Penggunaan ECC pada Timestamping

Gabrielle Wicesawati Poerwawinata-13510060[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13510060@std.stei.itb.ac.id

***Abstraksi*—Timestamping adalah proses dari kemananan komputer yang dapat menelusuri waktu pembuatan dan modifikasi dari sebuah dokumen. Teknik pembuatan sebuah timestamp berdasarkan pada *digital signature* dan fungsi hash. Algoritma yang biasanya digunakan pada pembuatan digital signature adalah DSA dan RSA. DSA dibangun pada bulan Agustus 1991 oleh U.S. National Institute of Standard and Technology (NIST) dan distandardisasikan oleh FIPS sehingga dinamakan Digital Signature Standard.**

Eliptical Curve Cryptography saat ini banyak sekali digunakan pada digital signatures dan key *agreement.* ECC menggunakan ukuran kunci yang lebih kecil dan implementasi yang lebih efisien. Oleh karena itu, Eliptic Curve Digital Signature Algorithm(ECDSA), dapat membuat komputasi menjadi lebih cepat dan mengurangi tenaga atau sumber daya yang digunakan pada saat proses berjalan.

Oleh karena itu, pada paper ini akan membahas implementasi algoritma Eliptical Curve Cryptography dalam pembuatan digital signature.

Index Terms—digital signatures*, DSA, ECC, *timestamp*, *trusted timestamp

I. PENDAHULUAN

DSA dibangun pada bulan Agustus 1991 oleh U.S. National Institute of Standards and Technology(NIST) dan distandardisasikan oleh FIPS. Keamanan dari DSA berdasarkan pada masalah logaritma diskrit dalam *prime-order subgroups*. Digital signatures dibangun untuk menggantikan tanda tangan manusia dan menyediakan kontrol penggunaan dan mode akses.

Akan tetapi DSA tidak effisien digunakan pada alat komputasi yang kecil dikarenakan komputasi untuk DSA ini membutuhkan waktu yang cukup lama. Oleh karena itu dapat diusunglah gagasan untuk menggunakan ESCDSA untuk menggantikan DSA dalam pembuatan digital signatures.

II. TIMESTAMP

Timestamping adalah proses dari kemananan komputer yang dapat menelusuri waktu pembuatan dan modifikasi dari sebuah dokumen. Keamanan yang dimaksudkan disini berarti tidak ada seorangpun, bahkan pembuat dokumen juga tidak berhak untuk mengubah dokumen yang sudah ditempeli dengan timestamper.

Terdapat banyak skema timestamping dengan tujuan keamanan yang berbeda-beda, antara lain:

1. PKI-based
2. Linking-based schemes
3. Distributed schemes
4. Transient key scheme
5. MAC
6. Database
7. Hybrid schemes

Timestamp dibangun berdasarkan digital signatures dan fungsi hash. Pertama-tama fungsi hash dikalkulasikan dari data. Data yang dimaksudkan disini dapat berupa text, gambar, dan media yang lainnya. Sehingga data satu dengan data yang lainnya akan mempunyai hasil dari fungsi hash yang berbeda-beda. Hasil dari fungsi hash ini akan dikirimkan ke Time Stamping Authority(TSA). Lalu TSA akan menggabungkan sebuah timestamp dengan fungsi hash yang sebelumnya dikirimkan kepada TSA. Hasil dari penggabungan ini akan dihashkan juga. Hasil dari hash ini akan mendapatkan digital signatures dengan kunci privat dari TSA. Hasil dari signed hash dan timestamp akan dikirimkan kembali pada objek yang mengirimkan data aslinya pada TSA.

Karena data yang asli tidak bisa didapatkan kembali dari fungsi hash yang sudah dihasilkan(fungsi hash bersifat satu arah), maka TSA tidak akan pernah dapat melihat data yang asli sehingga hal ini dapat menjamin *confidentiality* dari data.

