Implementasi *Pollard's Rho* dengan Algoritma Brent pada Lapangan Galois Prima

Chalvin
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514032@std.stei.itb.ac.id

***Abstract*—Pada era internet saat ini, kriptografi menjadi suatu hal yang sangat penting untuk menjamin keamanan penyeluran informasi. Banyak orang berusaha untuk memecahkan sistem kriptografi yang digunakan. Pada kriptografi kurva eliptik misalnya, terdapat sebuah serangan *brute force* bernama *pollard's rho*. Pada penelitian ini, akan diimplementasikan *pollard's rho* menggunakan dua buah metode pencarian siklik, algoritma floyd dan algoritma brent.**

Kata kunci—Elliptic Curve Cryptography, Pollard's Rho, Cycle-Finding, Cryptography attack

I. PENDAHULUAN

Pada era internet yang mudah diakses ini, pertukaran informasi menjadi pusat dari banyak kegiatan. Sosial media, hiburan, bisnis, dan aplikasi-aplikasi sosial lainnya bergerak dengan memanfaatkan pertukaran informasi.Terbukanya akses internet juga membuka cara baru bagi penjahat untuk melakukan aksinya. Tanpa kita tahu, bisa saja ada orang yang menyadap pertukaran informasi yang kita lakukan. Hal ini tentu sangat berbahaya terutama jika penyerang berhasil menyadap data-data rahasia kita. Oleh karena itu, dibutuhkan suatu metode untuk menjaga kerahasiaan data yang dikirim.

Salah satu solusi dari masalah tersebut terdapat dalam ilmu kriptografi. Kriptografi berasal dari bahasa Yunani, *kryptós* yang berarti tersembunyi atau rahasia dan γράφειν yang berarti ilmu. Kriptografi sendiri berarti suatu ilmu yang mempelajari cara-cara untuk mengamankan informasi dari pihak ketiga. Pada era modern ini, telah banyak produk dari ilmu ini yang berjasa dalam mengamankan pertukaran informasi kita. *Advanced Encryption Standard* (AES), *Rivest Shamir Adleman* (RSA), SHA1 merupakan beberapa contoh dari kontribusi kriptografi dalam membuat kita merasa aman dalam komunikasi kita. *Eliptical Curve Cryptography* (ECC) juga merupakan contoh produk kriptografi yang saat ini sedang dikembangkan.

Eliptical Curve Cryptography adalah suatu algoritma kriptografi kunci publik, seperti RSA, yang memanfaatkan suatu struktur aljabar bernama *eliptical curve. Eliptical Curve* sendiri merupakan suatu kurva eliptik yang dibangun diatas suatu lapangan. Umumnya, sistem kriptografi memakai kurva eliptik yang dibangun diatas langan yang terbatas. ECC mulai dipakai karena ECC menawarkan banyak sekali grup *finite* yang komutatif yang, walaupun besar, dapat dikomputasikan dalam waktu yang wajar. Keamanan dari sistem kriptografi ini terletak pada masalah matematika yang belum terpecahkan yang bernama masalah logaritma diskrit. Sayangnya, kriptografi yang memanfaatkan ilmu matematika yang lebih rumit seperti ini pun tidaklah sempurna.

Ada beberapa cara yang dapat dilakukan penyerang untuk memecahkan kriptografi yang digunakan. Cara paling naif adalah mencoba semua kemungkinan kata sandi yang digunakan dalam sistem kriptografi yang dibuat. Hal ini tentu memiliki peluang keberhasilan yang rendah. Selain itu, waktu bermain peran penting dalam kasus ini karena umumnya teknik ini dapat membutuhkan waktu lebih lama dari umur alam semesta itu sendiri untuk dipecahkan. Umumnya, penyerang akan berusaha mencari teknik-teknik yang dapat mengurangi waktu yang dibutuhkan hingga tinggal beberapa menit atau minggu saja.

