Elliptic Curve Digital Signature Algorithm (ECDSA)

Benny Roy P.N, Citrady L.M, dan Roni F. Sinaga

Departemen Teknik Informatika
Institut Teknologi Bandung
Jalan Ganesha 10 Bandung 40132

E-mail : if11014@students.if.itb.ac.id, if11061@students.if.itb.ac.id, if11083@students.if.itb.ac.id

Abstraksi

Elliptic Curve Digital Signature Algorithm (ECDSA) adalah salah satu algoritma yang diterapkan dalam pembuatan tanda tangan digital yang menggunakan analogi kurva elips. Tidak seperti logaritma diskrit biasa dan masalah faktorisasi integer, masalah logaritma diskrit kurva elips tidak mengenal algoritma perkalian sub-eksponensial. Karenanya, kekuatan per bit kunci algoritma yang menggunakan kurva elips lebih kuat secara substansial daripada algoritma biasa. Paper ini secara khusus membahas konsep dasar dari kurva elips, cara menggabungkan kurva elips dengan *Digital Signature Algorithm* (DSA), serangan – serangan yang pernah di-lakukan terhadap ECDSA, kelebihan dan ke-kurangan ECDSA, dan perbandingan ECDSA dengan algoritma *public key* yang lain.

Kata kunci : kurva elips, tanda tangan digital,

1. Pendahuluan

DSA yang dijadikan *Digital Signature Standard* (DSS) memiliki tingkat keamanan berdasarkan pada kemampuan komputasi dari logaritma diskrit dalam subgrup urutan prima Z_p^*. Elliptic curve cryptosystems (ECC) ditemukan oleh Neal Koblitz dan Viktor Miller pada tahun 1985. ECC dapat dilihat sebagai analogi kurva elips dari discrete logarithm (DL) cryptosystem yang lebih tua di mana subgrup Zp* diganti dengan sekelompok titik pada kurva elips pada bidang yang terbatas. Basis keamanan secara matematis dari ECC adalah kemampuan komputasi dari *elliptic curve discrete logarithm problem* (ECDLP).

Karena ECDLP secara signifikan lebih sulit daripada DLP, maka kekuatan per bit kunci pada sistem kurva elips lebih kuat daripada sistem logaritma diskrit konvensional. Karenanya, ECC dapat menggunakan parameter yang lebih kecil daripada sistem DL dengan tingkat keamanan yang sama. Keuntungan yang diperoleh dengan parameter yang lebih kecil adalah kecepatan, kunci dan sertifikat yang lebih kecil. Keuntungan ini menjadi penting pada saat proses power, tempat penyimpanan, bandwith atau konsumsi power terbatas.

ECDSA adalah analogi kurva elips pada DSA. ECDSA pertama kali diperkenalkan oleh Scott Vanstone pada tahun 1992 sebagai respon untuk permintaan NIST (National Institute of Standard and Technology) untuk komentar publik mengenai proposal pertama mereka untuk DSS.

2. Konsep Dasar

Untuk memahami konsep dari ECDSA maka ada 3 hal yang perlu kita pahami terlebih dahulu yaitu bidang berhingga yang kali ini hanya ditinjau pada bidang berhingga F_2^m, konsep dari Kurva Ellips itu sendiri dan tandatangan digital. Berikut akan dijelaskan mengenai kedua konsep tersebut

2.1. Bidang berhingga F_2^m

Bidang berhingga F_{2^m} adalah karakteristik 2 bidang berhingga yang mengandung 2^m elemen. Walaupun hanya ada satu karakteristik 2 bidang berhingga F_{2^m} untuk setiap perpangkatan 2^m dengan 2 dengan m ≥ 1, tapi ada banyak cara untuk merepresentasikan elemen dari F_{2^m}.

Elemen dari F_{2^m} seharusnya direpresentasikan dengan polinomial biner dengan derajat m-1 atau kurang :

$r = r_{m-1}x^{m-1} + ... + r_0$

$r_i \equiv a_i + b_i$

$\{a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + ... + a_1x + a_0 : a_i \in \{0,1\}\}$

dengan penambahan atau perkalian terdefinisi dalam polinomial biner tak tereduksi f(x) dengan derajat m, dikenal dengan polinomial tereduksi, seperti :

- Penambahan : Jika $a = a_{m-1}x^{m-1} + ... + a_0$, $b = b_{m-1}x^{m-1} + ... + b_0 \in F_{2^m}$, maka a + b = r dalam F_{2^m}, di mana $r = r_{m-1}x^{m-1} + ... + r_0$ dengan $r_i \equiv a_i + b_i \pmod 2$.
- Perkalian : Jika $a = a_{m-1}x^{m-1} + ... + a_0$, $b = b_{m-1}x^{m-1} + ... + b_0 \in F_{2^m}$, maka a,b = s dalam F_{2^m} di mana $s = s_{m-1}x^{m-1} + ... + s_0$ adalah pengingat ketika polinomial ab dibagi dengan f(x) dengan segala koefisien aritmatik menunjukan modulo 2.

Penambahan dan perkalian dalam F_{2^m} dapat dihitung secara efisien dengan menggunakan algoritma standar untuk integer biasa dan aritmatika polinomial. Dalam representasi F_{2^m} ini, identitas untuk penambahan atau elemen 0 adalah polinomial 0, dan identitas untuk perkalian adalah polinomial 1.

Mudah untuk mendefinisikan pengurangan dan pembagian untuk elemen bidang. Untuk itu, invers dari penambahan dan invers dari perkalian dari elemen bidang harus diberikan :

- Invers penambahan : jika $a \in F_{2^m}$, maka invers penambahan (-a) dari a pada F_{2^m} adalah solusi unik untuk persamaan a + x = 0 dalam F_{2^m}
- Invers perkalian : jika $a \in F_{2^m}$, $a \neq 0$, maka invers perkalian a^{-1} dari a dalam F_{2^m} adalah solusi unik untuk persamaan a,x=1 dalam F_{2^m}

Invers penambahan dan invers perkalian dalam F_{2^m} dapat dihitung secara efisien dengan menggunakan algoritma Euclidean yang dikembangkan (*extended*). Pembagian dan pengurangan didefinisikan dalam dalam term invers penambahan dan invers perkalian : a-b dalam F_{2^m} adalah a + (-b) dalam F_{2^m} dan a/b dalam F_{2^m} adalah $a.(b^{-1})$ dalam F_{2^m}

Dalam hal ini, karakteristik 2 bidang berhingga F_{2^m} harus memiliki : $m \in \{113,131,163,193,233,239,283,409,571\}$ dan penambahan dan perkalian dalam F_{2^m} harus ditampilkan dengan menggunakan polinomial biner tak tereduksi dengan derajat m dalam tabel 1.

