Studi dan Analisis Skema Benaloh untuk Pembagian Rahasia dengan Verifikasi beserta Studi dan Implementasi Skema Ambang Shamir

Michell Setyawati Handaka / 135 08 045
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
If18045@students.if.itb.ac.id

Terkadang diperlukan suatu mekanisme untuk membagi suatu rahasia kepada sekelompok orang tertentu sejumlah* n *yang mana untuk dapat merekonstruksi ulang rahasia tersebut dibutuhkan sedikitnya* t *bagian dari rahasia yang dibagikan, dimana* t ≤ n*. Kendatipun demikian, terkumpulnya* m *buah bagian, dimana* m < t*, tidak akan memberikan informasi tambahan apapun dibandingkan dengan tidak adanya kepemilikan bagian sama sekali. Salah satu skema pembagian rahasia yang banyak digunakan adalah skema Shamir dengan ambang.

Permasalahan berikutnya yang muncul dengan skema pembagian rahasia ini adalah bahwa anggota kelompok mungkin saja tidak saling mengetahui satu sama lainnya sehingga verifikasi keotentikan dari kepemilikan bagian dari rahasia menjadi sulit, karena dapat saja seseorang berpura-pura memiliki bagian rahasia untuk mendapatkan bagian milik orang lain dalam kelompok.

Skema pembagian rahasia yang dapat memastikan bahwa tidak ada partisipan yang berpura-pura dalam kepemilikan bagian rahasia disebut sebagai skema pembagian rahasia dengan verifikasi.

Skema pembagian rahasia dengan verifikasi lainnya adalah skema yang memungkinkan partisipan untuk mengecek integritas daripada* dealer *itu sendiri.

Salah satu skema pembagian rahasia dengan verifikasi adalah skema Benaloh. Pada makalah ini akan dicoba untuk dilakukan studi analisis terhadap skema tersebut dan pengimplementasian dari skema ambang Shamir dengan menggunakan platform .NET.

Benaloh scheme, secret sharing, Shamir treshold scheme, verifiable secret sharing

I. Studi : Pembagian Data Rahasia – Skema Ambang Shamir

A. Terminologi

Terdapat beberapa terminologi istilah yang digunakan secara umum pada skema pembagian data rahasia, yang antara lain adalah seperti yang disebutkan berikut ini, yakni :

- *Secret*
  adalah merupakan informasi rahasia yang direpresentasikan sebagai sebuah *integer M*.
- *Share*
  merupakan hasil pembagian *secret*.
- *Dealer*
  adalah pihak yang melakukan pembagian *secret* menjadi *share-share*.
- *Participant*
  adalah *n* pihak yang memperoleh *share* yang berbeda satu sama lainnya.

B. Skema Ambang Shamir

Skema yang ditemukan oleh Shamir (1979) ini memungkinkan konstruksi *secret* dari *t* buah *share* yang mana pada mulanya *secret* tersebut dibagikan kepada *n* orang *participant*, sehingga terdapat *w* buah *share* yang berbeda satu sama lainnya, dimana berlaku $t \leq w = n$. Dengan kata lain, skema ini merupakan suatu metode pembagian *secret M* menjadi *w* buah *share* sedemikian sehingga sebarang himpunan bagian dari *w* yang terdiri atas sedikitnya *t* dapat merekonstruksi ulang *M*, tetapi tidak jika kurang dari *t*. *t* adalah merupakan nilai ambang untuk rekonstruksi.

Ide dari skema ini berasal dari persoalan interpolasi :

> *untuk membentuk polinomial* $y = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$ *diperlukan sedikitnya (n + 1) buah titik*