Pada ECC, telah berkembang suatu algoritma untuk menyerang sistem ini yang bernama *pollard's rho algorithm.* Awalnya, algoritma ini dipakai sebagai algoritma untuk menfaktorkan bilangan bulat. Namun, akhir-akhir ini algoritma ini dimodifikasi sehingga mampu menangani faktorisasi pada *eliptic curve* juga. Dalam implementasinya, algoritma ini memakai suatu algoritma pendeteksi siklik bernama *Floyd's cycle-finging algorithm*. Algoritma ini dapat menghasilkan kinerja yang cukup memuaskan. Akan tetapi, ada satu lagi algoritma pendeteksi siklik yang dapat digunakan yaitu *Brent's cycle-finding algorithm.* Pada beberapa kasus, *Brent's cycle-finding algorithm* dapat mengalahkan waktu yang dibutuhkan oleh *Floyd's cycle-finging algorithm* dalam mencari siklik. Dalam makalah ini, akan diteliti hubungan antara *Brent's cycle-finding algorithm* dengan kinerja dari algoritma *pollard's rho.*

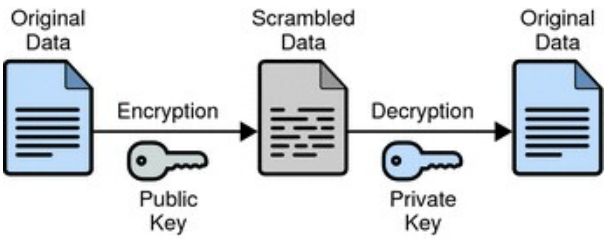
II. DASAR TEORI

A. *Kriptografi Kunci Publik*

Pada awalnya, kriptografi menggunakan kunci yang sama pada proses enkripsi dan dekripsinya. Hal ini kita kenal sebagai kriptografi simetri. Kelemahan dari kriptografi ini terletak pada proses transportasi kunci dari pengirim ke penerima. Jika pada suatu titik pada transportasi kunci berhasil dicuri, penyerang dapat dengan mudah memecahkan enkripsi pada pesan tersebut. Pada era digital ini, beberapa algoritma seperti *Diffie Helman* mencoba untuk menyediakan cara pertukaran kunci yang aman walaupun melalui saluran yang tidak aman. Kriptografi kunci publik menjawab permasalahan ini dengan pendekatan yang berbeda.

Kriptografi kunci publik, atau kadang disebut kriptografi asimetrik, adalah algoritma kriptografi dimana kunci yang digunakan pada proses enkripsi dan dekripsi berbeda. Umumnya, kunci publik digunakan untuk proses enkripsi dan dapat disebarluaskan. Kunci privat hanya diketahui oleh orang-orang tertentu dan digunakan dalam proses dekripsi. Hal ini membuat kunci privat, yang dipakai dalam proses dekripsi, tidak perlu ditransportasikan. Yang perlu ditransportasikan cukup kunci publik penerima, yang mana tidak menjadi masalah bila diketahui oleh penyerang. Algoritma-algoritma pada kriptografi kunci publik umumnya tidak dirahasiakan. Keamanan dari kriptografi ini bergantung dari seberapa sulit didapatkannya kunci privat diberikan kunci publiknya. Pada gambar II-1, dapat dilihat proses enkripsi pada kriptografi kunci publication

Gambar II-1 Enkripsi pada kriptografi kunci publik



Kriptografi kunci publik umumnya didasarkan pada masalah-masalah matematika yang sampai saat ini belum ada cara yang efisien untuk dipecahkan. Beberapa contoh permasalahan tersebut antara lain faktorisasi bilangan dan logaritma diskrit pada *Eliptical Curve Cryptography*. Walaupun kriptografi kunci publik kuat karena berdasarkan permasalahan matematika dan tidak memerlukan transportasi melalui jalur yang aman, kriptografi kunci publik tidaklah sempurna.

Kriptografi kunci publik memiliki kompleksitas komputasi yang lebih besar dibandingkan kriptografi simetri. Hal ini membuat kriptografi kunci publik umumnya digunakan untuk mengenkripsi data-data kecil. Umumnya, data yang besar akan dienkripsi terlebih dahulu menggunakan suatu algoritma kriptografi simetri. Kemudian, kunci dari enkripsi tersebut dienkripsi dengan kriptografi kunci publik untuk kemudian ditransportasikan dengan aman bersama dengan pesan yang dienkripsi kepada penerima pesan. Penerima pesan kemudian melakukan dekripsi kunci pesan terlebih dahulu dengan kunci privatnya sebelum mendekripsi isi pesannya sendiri.