Field	Reduction Polynomial(s)
$\mathbb{F}_{2113}$	$f(x) = x^{113} + x^9 + 1$
$\mathbb{F}_{2131}$	$f(x) = x^{131} + x^8 + x^3 + x^2 + 1$
$\mathbb{F}_{2163}$	$f(x) = x^{163} + x^7 + x^6 + x^3 + 1$
$\mathbb{F}_{2193}$	$f(x) = x^{193} + x^{15} + 1$
$\mathbb{F}_{2233}$	$f(x) = x^{233} + x^{74} + 1$
$\mathbb{F}_{2239}$	$f(x) = x^{239} + x^{36} + 1$ or $x^{239} + x^{158} + 1$
$\mathbb{F}_{2283}$	$f(x) = x^{283} + x^{12} + x^7 + x^5 + 1$
$\mathbb{F}_{2409}$	$f(x) = x^{409} + x^{87} + 1$
$\mathbb{F}_{2571}$	$f(x) = x^{571} + x^{10} + x^5 + x^2 + 1$

Tabel 1 : representasi F2m

Aturan yang digunakan untuk mengambil m yang diterima : dalam setiap interval diantara integer-integer berikut ini :
{112,128,160,192,224,256,384,512,1024}

2.2.**Konsep dasar DSA(Digital Signature Algorithm)**

Tanda tangan digital DSA berbentuk sepasang besar angka yang ditampilkan komputer sebagai string dari digit biner. Tanda tangan digital dihitung dengan menggunakan sejumlah aturan dan sejumlah parameter sehingga identitas pemilik dan integritas data dapat diverifikasi. Pembuat tanda tangan menggunakan kunci privat untuk membuat tanda tangan; sedangkan kunci publik, yang berkorespodensi dengan kunci privat namun tidak sama, digunakan untuk memverifikasi tanda tangan. Setiap user memiliki sepasang kunci publik dan kunci privat. kunci publik diasumsikan diketahui public secara umum, sedangkan kunci privat tidak pernah disebar.

DSA dapat dilihat sebagai variasi dari skema tanda tangan ElGamal. Keamanan DSA berdasarkan pada kemampuan logaritma diskrit dalam urutan bilangan prima Z_p^*.

Domain parameter DSA dibangkitkan untuk setiap entitas dalam domain keamanan tertentu.

1. Pilih bilangan prima sepanjang 160 bit dan 1024 bit dengan kondisi : q | p – 1
2. Pilih pembangkit g yang memiliki kelompok putaran yang unik di mana q berada dalam Z_p^*.)

   Pilih sebuah elemen h $\in$ Z_p^* dan hitung $g = h^{(p-1)/q} \bmod p$.

   (ulangi hingga $g \neq 1$)
3. Parameter domain adalah p, q dan g.

Setiap entitas A dalam domain, dengan domain parameter (p,q,g) melakukan :

1. pilih bilangan acak x dengan ketentuan $1 \le x \le q-1$
2. hitung $y = g^x \bmod p$.
3. kunci publik A adalah y, dedangkan kunci privat A adalah x

Untuk menandatangani pesan m, A melakukan

1. pilih bilangan acak k dengan ketentuan $1 \le k \le q-1$
2. hitung $X = g^k \bmod p$ dan r = X mod q. Jika r = 0, lakukan langkah 1
3. hitung $k^{-1} \bmod q$
4. hitung e = SHA-1(m)
5. hitung $s = k^{-1}\{e + xr\} \bmod q$. Jika s = 0, lakukan langkah 1
6. tanda tangan A untuk pesan m adalah (r,s)

Untuk memverifikasi tanda tangan A (r,s) pada pesan m, B mendapat salinan sah dari domain parameter A (p,q,g) dan kunci publik y dan melakukan :

1. verifikasi bahwa r dan s berada dalam interval [1,q-1]
2. hitung e = SHA-1(m)
3. hitung $w = s^{-1} \bmod q$
4. hitung $u_1 = ew \bmod q$ dan $u_2 = rw \bmod q$
5. hitung $X = g^{u_1} y^{u_2} \bmod p$ dan v = X mod q
6. tanda tangan benar jika dan hanya jika v = r

Analisa keamanan : karena r dan s masing-masing integer lebih kecil dari q, maka tanda tangan DSA berukuran 320 bit. Keamanan DSA tergantung pada dua logaritma diskrit yang berbeda namun saling berhubungan. Yang pertama adalah logaritma diskrit dalam Z_p^* di mana algoritma penyaring bilangan dalam bidang digunakan. Algoritma ini memiliki waktu berjalan yang subeksponensial, lebih tepatnya digambarkan dalam rumus : $o(\exp((c+o(1))(\ln p)^{1/3}(\ln \ln p)^{2/3}))$. Di mana c ≈ 1.923. Jika p adalah bilangan prima sepanjang 1024 bit, maka rumus di atas menunjukan jumlah komputasi yang sangat banyak; jadi DSA dengan bilangan prima sepanjang 1024 bit tidak cocok untuk diserang dengan jenis serangan ini. Logaritma diskrit kedua yang dapat bekerja pada basis g dalam subgrup q dalam Z_p^* : diberikan p, q, g dan y, temukan x sehingga kondisi $y \equiv g^x (\bmod p)$ terpenuhi. Untuk P yang besar (seperti 1024 bit), dengan meggunakan algoritma terbaik yang diketahui, membutuhkan waktu sekitar $\sqrt{\pi q/2}$ langkah. Jika $q \approx 2^{160}$, maka rumus waktu tadi menghasilkan komputasi yang sangat banyak; jadi DSA tahan dengan serangan ini. Namun perlu diperhatikan bahwa keamanan DSA terletak pada ukuran p dan q, jadi memperbesar salah satu tanpa memperbesar yang lainnya tidak akan efektif meningkatkan keamanan. Lebih dari itu,

pengembangan salah satu dari rumus ini dapat melemahkan DSA.