II. Implementasi : Pembagian Data Rahasia – Skema Ambang Shamir

A. Algoritma *Pembangkitan* Share

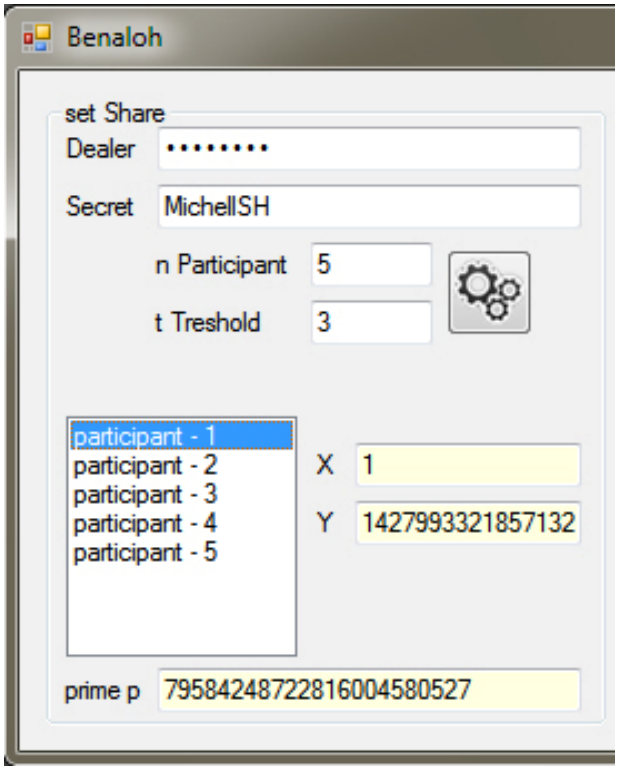
Tahapan-tahapan untuk melakukan pembagian *share* sejumlah *w* untuk *n* buah *participant* dengan ambang rekonstruksi *t* dari *secret M* dapat dijabarkan sebagai berikut ini :

- Bangkitkan bilangan prima acak *p* yang memenuhi kondisi $p > M$ dan $p > n$, *p* tidak rahasia.
- Bangkitkan (*t* - *1*) buah bilangan bulat acak

lainnya dalam modulus p, yang selanjutnya diacu sebagai s_1 .. $s_{(t-1)}$, s_i rahasia.

- Pilih w buah bilangan bulat berbeda x untuk n *participant*, juga dalam modulus p. Dalam implementasi kali ini, dipilih $x_i = i$.
- *Share* yang terbentuk adalah merupakan pasangan (x_i, y_i) yang dalam hal ini $y_i \equiv s_i(x_i)$ dimana berlaku

$$s(x) \equiv M + s_1 x + s_2 x^2 + \ldots + s_{(t-1)} x^{(t-1)} \pmod p$$



Gambar II-1. *Screenshot* Antarmuka Pembangkitan *Share*

B. Algoritma Rekonstruksi Secret

Selanjutnya, tahapan-tahapan untuk melakukan rekonstruksi ulang *secret M* dari *t* buah *share*, diketahui *p* dapat dijabarkan sebagai berikut ini :

- Bentuk *t* buah persamaan :

$$y_k \equiv M + s_1 {x_k}^1 + \ldots + s_{(t-1)} {x_k}^{(t-1)} \pmod p$$

dengan $1 \leq k \leq t$, untuk kemudian dikalkulasi nilai dari M

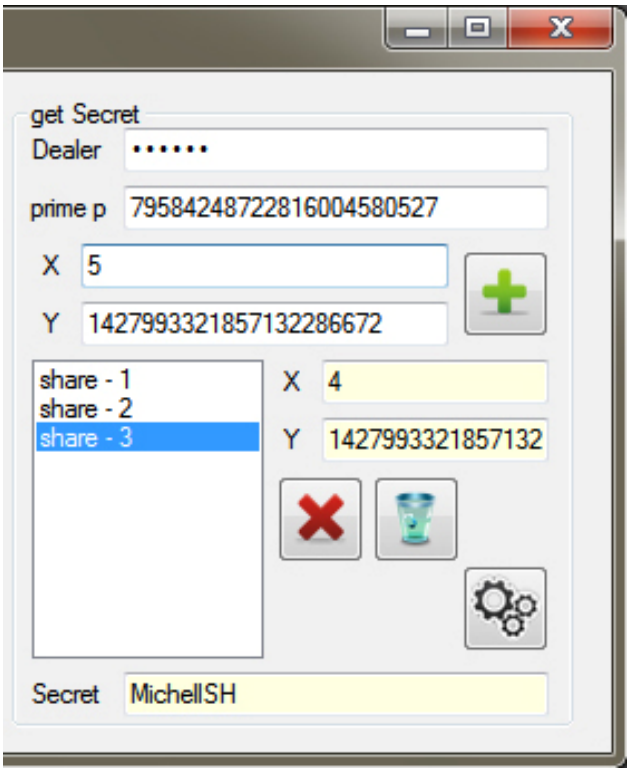
Jika dimisalkan $s_0 = M$, maka didapat suatu sistem persamaan dalam bentuk matriks :

$$\begin{pmatrix} 1 & x_1 & \cdots & {x_1}^{(t-1)} \\ 1 & x_2 & \cdots & {x_2}^{(t-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_t & \cdots & {x_t}^{(t-1)} \end{pmatrix} \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{(t-1)} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_t \end{pmatrix}$$

yang merupakan suatu sistem persamaan linier.