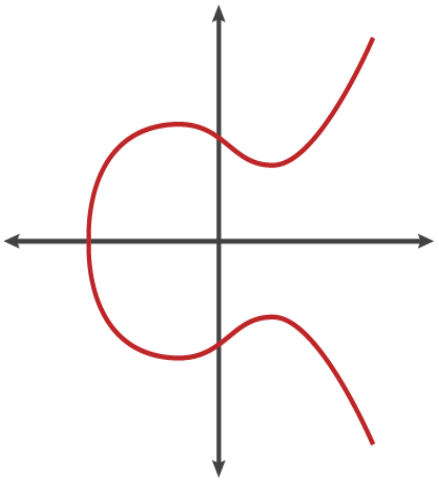
B. *Kurva Eliptik*

Kurva eliptik adalah suatu grup yang secara topologi berbentuk kurva. Kobiltz (1994) mendefinisikan kurva eliptik sebagai berikut. Misalkan K adalah sebuah lapangan dengan karakteristik tidak sama dengan 2 atau 3. Misalkan juga $x^3 + ax + b$ dengan a,b ∈ K adalah suatu polinomial pangkat tiga tanpa akar kembar. Secara formal, kurva eliptik yang dibangun diatas K adalah himpunan titik (x,y), dengan x,y∈ K yang memenuhi persamaan

$$y^2 = x^3 + ax + b. \quad (1)$$

Pada himpunan tersebut, didefinisikan juga sebuah elemen O yang menandakan titik ketakhinggaan. Lapangan K dapat berupa lapangan bilangan real, lapangan bilangan rasional, lapangan bilangan kompleks, maupun lapangan berhingga lainnya seperti lapangan galois. Pada gambar II-2, dapat dilihat contoh dari hasil pemetaan pasangan titik pada kurva eliptik

Gambar II-2 Contoh Kurva Eliptik



Jika K adalah sebuah lapangan dengan karakteristik 2, kurva eliptik yang dibangun adalah himpunan titik (x,y) yang memenuhi persamaan

$$y^2 + cy = x^3 + ax + b. \quad (2)$$

atau

$$y^2 + xy = x^3 + ax^2 + b. \quad (3)$$

dengan O sebagai titik ketakhinggaan.

Pada lapangan dengan karakteristik 3, kurva eliptik yang dibangun adalah himpunan titik (x,y) yang memenuhi persamaan

$$y^2 = x^3 + ax^2 + b + x \quad (4)$$

dengan O sebagai titik ketakhinggaan.

Sebagai sebuah grup, kurva eliptik tentu saja memenuhi aksiom dari grup antara lain:

1. Untuk setiap pasang elemen pada kurva eliptik, hasil operasi dari pasangan elemen tersebut merupakan anggota dari kurva eliptik. Sifat ini disebut sifat ketertutupan,
2. Operasi elemen pada kurva eliptik memiliki sifat asosiatif,
3. Ada suatu elemen I sehingga untuk setiap elemen pada kurva eliptik yang dioperasikan dengan I, baik dari kiri maupun kanan, akan menghasilkan dirinya sendiri. I pada kurva eliptik tidak lain adalah titik ketakhinggaan O,
4. Untuk setiap elemen x pada kurva eliptik, ada suatu elemen x^{-1} sehingga bila x dioperasikan dengan x^{-1}, baik dari kiri maupun dari kanan, akan menghasilkan elemen identitas.

C. Operasi Kurva Eliptik pada Lapangan Z_p

Pada kurva eliptik C yang dibangun diatas lapangan galois Z_p, terdapat operasi sebagai berikut

1. Misalkan $P = (x_p , y_p), Q = (x_q , y_q) \in C$ terdefinisi suatu operatosi penjumlahan (+) sebagai berikut

$$P + Q = R, R \in C \quad (5)$$

Koordinat dari R dapat dikalkulasikan sebagai berikut

$$X_r = \lambda^2 - X_p - X_q \bmod p \quad (6)$$

$$Y_r = \lambda(X_p - X_r) - Y_p \bmod p \quad (7)$$

dengan λ adalah gradien. Pada kasus $P \neq \pm Q$, λ dapat dihitung dengan rumus berikut

$$\lambda = (Y_p - Y_q) / (X_p - X_q) \bmod p. \quad (8)$$

Pada kasus dimana P = Q, λ dapat dihitung dengan rumus berikut

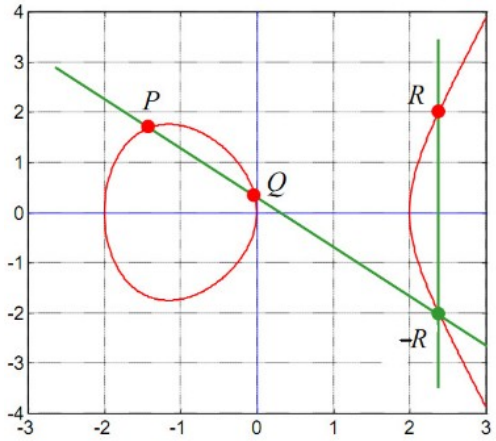
$$X_r = \lambda^2 - 2X_p \bmod p \quad (9)$$

