakan dihasilkan sebuah *Verifiable Ring Signature* sesuai dengan dokumen yang ada.

Untuk proses verifikasi kita hanya tinggal menggunakan fungsi-fungsi yang telah digunakan pada proses pembangkitan tanda tangan dan mencocokannya dengan nilai tanda tangan yang ada pada dokumen. Namun untuk pengecekan indivdu yang mengirim diperlukan satu fungsi tambahan, yaitu:

```
public bool isSenderValid(Key key,
BigInt gs, BigInt k)
{
 return key.alphaS.Equals(key.yValue *
Math.Pow(Math.Pow(gs, k), Math.Pow(-gs,
k)));
 }
```

Apabila fungsi tersebut bernilai *true*, maka kita telah berhasil mengidentifikasi sang penanda tangan. Dengan aplikasi *Verifiable Ring* Signature di atas kita dapat membuktikan metode tersebut dapat memverivikasi kelompok dan individu yang secara langsung mengirim dokumen tersebut. Fungsi-fungsi tersebut telah teruji dengan jumlah anggota sebanyak 6 orang. Untuk jumlah anggota yang lebih dari itu, maka diperlukan waktu komputasi yang cukup lama. Hal ini kemungkinan besar dikarenakan proses pembangkitan kunci tiap anggota dan juga kalkulasi fungsi komposisi yang cukup rumit.

E. Kesimpulan

- *Verifiable Ring Signature* dapat memvalidasi sebuah dokumen yang dikirim oleh sekelompok orang yang terdapat dalam sebuah lingkar kerahasiaan. Namun tidak seperti *Ring Signature* biasa, *Verifiable Ring Signature* juga memvalidasi individu yang mengirimkan dokumen tersebut.
- Karakteristik tersebut dibuktikan oleh program aplikasi *Verifiable Ring Signature* yang dibuat oleh penulis dengan menggunakan bahasa C# pada *environment* .NetFramework.

DAFTAR PUSTAKA

[1] Munir, Rinaldi, Diktat *Kuliah IF5054 Kriptografi*, Penerbit ITB 2006

[2] Rivest, Ronald L. et al. *How to Leak a Secret: Theory and Applications of Ring Signature.* 2001

[3] LV, Jiaqiang dan Wang, Xinwei. *Verifiable Ring Signature.* 2003

[4] Zufri, Ahmad, *Implementasi Secure Remote Password (SRP) dengan Modifikasi Penerapan Algoritma Hash SHA-256.* 2008