Berikut ini merupakan contoh dari timestamp:

- 2005-10-30 T 10:45 UTC
- 2007-11-09 T 11:20 UTC
- Sat Jul 23 02:16:57 2005
- 12569537329

Orang yang akan menggunakan data ini akan menggunakan timestamp ini untuk memastikan bahwa data tidak dibuat atau dimodifikasi setelah waktu yang telah dicatat oleh timestamper. Pengecekan dilakukan dengan public key yang dimiliki oleh TSA.

III. MODIFIKASI DSA DENGAN ECDSA

A. Digital Signature

Digital Signature adalah skema matematika untuk menyediakan fitur keamanan autentikasi dari sebuah pesan digital atau dokumen. Digital signatures digunakan untuk menyediakan beberapa layanan kriptografi seperti:

- data integrity yang dimaksudkan untuk memastikan data tidak dimodifikasi oleh pelaku yang tidak mempunyai otorisasi pada data tersebut
- membuktikan keaslian data
- non-repudiation (sebagai barang bukti dari aksi sebuah entity)

Skema kriptografi dari digital signatures dimulai dari faktorisasi integer yaitu menentukan dua buah bilangan prima yang besar yaitu p dan q. Lalu dilakukan perhitungan n =p*q dan z = (p-1)*(q-1). Lalu pilih sebuah bilangan e yang besarnya kurang dari n dan tidak mempunyai faktor dengan z. Dan yang terakhir adalah pilih bilangan d, dengan e*d-1 yang dapat dibagi dengan z. Public key adalah pasangan bilangan (n,e) sedangkan private key adalah pasangan bilangan(n,d). Enkripsi dilakukan dengan rumus $c = m^c \bmod n$. Untuk mengenkripsi pesan c dapat dilakukan dengan $m = c^d \bmod n$. Faktorisasi dari bilangan prima ini mempunyai masalah karena akan memakan waktu sebanyak $\exp[1.923*(\log n)^{1/3}*(\log \log n)^{2/3}]$.

B. DSA

DSA menggunakan logaritma diskrit. Logaritma yang biasa $\log_a(b)$ adalah solusi dari persamaan $a^x=b$ pada bilangan real dan kompleks. Jika g dan h adalah elemen dari finite cyclic group G lalu solusi dari x dari persamaan $g^x=h$ dinamakan logaritma diskrit pada grup G. Operasi-operasi pada logaritma diskrit memenuhi beberapa atribut berikut ini:

1. Closure: a*b ε G untuk semua a,b ε G
2. Associativity: a * (b * c) = (a * b) * c untuk semua a,b ε G.
3. Eksistensi dari Identitas : Terdapat elemen identitas e ε G, dengan e * a = a * e = a untuk semua a ε G.
4. Eksistensi dari inverse : Untuk setiap a ε G terdapat elemen b ε G, dengan a * b = b * a = e. Elemen b adalah inverse dari a.

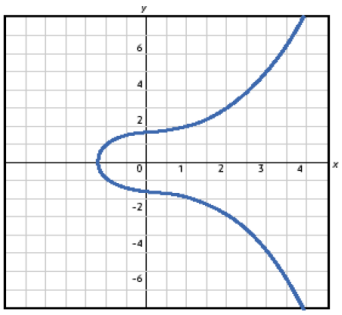
Grup G dikatakan abelian jika a*b = b* a untuk semua a,b ε G. Group order pada G merupakan banyaknya elemen dari G.

Permasalahan dari logaritma diskrit ini adalah untuk mencari integer x, $0 \le x \le n-1$, dengan $g^x \equiv h \pmod p$, untuk setiap g ε Z*p pada n dan h ε Z*p. Maka integer x dinamakan logaritma diskrit pada h dengan base g.

C. Eliptical Curve Discrete Logarithm

Sebuah kurva elips E_k, didefinisikan pada K dengan karakteristik ≠ 2 atau 3 adalah set solusi (x,y) ε K' pada persamaan

$y^2 = x^3 + ax + b$, yang ditunjukkan pada Gambar 1.