$$Y_r = \lambda(X_p - X_r) - Y_p \bmod p \quad (10)$$

dengan λ adalah gradien yang dapat dihitung dengan rumus berikut

$$\lambda = (3X_p^2 + a) / 2Y_p \bmod p. \quad (11)$$

Gambar II-3 Penjumlahan pada kurva eliptik



Secara geometri, operasi ini dapat kita gambarkan seperti gambar II-3.

2. Misalkan $P = (x_p , y_p), Q = (x_q , y_q) \in C$ serta $-Q = (x_q , -y_q)$ adalah invers dari Q. Terdefinisi suatu operator pengurangan (-) sebagai berikut

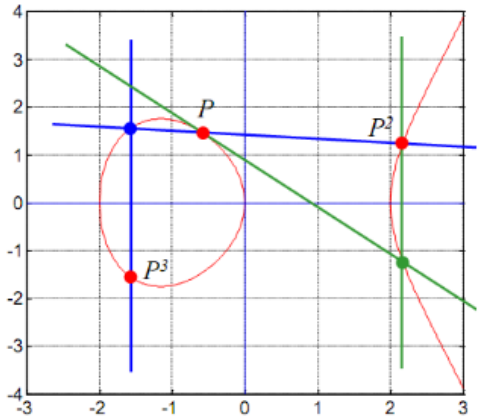
$$P - Q = P + (-Q). \quad (12)$$

3. Misalkan $P = (x_p , y_p) \in C$, serta $k \in Z$ adalah suatu konstanta. Perkalian skalar *kP* pada kurva eliptik pada lapangan galois Z_p didefinisikan sebagai penjumlahan elemen P dengan dirinya sendiri sebanyak k kali. Hal ini secara formal dapat dituliskan sebagai berikut

$$kP = \underbrace{P + P + \ldots + P}_{k \text{ kali}} \quad (13)$$

Secara geometri, perkalian skalar ini dapat digambarkan seperti gambar II-4

Gambar II-4 Perkalian skalar pada kurva eliptik



D. *Kriptografi Kurva Eliptik*

Pada kriptografi kunci publik, sebagian besar implementasinya memanfaatkan permasalahan matematika yang belum dapat dipecahkan secara efisien. Pada RSA, hal ini dilakukan dengan memanfaatkan sulitnya mencari faktorisasi prima dari sebuah bilangan, terutama bila bilangan tersebut cukup besar. Permasalahan ini sudah ada sejak lama dan belum ada solusi efisiennya hingga saat ini. Pada awal RSA diciptakan, RSA sangatlah kuat dan hampir tidak mungkin untuk dipecahkan. Namun seiring berjalannya waktu, kekuatan komputasi pun makin meningkat cepat. Beberapa kunci RSA bahkan sudah mulai berhasil untuk dipecahkan. Tentu saja ini tidak berarti RSA sudah tidak dapat dipakai lagi. Hanya saja, ukuran kunci yang dipakai dalam RSA semakin lama harus semakin besar. Hal ini tidak terlalu masalah pada komputer biasa. Akan tetapi sejak munculnya *internet of thing*, proses enkripsi dan dekripsi RSA pada perangkat dengan perangkat keras yang minimal menjadi suatu masalah tersendiri. Orang-orang kemudian mencoba mencari alternatif dari RSA. Salah satu sistem kriptografi yang dicoba adalah sistem kriptografi kurva eliptik.

Kriptografi kurva eliptik, seperti namanya, adalah suatu sistem kriptografi yang memanfaatkan struktur kurva eliptik sebagai fondasinya. Kriptografi pada sistem ini memanfaatkan permasalahan matematika bernama logaritma diskrit. Pada kurva eliptik, masalah tersebut didefinisikan sebagai berikut. Misalkan C adalah kurva eliptik dengan order n yang dibangun diatas suatu lapangan galois Z_p. Misalkan juga a sebagai pembangun dari C dan suatu elemen b ∈ C. Berapakah nilai k terkecil sehingga

$$b = a^k \mod p \tag{14}$$

Pada grup dengan order yang kecil, tentu saja hal ini tidak terlalu menjadi masalah. Akan tetapi, jika order yang dimiliki oleh C sangatlah besar, perlu waktu lama untuk mengiterasi seluruh elemen dari C. Hal ini tentu membuat kriptografi kurva eliptik menjadi sangat aman, terutama jika kurva eliptik yang dipilih tepat. Merurut Certicom (2001), terdapat parameter-parameter yang perlu diperhatikan saat memilih kurva eliptik yang dibangun diatas Z_p