2.3. Konsep Dasar kurva elips

Kurva ellips yang digunakan disini adalah kurva ellips yang berada di bidang datar yang berhingga dan dibagi atas 2 bagian yaitu :

a. Kurva ellips yang berada di F_p
b. Kurva ellips yang berada di F_2^m

a. Kurva ellips yang berada di F_p

Jika dimisalkan F_p adalah bidang berhingga yang prima dimana p adalah bilangan prima yang ganjil dan misalkan a,b ε F_p memenuhi $4a^3 + 27b^2 /\equiv 0$ (mod p) maka suatu kurva ellips $E(F_p)$ melalui F_p yang didefenisikan oleh parameter a,b ε F_p terdiri dari sekumpulan himpunan penyelesaian atau titik P = (x,y) dimana x,y ε F_p yang sesuai dengan persamaan $y^2 \equiv \_ x^3 + a{:}x + +b$ *(*(mod *p)* ditambah dengan titik tambahan *O* yang disebut titik infinity. Persamaan $y^2 \equiv \_ x^3 + a{:}x + +b$ *(*(mod *p)* disebut defining equation dari $E(F_p)$. untuk titik P (x_p,y_p), x_p disebut koordinat x dan y_p disebut koordinat y.

Jumlah titik dari E (F_p) disimbolkan dengan $\#E(F_p)$. menurut teori dari Hasse diketahui bahwa jumlah titik dari E (F_p) memenuhi rentang dibawah ini :

$p + 1 - \_2p\sqrt{p} \le \_ \#E(F\ p \le) \_ p + +1 + +2p\sqrt{p}$.

Ada beberapa tambahan aturan yang ditambahkan pada E antara lain :

1. Penambahan titik infinity terhadap dirinya sendiri.
   $O + O = O$
2. Penambahan titik infinity dengan titik lainnya.
   $(x, y) + O = O + (x, y)$ untuk semua (x,y) $\in (F_p)$
3. Aturan penambahan 2 titik dimana koordinat x nya sama sedangkan titik koordinat y nya bisa beda atau *O*
   $(x, y) + (x, -y) = O$ untuk semua (x,y) $\in (F_p)$
4. Aturan penambahan 2 titik yang berbeda dimana $(x_1,y_1) \in (F_p)$ dan $(x_2,y_2) \in (F_p)$ dimana $(x_1,y_1) + (x_2,y_2) = (x_3,y_3)$.
   Untuk mencari nilai x_3 dan y_3 digunakan rumus sebagai berikut :
   $x_3 = \lambda^2 - x_1 - x_2$ (mod p), $y_3 = \lambda.(x_1 - x_3) - y_1$ (mod p) dimana $\lambda \equiv \frac{y2 - y1}{x2 - x2}$ (mod p)
5. Aturan penambahan titik, bukan titik infinity, dengan titik itu sendiri (dirinya sendiri) yaitu $(x_1,y_1) + (x_1,y_1) = (x_3,y_3)$, dimana :
   $x_3 = \lambda^2 - 2x_1$ (mod p), $y_3 = \lambda.(x_1 - x_3) - y_1$ (mod p) dan $\lambda = \frac{3x1^2 + a}{2y1}$ (mod p)

Himpunan titik – titik pada $E(F_p)$ membentuk sebuah kelompok dengan aturan ini. Lebih lanjut, dalam kelompok ini berlaku hukum komutatif, yaitu $P_1 + P_2 = P_2 + P_1$, untuk semua titik – titik $P_1,P_2 \in E(F_p)$. Perlu dicatat bahwa aturan ini dapat selalu dikomputasikan secara efisien dengan menggunakan operasi aritmatika sederhana.

Skema kriptografi yang berdasarkan ECC bergantung kepada multiplikasi skalar dalam titik – titik kurva elips. Misalnya diketahui k adalah integer, dan P adalah titik,dimana P $\in$ $E(F_p)$, multiplikasi skalar adalah proses menambahkan P dengan dirinya sendiri sebanyak k kali. Hasil dari proses ini adalah dinyatakan dengan k x P atau kP. Proses ini dapat dikombinasikan dengan aturan – aturan tambahan yang telah dijelaskan di atas.

b. kurva elips yang berada di F_2^m

jumlah titik – titik pada $E(F_2^m)$ dinyatakan dengan $\#E(F_2^m)$. Teorema Hasse menyatakan bahwa :

$2^m + 1 - 2\sqrt{2^m} \le \#E(F_2^m) \le 2^m + 1 + 2.$

Seperti pada bagian sebelumnya, pada bagian ini juga dimungkinkan pendefinisian aturan – aturan tambahan untuk menambahkan titik – titik pada E, yaitu :

1. Penambahan titik infinity terhadap dirinya sendiri.
   $O + O = O$
2. Penambahan titik infinity dengan titik lainnya.
   $(x, y) + O = O + (x, y)$ untuk semua (x,y) $\in E(F_p)$
3. Aturan untuk menambahkan 2 titik yang mempunyai koordinat x yang sama pada saat titik – titik tersebut berbeda satu dengan yang lain, ataupun koordinat x-nya = 0
   $(x, y) + (x, x + y) = O$ untuk semua (x,y) $\in$ $E(F_p)$
4. aturan untuk menambahkan 2 buah titik dengan koordinat x yang berbeda dimana $(x_1,y_1) \in E(F_2^m)$ dan $(x_2,y_2) \in E(F_2^m)$

menjadi 2 buah titik, sehingga $x_1 \neq x_2$, maka $(x_1,y_1) + (x_2,y_2) = (x_3,y_3)$, dimana:

$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$ dalam $E(F_2^m)$,

$y_3 = \lambda.(x_1 + x_3) + x_3 + y_1$ dalam $E(F_2^m)$,

dan $\lambda \equiv \frac{y_1 + y_2}{x_1 + x_2}$ dalam $E(F_2^m)$.

5. aturan untuk menambahkan sebuah titik dengan dirinya sendiri (menggandakan sebuah titik),dimana $(x1,y1) \in E(F_2^m)$ adalah sebuah titik dengan $x1 \neq 0$. sehingga $(x_1,y_1) + (x_2,y_2) = (x_3,y_3)$, dimana :

$x_3 = \lambda^2 + \lambda + a$ dalam $E(F_2^m)$,

$y_3 = x_1^{\ 2} + (\lambda + 1).x_3$ dalam $E(F_2^m)$, dan

$\lambda \equiv x_1 + \frac{y_1}{x_1}$ dalam v $E(F_2^m)$

Himpunan titik – titik pada $E(F_2^m)$ membetuk kelompok dengan sifat komutatif. Aturan ini juga dapat juga dapat dikomputasi dengan menggunakan operasi aritmatika sederhana.