Dengan menggunakan metode eliminasi Gauss-Jordan, dapat dicari suatu solusi untuk $s_0 = M$.

Alternatif lainnya untuk menyelesaikan persoalan interpolasi di atas adalah dengan menggunakan metode interpolasi Langrange yang akan dibahas kemudian pada Bab III.



Gambar II-2. *Screenshot* Antarmuka Rekonstruksi *Secret*

III. PENGUJIAN DAN ANALISIS : PEMBAGIAN DATA RAHASIA – SKEMA AMBANG SHAMIR

A. Batasan-batasan yang Ditemui

Terdapat beberapa batasan yang dijumpai pada saat melakukan implementasi skema pembagian data rahasia menggunakan ambang Shamir pada linkungan implementasi ber-*platform* Microsoft® Windows™ 7™ dengan menggunakan kakas Microsoft® Visual Studio® 2010 dengan bahasa pemrograman C#.NET©. Batasan ini antara lain sebagian besar disebabkan oleh representasi BigInteger yang digunakan. Untuk pengimplementasian, digunakan *library* BigInteger yang disediakan oleh Org.BouncyCastle.Math[1].

- Batasan pesan *secret*

```
BigInteger secret =
   new BigInteger(
      new System.Text.UTF8Encoding().GetBytes(M)
   );
```

[1] http://www.bouncycastle.org/csharp/download/bccrypto-net-1.7-bin.zip 08/05/2011 : 18:00

Pada implementasi yang dilakukan, *secret* dapat berupa *string* apa saja. Representasi *integer* dari *secret* ini didapatkan dengan membuat suatu representasi BigInteger dari hasil konversi *string* ke dalam bentuk *array of byte* menggunakan *encoding UTF8*. Hal ini menyebabkan ukuran *bitlength* dari representasi *integer* menjadi sangat besar karena setiap karakter pada pesan akan dikonversi menjadi satu buah byte dari pembangun BigInteger.

Hal ini kemudian akan berdampak kepada pembangkitan bilangan acak prima p yang harus memenuhi kondisi bernilai lebih besar dari semua kemungkinan nilai pesan *secret M* dan juga melebihi jumlah pesan *share w* yang dibagikan. Dengan kata lain semakin panjang pesan *secret M*, akan semakin besar pula nilai p.

$$p \approx |M|$$

Tentulah representasi bilangan BigInteger itu sendiri memiliki batasan nilai maksimumnya sekalipun dikatakan bahwa BigNumber adalah suatu representasi bilangan yang dapat mencapai ratusan digit.

Adalah merupakan suatu hal yang disayangkan bahwa *library* yang dipakai tidak memfasilitasi informasi berapa nilai maksimum yang dapat ditampung oleh representasi BigInteger yang digunakan sehingga harus dilakukan pengujian untuk menemukan panjang maksimum dari pesan *secret M* yang mana tidak menyebabkan nilai bilangan prima acak p minimum yang dapat dibangkitkan tidaklah melebihi kapasitas BigInteger yang digunakan.

Berdasarkan pengamatan dan uji coba yang dilakukan, *drawback* untuk menyediakan layanan pesan dapat berupa *string* apa saja ternyata sangat besar, yakni : panjang pesan maksimum yang ideal sehingga bilangan acak prima p yang dibangkitkan stabil adalah hanya 10 karakter saja.