1. bilangan prima p
2. koefisian persamaan kurva eliptik
3. pembangun kurva eliptik. Kadang disebut basis,
4. orde dari kurva eliptik
5. kofaktor

kekuatan dari sistem kriptografi ditentukan dari pemilihan parameter-parameter tersebut.

Kriptografi kurva eliptik menawarkan keamanan yang setara dengan RSA dengan panjang kunci yang lebih kecil. Hal ini terlihat pada ukuran kunci yang dibutuhkan untuk mengenkripsi kunci AES yang direkomendasikan oleh NIST. Hal tersebut dapat dilihat pada tabel II-1

Tabel II-1 Rekomendasi panjang kunci oleh NIST

Panjang kunci ECC (bits)	Panjang Kunci RSA (bits)	Ratio panjang kunci RSA dengan ECC	Panjang kunci AES (bits)
163	1024	1:6	
256	3074	1:12	128
184	7680	1:20	192
512	15 360	1:30	256

Seperti halnya kriptografi kunci publik lainnya, proses enkripsi dan dekripsi pada kriptografi kurva eliptik memanfaatkan kunci publik serta kunci privat. Kunci publik pada kriptografi kurva eliptik terdiri dari bilangan prima p, generator kurva eliptik a, serta suatu elemen pada kurva eliptik b. Kunci privat adalah suatu konstanta k sehingga

$$b = a^k \mod p \tag{15}$$

E. Serangan *Kriptografi Kurva Eliptik*

Terdapat banyak cara untuk mendapatkan kunci privat dari kunci publik pada kriptografi kurva eliptik. Cara paling naik adalah mengiterasi nilai k dari 1..p-1 sehingga

$$b = a^k \mod p \tag{16}$$

Cara ini dapat dipakai jika orde dari kurva eliptik yang dipakai terbilang kecil. Akan tetapi pada prakteknya, orde yang dipakai dalam kriptografi kurva eliptik memiliki orde yang besar sehingga cara naif seperti iterasi nilai k akan memakan waktu yang sangat lama. Cara lain yang digunakan memanfaatkan *meet in the middle* dan umumnya disebut algoritma *baby-step giant-step.*

Algoritma *baby-step giant-step* berhasil menurunkan kompleksitas pencarian kunci privat dari O(n) menjadi $O(\sqrt{n} \log n)$ dengan n adalah orde dari kurva eliptik yang dipakai. Pada algoritma ini, nilai k yang dicari ditulis ulang sebagai berikut

$$k = im + j \tag{17}$$

dengan $m = \text{ceil}(\sqrt{n})$ dan $0 \le i,j \le m$. Persamaan 16 dapat ditulis ulang sebagai berikut

$$b(a^{-m})^i = a^j \mod p \tag{16}$$

Langkah selanjutnya adalah menghitung seluruh pasangan (a^j,j) untuk $0 \le j \le m$ dan seluruh pasangan $(b(a^{-m})^i,i)$ untuk $0 \le i \le m$. Konstruksi dari pasangan ini akan memakan waktu sebanyak O(m). Kemudian setelah seluruh pasangan nilai dihitung, akan dicari nilai i dan j dari pasangan (a^j,j) dan $(b(a^{-m})^i,i)$ sehingga $b(a^{-m})^i = a^j$. Hal

ini dapat dilakukan dengan *binary search* dengan kompleksitas waktu O(mlog m). Berikut adalah *pseudocode* dari algoritma *baby-step giant-step*

```
Input : Kurva eliptik C dengan orde n,
basis a, dan b, b elemen dari C
Output : suatu nilai k sehingga b = a^k
1.  m ← Ceiling(√n)
2. Untuk semua j ,  0 ≤ j≤ m
   1. Hitung seluruh pasangan (a^j,j)
3. Hitung a^-m
4. y ←b
5. Untuk semua i ,  0 ≤ i≤ m
   1. Cari apakah ada j sehingga y = a^j
   2. Jika ada, kembalikan im + j
   3. Jika tidak, y ←y + a^-m
```

Algoritma ini bersifat deterministik yang artinya algoritma ini pasti berhenti. Sayang, penurunan kompleksitas waktu algoritma ini ternyata tidaklah tanpa efek samping. Kompleksitas ruang yang dibutuhkan pada solusi naif adalah O(1) sementara kompleksitas ruang dari algoritma ini adalah O($\sqrt{n}$). Untuk menjawab masalah tersebut, akan diperkenalkan suatu algoritma baru bernama *pollard's rho*.