Proses pembangkitan parameter domain kurva elips :

Input : integer $t \in \{56,64,80,96,112,128,192,256\}$

Output : parameter domain kurva elips terhadap F_p:

T = (p,a,b,G,n,h)

Proses :

1. pilih p, bilangan prima, sehingga $[\log_2 p] = 2t$ jika $t \neq 256$, dan $[\log_2 p] = 521$ jika $t = 256$ untuk menentukan bidang F_p
2. pilih elemen $a,b \in F_p$ untuk menentukan kurva elips $E(F_p)$ yang didefinisikan dengan persamaan :

   $E : y^2 \equiv x3 + ax + b \pmod p$

   titik dasar $G = (x_G,y_G)$ pada $E(F_p)$,n bilangan prima dan kofaktor $h = \#E(F_p) / n$, dengan aturan :
   - $4a^3 + 27b^2 / \equiv 0 \pmod p$
   - $\#E(F_p) \neq p$
   - $p^B / \equiv 1 \pmod n$ untuk semua $1 \leq B < 20$
   - $h \leq 4$
3. output T = (p,a,b,G,n,h)

validasi parameter domain kurva elips terhadap F_p

ada 4 metode yang digunakan sebuah entitas U untuk menerima jaminan bahwa parameter domain kurva elips terhadap F_p adalah valid. Metode – metode tersebut adalah :

1. U memvalidasi parameter domain kurva elips terhadap F_p itu sendiri dengan menggunakan primitif validasi yang akan dijelaskan berikutnya
2. U membangkitkan parameter domain kurva elips terhadap F_p itu sendiri dengan menggunakan sistem yang terpercaya dengan menggunakan primitif yang telah dijelaskan sebelumnya.
3. U menerima jaminan dalam cara yang otentik bahwa suatu pihak terpercaya terhadap penggunaan U dalam parameter domain kurva elips terhadap F_p telah memvalidasi parameter – parameter dengan menggunakan primitif validasi yang akan dijelaskan kemudian.
4. U menerima jaminan dalam cara yang otentik bahwa suatu pihak terpercaya terhadap penggunaan U dalam parameter domain kurva elips terhadap F_p telah membangkitkan parameter – parameter dengan menggunakan sistem yang terpercaya dengan menggunakan primitif yang telah dijelaskan sebelumnya.

Primitif validasi parameter domain kurva elips terhadap F_p

Input : parameter domain kurva elips terhadap F_p :

T = (p,a,b,G,n,h)

Dengan integer $t \in \{56,64,80,96,112,128,192,256\}$

Output : indikasi apakah parameter tersebut valid atau tidak.

Proses :

1. cek bahwa p adalah bilangan prima ganjil, sehingga $[\log_2 p]] = 2t$ jika $t \neq 256$, dan $[\log_2 p] = 521$ jika $t = 256$
2. cek bahwa a,b,x_G,dan y_G adalah integer dalam interval $[0,p-1]$
3. cek bahwa $4a^3 + 27b^2 / \equiv 0 \pmod p$
4. cek bahwa $y_G^{\ 2} \equiv x_G^{\ 3} + ax_G + b \pmod p$
5. cek apakah n bilangan prima
6. cek bahwa $h \leq 4$, dan $h = \left\lfloor \left(\sqrt{p+1}\right)^2 / n \right\rfloor$
7. cek bahwa nG = 0
8. cek bahwa $q^B / \equiv 1 \pmod n$ untuk semua $1 \leq B < 20$, dan $nh \neq p$

9. jika salah satu pengecekan salah, maka parameter tersebut tidak valid, jika tidak maka valid.

Domain Parameter Kurva Ellips pada bidang F_2^m

Tupple dari domain parameter dari kurva ellips adalah sebagai berikut :

$T = (m, f(x), a, b, G, n, h)$ dimana :

m : nilai integer yang menspesifikasikan bidang berhingga dari F_2^m. 2 variabel lainnya a,b, $\in F_2^m$, menspesifikasikan kurva ellips E (F_2^m) yang didefenisikan dengan persamaan :

$y^2 + x.y = x^3 + ax^2 + b$ ini F_2^m. Base point dari G adalah = (x_G,y_G) pada E (F_2^m), bilangan prima n yang merupakan pangkat dari G, dan h merupakn kofaktor h = # E(F_2^m)/n

Membuat primitif parameter domain dari kurva ellips pada bidang F_2^m.

Domain paramter dari F_2^m dibuat dengan cara sebagai berikut :

Input

Tingkatan keamanan yang berupa bit didapat dari domain parameter kurva ellips. Tingkatan keamanan ini harus direpresentasikan dengan integer. Nilai tingkat keamanan antara lain :

T $\in$ {56,60,80,96,112,128,192,256}

Output

Domain parameter kurva ellips pada bidang F_2^m.

Aksi

Membuat domain parameter pada F_2^m yang dilakukan dengan cara sebagai berikut :

1. misalkan t' menyatakan integer terkecil yang lebih besar dari t didalam himpunan {64,80,96,112,128,192,256,512}. Pilih m $\in$ {113,131,163,193,233,239,283,409,571} sehingga $2^t < m < 2t'$ untuk mendapatkan F_2^m.
2. Pilih derajat polinomial pada tabel yang akan menerangkan representasi dari F_2^m.
3. Pilih element a,b $\in F_2^m$ untuk menjelaskan kurva ellips E (F_2^m) yang dijelaskan dengan persamaan :
   E : $y^2 + x.y = x^3 + a.x^2 + b$ in F_2^m.
   Sebuah base point G (x_G,y_G) pada E (F_2^m), sebuah bilangan prima n yang merupakan order dari G, dan sebuah integer h yang merupakan kofaktor
   h = #E(F_2^m) / n, mengandung batasan-batasan sebagai berikut :
   - b $\neq 0$ dalam F_2^m
   - #E(F_2^m) $\neq 2^m$
   - $2^{mb} / \equiv 1 \pmod n$ untuk $1 \leq B < 20$
   - h ≤ 4
4. Keluaran T = $(m, f(x), a, b, G, n, h)$

Validasi primitif domain parameter kurva ellips pada F_2^m

Input

Domain parameter kurva ellips pada F_2^m yaitu : T = $(m, f(x), a, b, G, n, h)$

Output

Sebuah indikasi apakah domain parameter dari kurva ellips adalah valid atau tidak

Aksi

Validasi domain parameter dilakukan sebagai berikut :

1. misalkan t' menyatakan integer terkecil yang lebih besar dari t didalam himpunan {64,80,96,112,128,192,256,512}. Pilih m $\in$ {113,131,163,193,233,239,283,409,571} sehingga $2^t < m < 2t'$
2. Periksa bahwa $f(x)$ adalah derajat polinomial yang irreucible yang ada pada tabel diatas
3. Periksa bahwa a,b,x_G, dan y_G derajat polinomial biner m – 1 atau kurang
4. Periksa apakah b $\neq 0$ dalam F_2^m
5. Periksa bahwa $y_G^2 + x_G.y_G = x_G^3 + a.x_G^2 + b$ dalam F_2^m
6. Periksa bahwa n adalah bilangan prima
7. Periksa bahwa h $\leq$ 4 dan bahwa $h = \left[(\sqrt{2^m} + 1)^2 / n\right]$
8. Periksa apakah nG = 0
9. Periksa bahwa $2^{mb} / \equiv 1 \pmod n$ untuk setiap $1 \leq B < 20$ dan bahwa nh $\neq 2^m$
10. Jika ada kesalahan maka munculkan invalid jika tidak maka munculkan valid.
11.