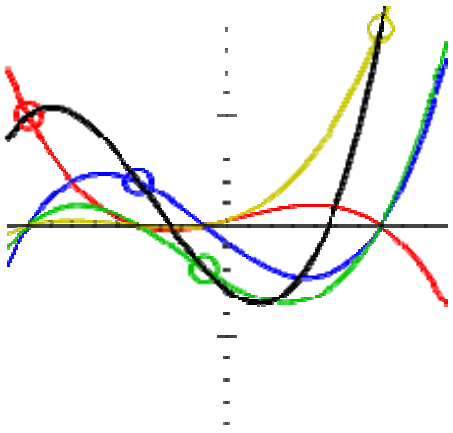
B. Implementasi Interpolasi Polinomial

Pada implementasi yang dilakukan, penyelesaian persoalan interpolasi polinomial untuk rekonstruksi ulang *secret* tidaklah dilakukan dengan pembangunan matriks Vandermonde untuk kemudian diselesaikan dengan eliminasi Gauss seperti yang pada umumnya dilakukan, melainkan diimplementasikan dengan menggunakan metode interpolasi polinomial Lagrange. Formula Lagrange dipilih karena koefisien dari polinomial bukanlah fokus utama dan tidak diperlukan untuk diketahui nilainya, mengingat solusi yang diinginkan hanyalah hasil kalkulasi komputasi $p(x)$ untuk $x = 0$ yang mana tidak ada dalam himpunan data yang diketahui. Pada kasus penggunaan metoda polinomial Lagrange, kompleksitas perhitungan berkurang menjadi $O(n^2)$ dari $O(n^3)$ jika dibandingkan dengan penyelesaian menggunakan solusi Vandermonde.

$$p(x) \equiv y_1 L_1(x_1) + y_2 L_2(x_2) + \ldots + y_t L_t(x_t) \ (mod\ p)$$

yang dalam pada ini,

$$L_k(x) = \prod_{i=1;i\neq k}^{t} \frac{x - x_i}{x_k - x_i} \quad | \quad k = 1, 2, \ldots, t$$



Gambar III-1. Representasi Umum dari Polinomial Lagrange

```
BigInteger px = BigInteger.Zero;
for (int i = 0; i < Share.Count; ++i) {
 BigInteger pemb = BigInteger.One;
 BigInteger peny = BigInteger.One;
 for (int j = 0; j < Share.Count; ++j) {
  if (j != i) {
   pemb = pemb.Multiply(
            x.Subtract(Share.ElementAt(j).Key)
          );
   peny = peny.Multiply(
            Share.ElementAt(i).Key.Subtract(
               Share.ElementAt(j).Key
             )
          );
  }
 }
 px =
   px.Add(
      Share.ElementAt(i).Value.Multiply(pemb).
      Divide(peny)
   );
}
```

Seperti yang dapat dilihat, persoalan yang terjadi dengan digunakannya metode interpolasi Lagrange adalah adanya operasi pembagian.

> *operasi pembagian pada bilangan bulat bersifat terbuka karena pembagian sebuah bilangan bulat dengan bilangan bulat lainnya dapat saja menghasilkan bilangan di luar himpunan bilangan bulat*

Dengan sifat pembagian yang terbuka, maka bila hasil bagi dua buah BigInteger tersebut disimpan di dalam

sebuah BigInteger lainnya, maka tingkat ketelitian operasi dapat dipastikan menurun. Terdapat dua buah alternatif dalam melakukan komputasi kalkulasi dari Π, yakni melakukan seluruh perkalian terlebih dahulu untuk kemudian melakukan pembagian (1) atau melakukan pembagian terlebih dahulu sebelum melakukan perkalian (2). Kedua alternatif ini memiliki *drawback*-nya masing-masing.

Pada alternatif (1) *drawback* yang ada adalah hasil perkalian dapat saja sedemikian besarnya sehingga melampaui kapasitas yang dapat ditampung oleh BigInteger, akan tetapi tingkat akurasi yang didapatkan lebih tinggi bila dibandingkan dengan alternatif (2). Sementara alternatif (2) menawarkan operasi yang lebih aman dalam artian kemungkinan nilai yang dihasilkan melebihi ambang batas nilai maksimum yang dapat ditampung oleh BigInteger menjadi lebih rendah, akan tetapi dengan *trade-off* rendahnya akurasi yang ditawarkan.

Pada implementasi kali ini, dilakukan perhitungan polinomial Lagrange dengan alternatif (1) mengingat operasi perkalian dilakukan terhadap selisih dari dua buah BigInteger yang mana setiap sukunya maksimum bernilai sama dengan jumlah *participant n* sehingga diasumsikan kemungkinan bagi hasil perkalian untuk melampaui batas sangatlah kecil. Kendatipun demikian tingkat akurasi tetap tidak dapat dijaga 100% karena setiap suku yang dipisahkan oleh operator penjumlahan dikalkulasi terlebih dahulu.