Algoritma *pollard's rho* memiliki kompleksitas waktu yang sama dengan algoritma *baby-step giant-step.* Kelebihannya, algoritma ini mempunyai kompleksitas ruang sebesar O(1)! Sayangnya, algoritma *pollard's rho* bersifat probabilistik, yang berarti algoritma ini kemungkinan besar akan berhenti pada waktu yang diharapkan. Akan tetapi, tidak ada jaminan algoritma ini akan berhenti pada waktu tersebut.

Misalkan ada suatu kriptografi kurva eliptik dengan C sebagai kurva eliptiknya, α sebagai pembangkit dari C, dan β adalah elemen dari C. Algoritma ini bekerja dengan terlebih dahulu mempartisi sama rata elemen-elemen pada kurva eliptik ke dalam tiga partisi yang dinamakan S_1, S_2, S_3 . Kemudian, pilih sembarang titik A_0 dimana A_0 elemen dari kurva eliptik dan merupakan kelipatan skalar dari titik basis. Kemudian, kita definisikan suatu fungsi f(x) sebagai berikut

$$A_{i+1} = f(A_i) = \begin{cases} A_i + \alpha \; jika \; A_i \in S_1, \\ 2A_i \; jika \; A_i \in S_2, \\ A_i + \beta \; jika \; A_i \in S_3 \end{cases} \quad (18)$$

Kita dapat menuliskan A_i ke dalam bentuk berikut

$$A_i = c_1 \alpha + c_2 \beta \quad (19)$$

untuk suatu nilai c_1 dan c_2. Ketika ditemukan $A_X = A_Y$ dengan $x \neq y$, kita akan mendapatkan persamaan berikut

$$c_{1X} \alpha + c_{2X} \beta = c_{1Y} \alpha + c_{2Y} \beta. \quad (20)$$

Dari persamaan 20, kita dapatkan persamaan berikut

$$(c_{1X} - c_{2X}) / (c_{1Y} - c_{2Y}) \; \alpha = \beta. \quad (21)$$

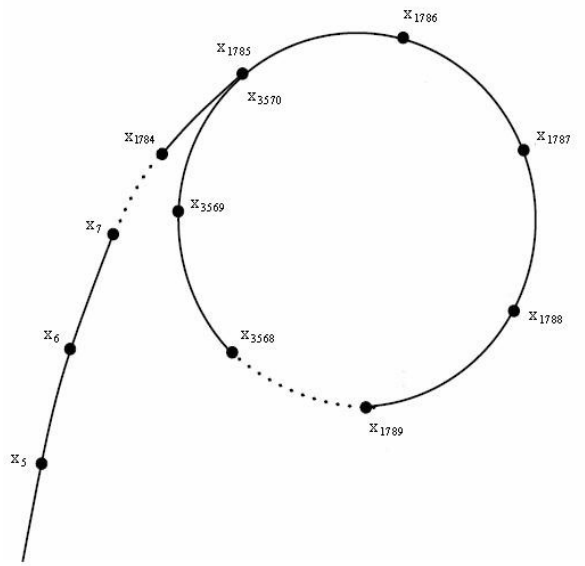
Saat fpb($c_{1Y} - c_{2Y}$, n) = 1, kita dapat dengan mudah mendapatkan nilai k. Saat fpb($c_{1Y} - c_{2Y}$, n) > 1 = d, kita masih bisa mendapatkan nilai k jika nilai d masih relatif kecil. Umumnya, nilai d yang didapatkan masih kecil.