Pasangan kunci kurva ellips

Seluruh kunci public yang digunakan menggunakan pasangan kunci yang dikenal sebagai pasangan kunci kurva ellips (elliptic curve key pair).

Membuat primitif pasangan kunci kurva ellips

Input

Domain parameter dari kurva ellips yang valid yaitu (p,a,b,G,n,h) atau $(m, f(x),a,b,G,n,h)$

Output

Pasangan kunci dari kurva ellips (d,Q) yang sesuai dengan T

Aksi

Pembuatan pasangan kunci melibatkan langkah-langkah berikut :

1. Secara random atau semi random memilih suatu nilai integer d dalam interval [1, n - 1]
2. Hitung Q = dG
3. keluarkan (d,Q)

Validasi kunci public dari kurva ellips

Validasi ini digunakan untuk memeriksa apakah kunci public sudah valid atau tidak

Input

Domain parameter dari kurva ellips yang valid yaitu (p,a,b,G,n,h) atau $(m, f(x),a,b,G,n,h)$ yang sudah valid dan sebuah kunci public kurva ellips Q = (x_Q,y_Q) yang sesuai dengan T

Output

Suatu indikasi apakah kunci public valid atau tidak

Aksi

1. Periksa apakah $Q \neq 0$
2. Jika T merepresentasikan domain parameter kurva ellips pada F_p, periksa apakah x_Q dan y_Q adalah integer yang ada didalam range [1, p -1] dan bahwa :
   $y_Q^2 = x_Q^3 + a.x_Q + b \pmod p$
3. Jika T merepresentasikan domain parameter kurva ellips pada F_2^m periksa bahwa x_Q dan y_Q adalah derajat polinomial biner dimana paling tertinggi adalah m – 1 dan bahwa :
   $y_Q^2 + x_Q.y_Q = x_Q^3 + a.x_Q^2 + b$ dalam F_2^m
4. Periksa bahwa nQ = 0
5. Jika ada kesalahan munculkan invalid jika tidak munculkan valid.

Elliptic Curve Digital Signature Algorithm (ECDSA)

Skema tandatangan yang didesign digunakan oleh 2 pihak yaitu pihak penadatangan, yang selanjutnya disimbolkan dengan U, dan pihak yang akan melakukan verifikasi, yang selanjutnya disimbolkan dengan V.

Skema tandatangan meliputi beberapa operasi antara lain :

a. Operasi penandatanganan
b. Operasi pemverfiksian
c. Prosedur penggunaan setup dan kunci.

Secara umum penerapan skema tandatangan yang dilakukan baik U dan V dapat dijelaskan sebagai berikut :

Pertama-tama U dan V harus menggunakan prosedur setup untuk membangun opsi-opsi mana yang akan digunakan oleh skema tandatangan nantinya. Setelah itu U menggunakan prosedur penggunaan kunci untuk memilih pasangan kunci mana yang akan digunakan dan V harus mendapatkan public key dari U. U akan menggunakan pasangan kunci tersebut untuk mengendalikan operasi penandatanganan, dan V akan menggunakan public key untuk mengendalikan operasi pemverifikasian. Kemudian setiap kali U ingin mengirimkan sebuah pesan, kita katakan M, U harus menggunakan operasi tandatangan ke M dengan menggunakan pasangan kunci yang telah ada untuk mendapatkan tandatangan, kita katakan S, pada pesan M. Setelah selesai barulah mengirimkan M yang sudah ditandatangani kepada V. Akhirnya ketika V menerima pesan tersebut, V harus menggunakan operasi verifikasi terhadap pesan tersebut dengan menggunakan public key dari U untuk memverifikasi bahwa pesan tersebut berasal dari U. Jika autentikasi berhasil maka V menyimpulkan bahwa pesan tersebut benar dari U.

Dibawah ini akan dijelaskan lebih detail mengenai apa-apa saja yang dilakukan pada setiap operasi yang terjadi.

A. **Prosedur setup**

Dalam prosedur setup U dan V harus melakukan hal-hal dibawah ini :

1. U membangun fungsi Hash yang akan digunakan untuk membuat tandatangan.
2. U harus membangun domain parameter dari Kurva ellips (p,a,b,G,n,h) atau (m,f(x),a,b,G,n,h) dengan menggunakan panjang bit tertentu (misalkan 56,64,80,96,112,128,192,256). Domain parameter ini dibangun berdasarkan panjang bit tersebut. U harus memastikan terlebih dahulu bahwa domain parameter yang dimasukkan telah valid menggunakan metode-metode yang ada.
3. V harus menerima fungsi hash dan domain parameter yang dibangun oleh U.

B. **Prosedur Pembangunan kunci**

Dalam prosedur ini U dan V harus melakukan hal-hal berikut :

1. U membentuk pasangan kunci (d_u,Q_u) yang dihubungkan dengan domain parameter yang telah terbentuk pada prosedur setup. Pasangan kunci akan digunakan bersama dengan tandatangan. Pasangan kunci ini digenerate menggunakan primitif-primitif yang ada.
2. V harus mendapatkan kunci public Q_u yang dibuat oleh U.
3. V harus melakukan validasi terhadap kunci public yang digenerate oeh U menggunakan metode yang ada.

C. **Operasi Penandatanganan.**

U menandatangani pesan yang ada dengan menggunakan ECDSA. Hal-hal yang terjadi antara lain :

Input

Operasi penandatanganan (Signing Operation) menjadikan sebuah input string M 8 byte dimana pesan tersebut ditandatangani.

Output

Suatu tandatangan S = (r,s) pada M yang terdiri dari pasangan r dan s yang berupa integer. Jika tidak maka akan terjadi valid.