Berdasarkan hasil pengujian, pendekatan yang diambil ini cukup baik dalam artian nyaris seluruh *secret* dapat direkonstruksi kembali dengan adanya *t* buah *share*. Kendatipun demikian, terdapat beberapa kasus dimana *secret* tidak kembali seutuhnya, dalam artian terdapat cacat, dan hal ini ditentukan dengan pilihan kombinasi *share* yang dipilih.

Sebagai contoh, untuk kasus dimana berlaku :

secret	MichellSH
n	5
t	3
p	151607336436302497054723

participant	X	Y
1	1	1427993321857132286592
2	2	1427993321857132286192
3	3	1427993321857132285592
4	4	1427993321857132284792
5	5	1427993321857132283792

proses rekonstruksi tidak sepenuhnya terjadi dengan kombinasi acak dari tiga buah *share*, yang mana :

Kasus – 1

share - 1	*participant - 2*
share - 2	*participant - 3*
share - 3	*participant - 1*
secret	MichellSH

Kasus – 2

share - 1	*participant - 2*
share - 2	*participant - 4*
share - 3	*participant - 5*
secret	MichellSG

Kasus – 3

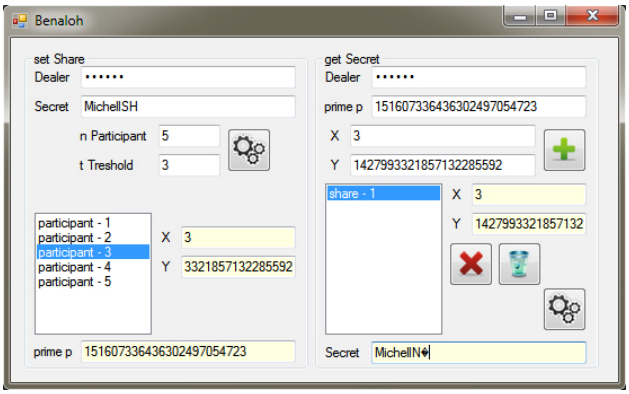
share - 1	*participant - 1*
share - 2	*participant - 2*
share - 3	*participant - 4*
share - 4	*participant - 3*
secret	MichellSH

Kasus – 4

share - 1	*participant - 1*
share - 2	*participant - 2*
share - 3	*participant - 4*
share - 4	*participant - 5*
secret	MichellSI

C. Beberapa Drawback *Lainnya*

Selain beberapa *drawback* yang sudah disebutkan sebelumnya, pada implementasi kali ini pembangkitan bilangan acak dilakukan dengan menggunakan umpan yang berasal dari *password dealer* dengan tujuan lebih terkontrolnya lingkungan pengujian dan lebih rendahnya kemungkinan bilangan acak yang dibangkitkan sama satu dengan lainnya. Akan tetapi hal ini berdampak kepada bilangan acak yang dibangkitkan tidak terlalu jauh berbeda nilainya (range pembangkitan bilangan acak terlalu kecil untuk representasi BigInteger), sehingga masing-masing *share* memegang peranan yang sangat besar terhadap *secret* yang dengan kata lain dapat disebutkan bahwa penebakan *secret* hanya dari sebuah *share* saja sudah sangat mudah untuk dilakukan.



Gambar III-2. Hasil Pengujian dengan Jumlah Share < t

IV. KESIMPULAN : PEMBAGIAN DATA RAHASIA – SKEMA AMBANG SHAMIR

A. Solusi Umum

- panjang pesan *secret*

  Terdapat beberapa alternatif solusi yang dapat diambil untuk mengatasi panjang *secret* yang hanya 10 karakter, antara lain adalah sebagai berikut ini, yakni :

  - Membatasi kemungkinan karakter yang

dapat dikandung oleh pesan *secret*

Dengan dibatasinya himpunan karakter yang mungkin dikandung oleh pesan *secret*, satu karakter tidak lagi memakan ukuran satu byte pada representasi BigIntegernya sehingga dengan batasan BigInteger yang tetap, pesan *secret* dapat lebih panjang.