Pencarian dua buah elemen dari barisan A_i dilakukan dengan menggunakan algoritma pencarian siklik. Pada awalnya, algoritma Floyd's cycle-finding dipakai sebagai modul dalam pencarian siklik. Belakangan ini, mulai dikembangkan alternatif algoritma pencarian siklik. Nama *pollard's rho* sendiri diambil berdasarkan bentuk pencarian yang mirip dengan huruf yunani rho. Ilustrasinya dapat dilihat pada gambar II-5

F. *Algoritma Floyd's cycle-finding*

Algoritma *Floyd's cycle-finding* adalah suatu algoritma yang memanfaatkan 2 buah penunjuk. Algoritma ini diciptakan oleh Robert W. Floyd dan disebarkan oleh Donald Knuth pada tahun 1960-an.Algoritma ini kadang disebut juga algoritma kelinci dan kura-kura. Penunjuk

Gambar II-5 Ilustrasi pollard's rho



pertama bergerak lebih cepat dibanding penunjuk pertama layaknya kelinci yang lebih cepat dibanding kura-kura.

Misalkan m adalah indeks awal dimana barisan memiliki siklik. Perhatikan bahwa untuk setiap i≥m, ada suatu k sehingga $x_i = x_{i+kl}$ dengan l adalah panjang siklik.

Pada awalnya, kedua penunjuk ditaruh pada titik mulai. Pada tiap iterasi, penunjuk yang lebih lambat maju 1 langkah dan penunjuk yang lebih cepat maju 2 langkah. Kemudian dilakukan perbandingan antara nilai dari elemen

pada barisan yang sedang ditunjuk oleh kedua penunjuk. Perbedaan indeks antara kedua penunjuk ketika kedua penunjuk menunjukan elemen yang sama tentulah memiliki jarak kl. Selanjutnya, akan dicari cara untuk menentukan l, panjang sikliknya.

Setelah menemukan elemen yang sama, penunjuk yang lebih cepat ditaruh kembali ke basis. Kemudian, masing-masing penunjuk digeser satu ke kanan sambil menjaga jarak mereka agar tetap sebanyal kl. Ketika kedua penunjuk menunjukan nilai yang sama lagi, kita akan mendapatkan nilai m, indeks dimana siklik dimulai.

Terakhir, untuk mendapatkan panjang siklik, taruh penunjuk yang lebih cepat disebelah kanan penunjuk yang lebih lambat. Kemudian, geser penunjuk yang lebih cepat satu per satu ke kanan. Iterasi pertama dimana kedua penunjuk menunjukan nilai yang sama lagi akan menunjukan panjang dari sikliknya. Berikut adalah *pseudocode* dari algoritma ini

```
Input : Barisan dengan siklik yang
dibangun oleh fungsi f(x)
Output : Indeks pertama siklik terjadi
dan panjang siklik

1. kura-kura ← f(x0)
2. kelinci ←f(f(x0))
3. selama kura-kura != kelinci
   1. kura-kura ← f(kura-kura)
   2. kelinci ←f(f(kelinci))
4. m ← 0
5. kelinci ← f(x0)
6. selama kura-kura != kelinci
   1. kura-kura ← f(kura-kura)
   2. kelinci ←f(kelinci)
   3. m ←m+1
7. l ←0
8. kelinci ←f(kura-kura)
9. Selama kura-kura != kelinci
   1. kelinci ←f(kelinci)
   2. l ← l + 1
10. kembalikan nilai m dan l
```

F. *Algoritma Brent*

Algoritma Brent merupakan algoritma alternatif untuk pencarian siklik. Seperti algoritma Floyd, algoritma Brent memakai 2 buah penunjuk. Bedanya, algoritma ini sangatlah dekar dengan perpangkatan 2. Algoritma ini akan membandingkan nilai pada $X_{2^{i}-1}$ dengan X_j, $2^i \leq j \leq 2^{i+1}$ dan akan berhenti ketika kedua penunjuk tersebut telah menunjukan nilai yang sama. Kelebihannya dibanding algoritma Floyd adalah kemampuannya untuk mendapatkan panjang siklik pada tahap yang sama dengan pencarian sikliknya. Berikut adalah *pseudocode* dari algoritma Brent.

```
Input : Barisan dengan siklik yang
dibangun oleh fungsi f(x)
Output : Indeks pertama siklik terjadi
dan panjang siklik
1. power ←1
2. l ← 1
3. kura-kura ←x0
4. kelinci ←f(x0)
5. Selama kura-kura != kelinci
   1. Jika power = l
      1.kura-kura ←kelinci
      2. power ←power * 2
      3. l = 0
   2. kelinci ← f(kelinci)
   3. l ← l + 1
6. m ←0
7. untuk i dari 0 hingga l
   1. kelinci ←f(kelinci)
8. Selama kura-kura != kelinci
   1. kura-kura = f(kura-kura)
   2. kelinci ← f(kelinci)
   3. m ←m + 1
9. Kembalikan nilai m dan l
```