Aksi

Menandatangani pesan M, dilakukan sebagai berikut :

1. Pilih pasangan kunci ephermal elliptic curve, (k,R) dimana R = (x_R,y_R) yang dihubungkan dengan domain parameter yang dibuat pada saat prosedur setup menggunakan pasangan kunci yang ada.
2. Ubah element x_R ke nilai integer $\overline{x_R}$ menggunakan rutin yang ada.
3. $r \equiv \overline{x_R} \pmod n$. Jika r = 0 kembali ke langkah 1.
4. Gunakan fungsi hash yang terpilih untuk menghitung fungsi Hash. H = Hash(M)
5. Hasilkan nilai integer e dari fungsi H.
6. lakukan penghitungan dengan menggunakan rumus :

   $s \equiv k^{-1}.(e + r.d_U) \pmod n$
7. Keluarkan S = (r,s)

D. **Operasi Verifikasi**

V memverifikasi pesan yang datang dari U dengan menggunakan kunci dan parameter yang dibangun pada saat prosedur setup dan prosedur membuat kunci. Input, Output dan Aksi yang dilakukan adalah sebagai berikut :

Input

1. Octet string M yang ada bersama pesan.
2. keluaran dari operasi tandatangan (S = (r,s))

Output

Suatu tanda apakah tandatangan pada M adalah valid atau tidak.

Aksi

Melakukan verifikasi tandatangan yang dilakukan sebagai berikut :

1. Jika s dan r salah-satunya tidak integer dalam interval [n - 1],munculkan invalid dan keluar.
2. Gunakan fungsi hash yang dibuat selama prosedur setup untuk menghitung nilai hash yang ada yaitu : H = hash(M)
3. Munculkan e dari H yang telah didapat
4. lakukan penghitungan sebagai berikut :

   $u_1 = e.s^{-1} \pmod n$ dan $u_2 = r.s^{-1} \pmod n$
5. Lakukan penghitungan sebagai berikut :

   $R = (x_R,y_R) = u_1G + u_2QU$

   Jika R = 0 maka muculkan invalid dan berhenti.
6. Ubah elememen x_R ke integer $\overline{x_R}$ menggunakan routine pengubahan.
7. assign $V = \overline{x_R} \pmod n$
8. lakukan perbandingan v dan r, jika v = r munculkan valid dan jika $v \neq r$ munculkan tidak valid.

Berikut akan dijelaskan serangan-serangan yang pernah dilakukan pada ECDSA

Serangan-serangan yang pernah atau masih teori terhadap ECDSA

Beberapa tipe serangan yang pernah terjadi pada ECDSA adalah :

1. Serangan yang dilakukan pada permasalahan logaritma elliptic curve discrete (Elliptic Curve Discrete Logarithm Problem)
2. Serangan pada fungsi HASH yang digunakan
3. Serangan lainnya.
4.

Elliptic Curve Discrete Logarithm Problem

Salah satu cara dimana “musuh” atau para penyerang berhasil adalah menghitung private key A (d) dari parameter domain A (q,FR,a,b,G,n,h) dan public key Q. Penyerang secara berturut-turut memalsukan tandatangan A pada setiap pesan yang terpilih. Secara matematis permasalahan dari Elliptic Curve Discrete

Logarithm Problem adalah sebagai berikut (seperti dibawah ini) :
Diberikan sebuah kurva E yang terdefenisi dalam wilayah finite F_q, sebuah titik P ($P \in E(F_q)$) dengan orde n, dan sebuah titik Q dimana Q = lP dimana $0 \le l \le n-1$. Hitung l.

Beberapa serangan yang pernah terjadi :

1. **Naïve Exhaustive Search**
   Pada metode ini, seseorang hanya tinggal menghitung perkalian dari P yaitu P,2P,3P,… sampai Q di dapat. Metode ini untuk kasus terburuk bisa mencapa n langkah.

2. **Algoritma POHLIG-HELLMAN**
   Algoritma ini menurut Pohling and Hellman [81] mencari semua faktorisasi dari n yang merupakan order dari P. algoritma ini akan mengurangi kompleksitas pencarian dari naïve Exhaustive Search menjadi l dibagi dengan factor-faktor prima dari n. bilangan l dapat ditemukan dengan menggunakan teori Chinese Remainder Problem.
   Pesan moral dari algoritma ini adalah untuk membentuk Elliptic Curve Discrete Logarithm Problem (ECDLP) yang paling sulit, seseorang harus memilih sebuah Elliptic Curve dimana orde-nya adalah suatu nilai yang dapat dibagi dengan bilangan n yang besar. Order yang ada seharusnya adalah bilangan prima atau bilangan "hampir prima" (bilangan prima yang dapat dibagi dengan bilangan integer yang kecil).

3. **Algoritma BABY-STEP dan GIANT-STEP**
   Algoritma ini merupakan time-memory trade-off dari metode Exhaustive Search. Algoritma ini membutuhkan penyimpanan dalam memori sebanyak $\sqrt{n}$ dan waktu untuk menjalankannya sekitar $\sqrt{n}$ (untuk kasus terburuk).

4. **Algoritma POLLARD'S RHO**
   Algoritma ini, menurut Pollard [83], merupakan nilai random dari algoritma BABY-STEP dan GIANT-STEP. Untuk menjalankan Algoritma ini dibutuhkan waktu yang hampir sama dengan algoritma BABY-STEP dan GIANT-STEP yaitu sekitar $\sqrt{\pi n/2}$ langkah, namun keunggulan dari algoritma ini dibandingkan dengan algoritma BABY-STEP dan GIANT-STEP adalah tidak terlalu memperhitungkan kapasitas penyimpanan dikarenakan membutuhkan ruang yang relatif kecil. Gallant, Lambert, dan Vanstone [31], dan Wiener dan Zuccherato [111] menunjukkan bahwa algoritma Pollard dapat dipercepat dengan faktor $\sqrt{2}$. Sehingga dengan demikian waktu yang dibutuhkan menjadi ($\sqrt{\pi n}$)/2 langkah.

5. **Algoritma PARALLIZED POLLARD'S RHO**
   Van Oorschot dan Wiener [80] menunjukkan bagaimana Pollard's Rho dapat diparallelkan sehingga ketika algoritma ini berjalan parallel pada r prosesor, waktu yang dibutukan algoritma ini sekitar ($\sqrt{\pi n}$) /2r langkah. Karena itulah, dengan menggunakan r prosesor mempercepat waktu sekitar r kali dari algoritma awal (pada bagian 4).