- Merubah tata cara konversi pesan *secret* menjadi bentuk integer yang berpadanannya

- keteracakan nilai *share* Y

Untuk meningkatkan keamanan dari *secret*, sebaiknya algoritma yang digunakan untuk membangkitkan bilangan acak semu tidaklah menggunakan umpan. Adalah lebih baik jika digunakan pengacakan yang aman untuk kriptografi seperti dengan pemanfaatan teori bilangan *chaotic*.

B. Implementasi Interpolasi Polinomial

Masih terdapat beberapa alternatif untuk pencarian ruang solusi dalam persoalan interpolasi polinomial, yang mungkin digunakan sebagai ganti metode persamaan interpolasi polinomial Lagrange. Sekalipun mungkin kompleksitas meningkat, akan tetapi integritas data pesan *secret* merupakan fokus kejaran utama pada skema pembagian rahasia sehingga tingkat akurasi yang salah seharusnya tidak dapat ditoleransi. Selain daripada itu, dapat dicoba penggunaan metode interpolasi *spline*.

V. STUDI ANALISIS : PEMBAGIAN DATA RAHASIA DENGAN VERIFIKASI – SKEMA BENALOH

Ide dasar dari skema Benaloh adalah memfasilitasi pemegang *share*, yang selanjutnya disebut sebagai *holder*, untuk melakukan verifikasi integritas *dealer* dalam membangkitkan *share* sejumlah *w* untuk *n participant* dari *secret* yang mana memiliki ambang *t*, dengan kata lain setiap bagian dari *share* haruslah *collectively t-consistent* (sebarang himpunan bagian *t* dari *w* akan menghasilkan polinomial yang sama dan benar tanpa menyingkapkan *secret*).

Pada skema ambang Shamir, *share* s_1, ..., s_n akan memenuhi *t-consistent* jika dan hanya jika interpolasi dari titik $(1, s_1)$, ... (n, s_n) membentuk suatu polinom berderajat maksimum $d = (t - 1)$.

> *diberikan derajat dari hasil penjumlahan dua buah polinomial F dan G tidak lebih besar dari t, maka berlaku baik derajat F dan G keduanya tidak lebih besar dari t atau keduanya berderajat lebih besar dari t*
>
> *- properti homomorphic fungsi polinomial*

Tujuan ini dapat dicapai dengan beberapa tahapan pembuktian interaktif yang dapat dijabarkan seperti pada berikut ini, yakni :

1. distribusikan *share* dari polinomial *P* seperti pada skema ambang Shamir
   dengan demikian setiap *holder* akan memiliki pasangan *share* *x*, *y* yang mana akan membawanya kepada *secret*.
2. konstruksikan sejumlah besar *k* polinomial acak P_1, ..., P_k berderajat *t* untuk lalu terkemudian didistribusikan *share*-nya
   pada langkah ini, setiap *holder* ke-*i* akan memiliki $x_{i_1} .. x_{i_k}$ beserta $y_{i_1} .. y_{i_k}$ pasangannya
3. *holder* memilih sebarang himpunan bagian *m* dari polinomial acak dimana berlaku $m < k$ tanpa perlu mengetahui persamaan polinomial yang dipilihnya tersebut
4. selanjutnya *dealer* memberitahukan polinom $P_{i_1} .. P_{i_m}$ dan jumlah dari $k - m$ polinomial sisanya dengan polinom P : $P + \sum_{j=m+1}^{k} P_j$ dan memberitahukan hasil penjumlahan tersebut
   holder dapat mengecek keabsahan dari $P_{i_1} .. P_{i_m}$ dengan cara mengkalkulasi $x_{i_1} .. x_{i_k}$ dengan harapan mendapatkan $y_{i_1} .. y_{i_k}$; sementara hasil penjumlahan akan digunakan untuk menentukan derajat daripada *P*, apakah benar tidak melebihi nilai *t*
5. dengan demikian setiap *holder* dapat memastikan bahwa seluruh polinom yang disingkapkan berderajat *t* dan berkorespondensi dengan *share* yang dimilikinya sementara dalam pengujian ini, *dealer* dapat lulus verifikasi integritas otorisasi tanpa harus membuka *secret* yang dipegangnya