Gambar 1Kurva Elips

Dua bilangan positif a dan b, yang kurang dari p dan memenuhi:

$4a^3 + 27b^2 \pmod p \neq 0$

Lalu $E_p(a,b)$ menandakan grup eliptic mod p dengan elemen (x,y) berupa bilangan positif dengan besar kurang dari p yang memenuhi:

$y^2 \equiv x^3 + ax + b \pmod p$

bersamaan dengan titik infinity pada Q.

Logaritma kurva diskrit dapat dinyatakan dengan bilangan prima p dan sebuah kurva elips

Q = xP

dengan xP merepresentasikan titik P pada kurva elips sebanyak x kali. Masalah selanjutnya adalah menentukan banyaknya x pada P dan Q. Algoritma terbaik untuk menyelesaikan ECDLP adalah algoritma Pollard-Rho dengan waktu yang dihabiskan adalah $\sqrt{(\Pi*n/2)}$

D. Elliptic Curve Cyptography

ECC ditemukan oleh Neal Koblitz dan Victor Miller. ECC menggantikan Z_p* pada logaritma diskrit dengan sekelompok titik pada kurva elips. Sebuah kurva elips E pada Z_p didefinisikan dengan persamaan

$y^2 = x^3 + ax + b$

dengan a,b ε Z_p dan $4a^3 + 27b^2 \neq 0 \pmod p$, dengan titik O yang dinamakan point pada titik tak hingga. Set $E(Z_p)$ terdiri dari titik (x,y), x, y ε Z_p, yang memenuhi persamaan diatas.

Setiap nilai dari a dan b memberikan kurva elips yang berbeda. Public key adalah titik dari kurva dan private key adalah bilangan random. Kunci privat didapatkan dari memilih secara random bilangan integer bukan nol pada grup order n. Public key didapatkan dengan mengalikan kunci privat dengan titik G pada kurva. Public key didapatkan dengan rumus Q=dG, dengan d adalah jumlah titik G pada kurva. Lalu public key dipilih secara random dari perhitungan Q pada titik G tersebut.

E. Elliptic Curve Digital Signatures

Penanda pada digital signature akan membuat pasangan kunci (d,Q) dengan d merupakan private key dan Q merupakan public key. Untuk menandai pesan m, penanda harus memilih sebuah bilangan k per pesan dengan syarat $1 \le k \le n-1$. Lalu dari titik (x_1,y_1) dihitung $(x_1,y_1)=kG$. Lalu dihitung $r = x_1 \bmod n$. Pesan m akan dihash dengan bitstring dengan panjangnya yang kurang dari panjang bit n. Sehingga signature dari pesan m adalah pasangan dari (r,s) dengan $s = k^{-1}(e+dr) \bmod n$. Dengan catatan r dan s

harus berbeda dari 0, dan k tidak boleh dipublikasi sehingga setiap message akan mempunyai k yang berbeda-beda. Masing-masing k pada pesan tidak boleh diketahui oleh pesan-pesan yang lain. Hal ini dikarenakan kunci privat d dapat diturunkan dari $d \equiv r^{-1}(ks-e)(\text{mod } n)$, r dan s tidak dirahasiakan pada digital signatures dan e dapat dihitung dari pesan yang sudah ditandai dengan digital signatures.

IV. IMPLEMENTASI ECC PADA TIMESTAMP

Implementasi algoritma ECC dan Timestamp akan dibuat dalam bahasa Java. Pertama-tama akan dibahas bagaimana mendapatkan timestamp dan kemudian timestamp tersebut digabungkan dengan pesan yang akan diberikan timestamp. Lalu setelah mendapatkan hasil hash dari pesan dan timestamp, akan dibentuk digital signatures dengan menggunakan algoritma ECC.