6. **Metode POLLARD'S LAMDA**
   Ini merupakan metode random lain dari Pollard [83]. Sama seperti metode Pollard's Rho, metode Lambda dapat juga diparallelkan dengan suatu kecepatan linear. Parallel dari metode lambda lebih lambat dibandingkan dengan parallel dari metode rho [80]. Metode lambda adalah, begitupun, lebih cepat dalam situasi dimana logaritma yang digunakan terletak dalam interval [0,b] dari [0,n - 1] dimana b < 0,39n [80].

7. **Multiple Logaritma.**
   R. Silverman dan Staptelon [87] menemukan bahwa jika sebuah ECDLP (Elliptic Curve Distance Logarithm Problem) untuk Kurva Elips E dan base pointnya P dapat dipecahkan dengan menggunakan metode Pollard's Rho yang diparallelkan, kemudian diselesaikan dengan. Kemudia ECDLP yang telah terpecahkan lagi dapat digunakan untuk memecahkan permasalahan ECDLP lainnya. Menurut hasil percobaan yang dilakukan oleh para peneliti menyebutkan bahwa jika ECDLP yang pertama dipecahkan dalam waktu t maka ECDLP yang kedua dapat

dipecahkan dalam waktu ($\sqrt{2}$ - 1)t atau sekitar 0.43t. untuk ECDLP yang ketiga dapat dipecahkan dengan ($\sqrt{3}$ - $\sqrt{2}$)t atau sekitar 0,32t dan begitu seterusnya. Hasil yang diperoleh ini menunjukkan bahwa jika satu ECDLP dipecahkan maka ECDLP lainnya akan lebih mudah dipecahkan.

8. Supersingular Elliptic Curve

Suatu Kurva E dikatakan Supersingular jika pencarian t dari E dapat dibagi oleh p dari F_q. Untuk kurva ini diketahui bahwa k $\leq$ 6. hal ini menyebabkan pengurangan waktu menjadi subexponential-time**.** Untuk alasan ini maka Supersingular Curve tidak dimasukkan kedalam ECDSA.

9. Prime-Field Anomalous Curves.

Sebuah kurva dikatakan prime-field-anomalous jika #E(F_p) = p. Semaev [93], Smart [98], dan Satoh dan araki [88] menunjukkan bagaiman pemecahan yang efisien terhadap kurva ini. Serangan ini tidak dapat digunakan untuk jenis kurva lainnya. Sebaliknya dengan memverifikasi jumlah titik pada suatu Elliptic Curve tidak sama dengan kardinal dari **underlying field,** seseorang dapat dengan mudah meyakinkan bahwa serangan Semaev-Smart-Satoh-Araki tidak digunakan.

10. Curve Defined Over A Small

Misalkan E adalah Elliptic Curve yang terdefenisi pada bidang hingga (F_q). Gallant, Lambert dan Vanstone [31], dan Wiener dan Zaccherato [111] menunjukkan bagaimana algoritma Pollard's rho untuk penghitungan logaritma Elliptic Curve dalam E (F_2^{ed}) dapat dipercepat dengan faktor $\sqrt{d}$. Dengan demikian waktu yang dibutuhkan menjadi ($\sqrt{\pi m / d}$)/2 langkah.

11. Curves Defined Over F_2^m

m adalah bilangan komposit.

Galbraith dan Smart [30], berdasarkan karya Frey [27,28], menjelaskan bagaimana Weil descent dapat digunakan untuk memecahkan ECDLP pada kurva yang terdefenisi pada F_2^m. Kemudian Gaudry, Hess dan Smart [32] memperbaiki ide ini dengan ide ini dengan mengemukakan beberapa kejadian dimana ketika m memiliki pembagi (l) yang kecil, misalkan l = 4, ECDLP untuk kurva yang terdefenisi pada F_2^m dapat dipecahkan lebih cepat dibandingkan dengan algoritma pollard's rho.

12. Non-Applicability Of Index-Calculus Methods.

Disetujui atau tidak terdapat algoritma subexponential-time untuk ECDLP adalah pertanyaan yang penting untuk dijawab, dan merupakan hal yang sangat penting terhadap keamanan dari ECDSA. Namun selama 24 tahun, lebih spesifik lagi selama 16 tahun pada ECDLP, penyelidikan terhadap DLP (Discrete Logarithm Problem) tidak menemukan adanya subexponential-time.

13. Serangan Xedni-Calculus

Serangan ini diperkenalkan oleh J. Silverman [95]. Salah-satu keunggulan dari Xedni-Calculus adalah kemampuannya beradaptasi untuk memecahkan permasalahan permasalahan ordinary logarithm dan pemfaktoran integer. Namun menurut penyelidikan yang dipimpin oleh J. Silverman menyatakan bahwa pada kenyataan serangan ini tidak dapat dilakukan.

14. HyperElliptic Curves

HyperElliptic Curves termasuk dalam keluarga Algebraic Curves of Arbitrary Genus. Karena itulah, Elliptic Curve dapat dipandang sebagai HyperElliptic Curve dari Genus 1. Adleman, DeMarrais dan Huang [1] (Lihat juga dalam Stein, Muller, dan Theil [106]) mempresentasikan algoritma subexponential-time untuk permasalahan logaritma diskrit (Discrete Logarithm Problem) dalam Jacobian dari large genus hyperelliptic curve pada bidang berhingga. Namun dalam kasus Elliptic Curves, algoritma ini lebih buruk dari Naive Exhaustive Search.

Namun menurut hasil percobaan yang telah dilakukan didapat bahwa algoritma yang terbaik untuk ECDLP adalah parallel dari algoritma Pollard's Rho dimana parallel algoritma ini membutuhkan waktu sekitar

$\sqrt{\pi n}$ /(2r) langkah dimana n adalah order bilangan prima dari bilangan dasar P dan r adalah jumlah prosesor yang digunakan.

Serangan pada Hardware

Van Oorschot dan Wiener [80] menjelaskan bahwa kemungkinan pengimplementasian algoritma pollard's rho yang diparallelkan dapat menggunakan hardware. Mereka memperkirakan bahwa jika $n \approx 10^{36} \approx 10^{120}$ maka sebuah mesin dengan r = 330.000 prosesor dapat dibangun sekitar $ 10 juta yang dapat menghitung suatu logaritma diskrit elliptic curve dalam waktu 32 hari. Namun sejak ANSI X9.62 menyatakan bahwa parameter n harus memenuhi $n > 2^{160}$ serangan hardware tidak dapat lagi digunakan dengan teknologi sekarang.