Adapun pembuktian dari kelima tahapan dalam protokol Benaloh ini dapat dijelaskan sebagai berikut ini :

- Diagnosa – 1 :
  Karena derajat dari $P + \sum_{j=m+1}^{k} P_j$ tidak melampaui *t* dan karena *dealer* membeberkan polinom lainnya $P_{i_1} .. P_{i_m}$, maka derajat dari polinom *P* haruslah tidak lebih besar dari *t*.
- Diagnosa – 2 :
  Paramerter kunci dari tujuan skema pembagian rahasia dengan verifikasi adalah penghindaran pembeberan *secret*. Properti ini tetap terjaga dengan penggunaan aljabar homomorphisme untuk melakukan validasi (suatu himpunan dari polinom acak berderajat maksimum *t* bersama dengan sekumpulan hasil operasi penjumlahan dari *P* dengan polinom lain dengan derajat maksimum *t* tidak akan memberikan informasi tambahan apapub mengenai *P*).

VI. LAMPIRAN

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Org.BouncyCastle.Math;

namespace Benaloh {
    class Shamir {
        public static BigInteger[] getShare (int seed, String M, int n, int t, out BigInteger
         p) {
            BigInteger[] RetVal_ysx = new BigInteger[n];

            int next = new Random(seed).Next();

            BigInteger secret = new BigInteger(new System.Text.UTF8Encoding().GetBytes(M));
            p = new BigInteger(((secret.BitLength > 32 ? (secret.BitLength + 8) : (32 + 8)) +
             1), new Random(next = new Random(seed).Next())).NextProbablePrime();
            BigInteger[] s = new BigInteger[(t - 1)];
            for (int i = 0; i < s.Length; ++i) {
                s[i] = new BigInteger(p.BitLength, new Random(next = new Random(seed).Next
                ())).Mod(p);
            }

            for (int i = 0; i < RetVal_ysx.Length; ++i) {
                RetVal_ysx[i] = secret;
                for (int j = 0; j < s.Length; ++j) {
                    RetVal_ysx[i] = RetVal_ysx[i].Add(s[j].Multiply(new BigInteger((i + 1).
                    ToString()).Pow(j + 1)));
                }
                RetVal_ysx[i] = RetVal_ysx[i].Mod(p);
            }

            return (RetVal_ysx);
        }

        public static String setSecret (int seed, BigInteger p, Dictionary < BigInteger,
        BigInteger > Share) { //using Lagrange Interpolation
            BigInteger x = BigInteger.Zero;

            BigInteger px = BigInteger.Zero;

            for (int i = 0; i < Share.Count; ++i) {
                BigInteger pemb = BigInteger.One;
                BigInteger peny = BigInteger.One;
                for (int j = 0; j < Share.Count; ++j) {
                    if (j != i) {
                        pemb = pemb.Multiply(x.Subtract(Share.ElementAt(j).Key));
                        peny = peny.Multiply(Share.ElementAt(i).Key.Subtract(Share.ElementAt(
                        j).Key));
                    }
                }
                px = px.Add(Share.ElementAt(i).Value.Multiply(pemb).Divide(peny));
            }

            return (new System.Text.UTF8Encoding().GetString(px.Mod(p).ToByteArray()));
        }
    }
}
```

REFERENCES

http://en.wikipedia.org/wiki/Secret_sharing; Minggu, 09 Mei 2011 : 18^{00}.

http://en.wikipedia.org/wiki/Verifiable_secret_sharing; Minggu, 09 Mei 2011 : 18^{00}.

http://en.wikipedia.org/wiki/Benaloh_cryptosystem; Minggu, 09 Mei 2011 : 18^{00}.

http://en.wikipedia.org/wiki/Interpolation; Minggu, 09 Mei 2011 : 18^{00}.

http://en.wikipedia.org/wiki/Polynomial_interpolation; Minggu, 09 Mei 2011 : 18^{00}.

http://en.wikipedia.org/wiki/Lagrange_polynomial; Minggu, 09 Mei 2011 : 18^{00}.

http://en.wikipedia.org/wiki/System_of_linear_equations; Minggu, 09 Mei 2011 : 18^{00}.

http://research.microsoft.com/en-us/um/people/benaloh; Minggu, 09 Mei 2011 : 18^{00}.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 09 Mei 2011

Michell Setyawati Handaka / 135 08 045