A.Implementasi Timestamp

Berikut ini adalah kode untuk mendapatkan timestamp pada waktu program dijalankan pada bahasa Java.

```
java.util.Date date= new java.util.Date();
System.out.println(new Timestamp(date.getTime()));
```

Keluaran dari program di atas adalah

```
2014-05-15 08:52:59.589
BUILD SUCCESSFUL (total time: 1 second)
```

B.Implementasi Hash

Fungsi hash yang digunakan adalah SHA-1. Berikut adalah pesan yang akan dijadikan contoh untuk pembuatan digital signatures dengan timestamp.

```
As you probably know, it is generally agreed that there are five core character traits from which all human personalities stem called… get this…The Big Five. They are: Openness, Conscientiousness, Extroversion, Agreeableness, and Neurotic. Each exists on a continuum with its opposite on the other end, and our personality is the expression of the dynamic interaction of each and all at any given time. One minute you may feel more agreeable, the next more neurotic, but fortunately, day-to-day, they collectively remain fairly stable for most of us.
```

Hasil hash dari text diatas didapatkan

```
e4eaf86dc15cce941703faab3b724857060e81e2
```

Kemudian hasil hash tersebut digabungkan dengan timestamp yang terbentuk pada pukul 08:52 tanggal 15 Mei 2014. Hasil penggabungan ini dihashkan kembali, sehingga didapatkan:

```
87dbd849b4c24cbe904f55e63c6d9f4862ca529d
```

Hasil hash ini selanjutnya akan dibangun menjadi sebuah digital signature dengan algoritma ECC .

C.Implementasi ECDSA

Fungsi enkripsi pada ECDSA adalah:

```
public byte[] encrypt(byte[] input,int numbytes, Key key) {
        ECKey ek = (ECKey) key;
        byte[] res=new byte[ek.mother.getPCS()+numbytes];
        hash.reset();

        BigInteger rk = new BigInteger(ek.mother.getp().bitLength() + 17, Rand.om);
        if (ek.mother.getOrder() != null) {
          rk = rk.mod(ek.mother.getOrder());
        }
        ECPoint gamma = ek.mother.getGenerator().multiply(rk);
        ECPoint sec = ek.beta.multiply(rk);
        System.arraycopy(gamma.compress(),0,res,0,ek.mother.getPCS());
        hash.update(sec.getx().toByteArray());
        hash.update(sec.gety().toByteArray());
        byte[] digest = hash.digest();
        for(int j = 0; j < numbytes; j++) {
          res[j+ek.mother.getPCS()]=(byte) (input[j]^digest[j]);
        }
        return res;
    }
```

Fungsi dekripsi pada ECDSA adalah:

```
public byte[] decrypt(byte[] input, Key key) {
        ECKey dk = (ECKey) key;
        byte[] res=new byte[input.length-dk.mother.getPCS()];
        byte[] gammacom=new byte[dk.mother.getPCS()];
        hash.reset();

        System.arraycopy(input,0,gammacom,0,dk.mother.getPCS());
        ECPoint gamma = new ECPoint(gammacom,dk.mother);
        ECPoint sec = gamma.multiply(dk.sk);
        if(sec.isZero()) {
          hash.update(BigInteger.ZERO.toByteArray());
          hash.update(BigInteger.ZERO.toByteArray());
        } else {
          hash.update(sec.getx().toByteArray());
          hash.update(sec.gety().toByteArray());
        }
        byte[] digest = hash.digest();
        for(int j = 0; j < input.length-dk.mother.getPCS(); j++) {
          res[j]=(byte) (input[j+dk.mother.getPCS()]^digest[j]);
        }
        return res;
    }
```

Public key pada percobaan dengan algoritma ECDSA adalah:

```
public x coord:
257079148622546608990569460072452455986560399832018850355345948277254989083 58
public y coord:
326197645169318959979019713080794155269565525513362317358740234546841222284 12
parameters :
secp256r1
```