Serangan pada fungsi HASH

Dibawah ini akan dijelaskan beberapa hal yang dapat membuat serangan pada fungsi HASH berhasil :

1. Jika SHA-1 bukan preimage resistant, maka seorang penyerang mampu untuk memalsukan A's signature sebagai berikut : E memilih sembarang nilai integer l, dan menghitung r sebagai koordinat x dari Q+lG mengurangi modulo n. E set E = r dan menghitung e = rl mod n. Jika E dapat menemukan suatu pesan m sehingga e = SHA-1 (m), maka (r,s) adalah tandatangan valid untuk m.
2. Jika SHA-1 bukan collision resistant, maka sebuah entitas A dapat menanggalkan tandatangan sebagai berikut : A pertama sekali membuat 2 pesan yaitu m dan m' sehingga SHA-1(m) = SHA-1(m'). Hal ini menyebabkan pasangan pesan disebut suatu collisiion bagi SHA-1. penyerang akan menandatangani m dan kemudian akan mengklaim bahwa penyerang menandatangani m'.

Serangan lain

1. **Serangan terhadap nilai k yang ada di setiap pesan**.
   Nilai rahasia k yang ada pada setiap pesan dalam ECDSA juga harus dirahasiakan. Karena jika penyerang dapat mempelajari nilai k, nilai k ini akan digunakan oleh A untuk membentuk tandatangan (r,s) pada beberapa pesan m, kemudian E dapat me recover kunci private A karena $d = r^{-1}(ks - e) \bmod m$ dimana e = SHA-1(m). Jika nilai k diketahui oleh penyerang maka kemungkinan besar kunci rahasia A juga dapat diketahui.

2. **Pengulangan penggunaan kunci k pada setiap pesan.**
   Kunci rahasia k yang digunakan untuk menandatangani 2 atau lebih pesan seharusnya dibentuk secara independent satu dengan lainnya. Secara khusus, nilai k yang berbeda seharusnya dibentuk untuk setiap pesan; jika tidak, kunci private, d, dapat ditemukan. Untuk membentuk nilai k yang berbeda-beda kita dapat menggunakan random atau pseudorandom number generator. Untuk membuktikan private key dapat digunakan karena kunci k digunakan berulang-ulang. Misalkan kunci k digunakan untuk membentuk tandatangan ECDSA (r,s1) dan (r,s2) pada 2 pesan m1 dan m2 yang berbeda. Kemudian kita ketahui bahw $s_1 = k^{-1}(e1 + dr) \pmod n$ dan $s_2 \equiv k^{-1} (e2 + dr) \pmod n$, dimana e_1 = SHA-1 (m_1) dan e_2 = SHA-1(m_2). Maka $ks_1 = e_1 + dr \pmod n$ dan $ks_2 = e_2 + dr \pmod n$. Pengurangan pada kedua persamaan diatas, $k(s_1 - s_2) = e_1 - e_2 \pmod n$. Jika $s_1 / \equiv s_2 \pmod n$, dimana dimungkinkan terjadi dengan probabilitas overwhelming, maka $k \equiv (s_1 - s_2)^{-1}(e_1 - e_2) \pmod n$. Dengan demikian seorang penyerang dapat menjelaskan k, dan kemudian hal ini dapat merecover nilai private key nya,d.

3. **Serangan Vaudenay**
   Vaudenay [109] mendemonstrasikan kelemahan DSA (secara teoritis) didasarkan pada pandangannya bahwa fungsi hash digunakan dalam DSA adalah SHA-1 modulo q, tidak hanya SHA-1, dimana q adalah 160-bit bilangan prima. (Karena SHA-1 adalah 160-bit fungsi hash, beberapa nilai output, ketika diubah ke integer, adalah lebih besar dari q). Karena itulah, secara umum, SHA-1(m) $\neq$ (SHA-1 (m) mod q). Kelemahan ini mengizinkan pemalsuan dari satu pesan jika penyerang dapat menyeleksi parameter asal. Kelemahan ini tidak muncul pada ECDSA karena kebutuhan bahwa n (suatu analogi kuantitas ke q dalam DSA) lebih besar dari 2^{160}.

4. **Duplicate-Signature Key Selection**
   Skema tandatangan dapat dikatakan memiliki properti duplikat kunci (duplicate-signature key selection atau DSKS) jika public key A, P_A, yang diberikan dan juga diberikan tandatangan A, S_A, pada pesan M, se penyerang E mampu untuk menyeleksi pasangan kunci yang valid (P_E,S_E) untuk S sehingga $_{SA}$ juga merupakan tandatangan E pada M. Sebagai catatan bahwa hal ini dipenuhi ketika S_E diketahui E. Blake-Wilson dan Menezes [11] menunjukkan bagaimana properti ini dapat digunakan untuk menyerang suatu key protocol yang menggunakan skema tandatangan. Mereka juga mendemostrasikan bahwa jika entitas diizinkan untuk memilih parameter asal (domain parameter) mereka sendiri, maka ECDSA memiliki properti DSKS. Untuk melihat hal ini, misalkan bahwa domain paramter A adalah D_A = (q,FR,a,b,G,n,h), pasangan kunci A adalah (Q_A,d_A), dan (r,s) adalah tandatangan A pada M. Penyerang E memilih sembarang nilai integer c, $1 \le c \le n-1$, misalkan t := $((s^{-1}e + s^{-1}rc) \bmod n) \neq 0$, hitung $X = s^{-1}eG + s^{-1}rQ$ (dimana e = SHA-1(M)) dan $\bar{G} = (t^{-1} \bmod n)X$. E kemudian membentuk D_E = (q, FR, a, b, $\bar{G}$, n, h) dan $Q_E = c\bar{G}$. Setelah itu maka kita dengan mudah memverifikasi bahwa D_E dan Q_E adalah valid, dan bahwa (r,s) juga merupakan tandatangan E pada M.

Kesimpulan

Beberapa kesimpulan yang dapat kamu tuliskan antara lain :

1. ECDSA merupakan salah-satu variasi algoritma tandatangan digital yang ada saat ini.
2. ECDSA merupakan penggabungan algoritma Elliptic Curve Cryptography dengan Algoritma tandatangan digital.

Referensi

[1] Jhonson Don, Menezes Alfred, Vanstone Scott www.comms.scitech.susx.ac.uk/fft/crypto/ecdsa.pdf diakses tanggal 7 - Januari – 2005 pukul 16 : 05

[2] A Certicom White Paper www.comms.scitech.susx.ac.uk/fft/crypto/ECC_SC.pdf diakses tanggal 7 - Januari – 2005 pukul 16 : 10

[3] http://www.faqs.org/rfcs/rfc3278.html diakses tanggal 7 - Januari – 2005 pukul 16 : 15