Private key yang digunakan adalah

```
private value:
386250215949653028613423594360745195417436637923769560194794649174276030361 43
public x coord:
257079148622546608990569460072452455986560399832018850355345948277254989083 58
public y coord:
326197645169318959979019713080794155269565525513362317358740234546841222284 12
parameters:
secp256r1
```

Hasil enkripsi digital signature yang didapatkan dari ECDSA ditunjukkan sebagai berikut:

```
014B11AB8730ED58A3162E504C81F1FEEB44E87C
152B6659F2375BC4C416925E05A6343882442A4C731F4F
CABFC62278633E338BF1
```

Jika timestamp diimplementasikan menggunakan RSA dengan kunci publik

```
110578714574006666791146138828762716192829612058631616041820692462838846287145934208741997087349955130475129939049578221665240941609429081020555375885969117,8575108116364668761280702790842504132642197220336470567205210680325397735397843867893210973071731051483692731056748759982946666804403933798058307103752167
```

dan private key

```
110578714574006666791146138828762716192829612058631616041820692462838846287145934208741997087349955130475129939049578221665240941609429081020555375885969117,8918090809185597560612213451311560759556693838035594108184447573490684945609709832403430743009851093182202772196337375967662837995061669273914843048222603
```

maka digital signature yang didapatkan adalah

```
5433D004227CA179C99EE356F52E49AD3ED43B5A659E590B86420D11528F3BFDB6CE58ED432FD96EBD2B279F95DCC27E8B7F0BB807A9FA1D11F0ACB4806169D7
```

V. ANALISIS

Dari hasil impelementasi ditunjukkan bahwa panjang kunci yang dibutuhkan untuk algoritma ECDSA lebih pendek daripada panjang kuncin pada RSA. Kemudian dari hasil enkripsi, ECDSA menghasilkan output yang lebih pendek daripada oleh RSA.

VI. KESIMPULAN

Algoritma enkripsi EC membutuhkan bilangan prima, titik pada kurva elips. Selain itu untuk menentukan kunci privat pada algoritma ini ditentukan secara random bilangan k, sehingga sangat sulit dideteksi oleh hacker.

Selain itu ECC lebih cocok jika diimpelementasikan pada alat komputasi yang kecil karena kunci yang digunakan ukurannya lebih kecil daripada algoritma RSA.

REFERENCES

[1] V.Gasoyo Martinez,L.Hernandez Encinas.*Implementing ECC with Java Standard Edition 7*. Spain:Indormation Security Institute(ISI), 2013

[2] Aqeel Khalique,Kuldip Singh, Sandeep Sood. *Implementation of Elliptic Curve Digital Signature Algorithm*. Internation Journal of Computer Application.India:Indian Institute of Technology Roorkee.2010

[3] Joppe W.Bos, J. Alex Halderman, Nadia Heninger, Jonathan Moore, Michael Nachrig, Eric Wustrow. *An Elliptic Curve Cryptography in Practice*. Microsoft Research.

[4] V. Gasoyo Martinez, L.Hernandez Encinas, C. Sanchez Avila. *A Java Impelementation of the Elliptic Curve Integrated Encyrption Scheme*. Spain: Institute of Applied Physics, CSIC.

[5] Johann, Dan Page, Stefan Tilich. *Efficient Java Implementation of Elliptic Curve Cryptography for J2ME-Enabled Mobile Devices*.Luxembourg: University of Luxembourg.

[6] Don Johnson,Alfred Menezes,Scott Vanstone. *The Elliptic Curve Digital Signature Algorithm*.Canada: University of Waterloo.

[7] Ikshwansn Nautiyal, Madhu Sharma. *Encryption using Elliptic Curve Cryptography using Java as Implementation Tool.Internaional Journal of Advanced Research in Computer Science and Software Engineering*.2014

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, 19 Mei 2014

ttd

Gabrielle Wicesawati Poerwawinata / 13510060