III. HASIL DAN EKSPERIMEN

Pada eksperimen kali ini, akan diimplementasikan serangan *pollard's rho* dengan dua metode. Metode pertama adalah *pollard's rho* standar yang menggunakan algoritma *Floyd's cycle-finding* sebagai modul pencarian sikliknya. Metode kedua adalah *pollard's rho* yang memanfaatkan algoritma Brent sebagai modul pencarian sikliknya. Dalam implementasinya, akan digunakan bahasa pemrograman *python* versi 2.7.14 dengan spesifikasi perangkat keras sebagai berikut

CPU : Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz

Ram : 8 GB

Pada Tabel III-1, dapat dilihat kurva yang dipakai dalam eksperiman kali ini

Tabel III-1 Kurva yang dipakai

bits	a	b	p	order
32	54699	89786	2788201261	2788141523
34	68849	51100	16259177923	16259152279
36	37638	46804	65927012101	65927213977

38	87605	65420	216095015429	216095712139
40	99616	18833	721120664207	721121511997
42	53457	12296	3966621900097	3966621645841
44	97442	60240	14099986717513	14099990462771
46	70769	4077	49723334606591	49723326690097
48	78248	51718	190186607866373	190186617351149
50	63830	5132	843724901827663	843724873985033
60	2533	45915	7.7370076629547 4E+017	7.7370076580 9979E+017

Tabel III-1 Kurva yang dipakai (samb.)

bits	α .x	α .y	β.x	β.y
32	1064666889	423597507	324260157	692351052
34	4050725804	809530138	3608274284	16146957385
36	35241287509	224240893	2484237956	40838666453
38	161019855877	81759848675	146145601336	22643022361
40	526631663594	551103130041	274226809022	617619289539
42	303374847676 4	326707531 9479	216124564 7462	766617115069
44	480933540920 9	125612206 30714	906322152 7554	280680749697
46	913548182429 3	249310086 04645	174288258 18361	329805035908 7
48	405998137873 65	168986606 879248	853968219 48302	109260892784 25
50	482091409481 491	827147510 246646	591813873 981261	282511116325 46
60	1.56894232 616327E+017	6.4009129 5129796E +017	3.7698203 9225105E +017	7.26055397 518852E+017

Tabel III-2 berisi hasil dari eksperimen ini

Tabel III-2 hasil eksperimen

bits	Waktu Floyd (s)	Waktu Brent (s)	Speedup
32	2	1	2
34	5	3	1.666
36	22	13	1.6923
38	26	19	1.3684
40	76	143	0.5314
42	109	87	1.2587
44	387	304	1.2730
46	924	694	1.3314
48	3228	2645	1.2204
50	640	463	1.3822
60	72824	59962	1.2145

IV. ANALISIS

Dari hasil eksperimen pada tabel III-2, terlihat adanya hubungan liniar antara ukuran kunci dengan waktu yang dibutuhkan untuk mendapatkan kunci privatnya. Walaupun demikian, ada kasus dimana waktu yang dibutuhkan justru menurun. Sebagai contoh, pada panjang kunci 42 bit, terlihat implementasi *pollard's rho* dengan algoritma brent memiliki waktu yang lebih kecil dibanding waktu yang dibutuhkan ketika panjang kunci 40 bit. Hal ini mungkin saja disebabkan ketika jumlah iterasi masih sedikit, algoritma brent sudah berhasil menemukan pasangan elemen yang sama.

Jika dibandingkan antara implementasi dengan algoritma floyd dan algoritma brent, umumnya algoritma brent mampu mempercepat kinerja dari *pollard's rho* sebanyak 20-40%, bahkan 60% untuk kasus-kasus kecil. Peningkatan 60% tersebut dapat diabaikan karena ukuran kunci yang kecil membuat penyerangan berhasil dilakukan dengan sangat cepat sehingga pemakaian ukuran detik menjadi tidak tepat. Namun, penggunaan algoritma brent juga tidaklah selalu memberikan hasil yang lebih cepat dibanding penggunaan algoritma floyd. Pada panjang kunci 40 bit misalnya, algoritma brent membutuhkan waktu lebih dari dua kali lipat lebih lama dibanding algoritma floyd. Hal ini sesuai dengan kenyataan bahwa algoritma brent sebenarnya probabilistik sehingga kemungkinan besar waktu yang dibutuhkan akan lebih kecil dibanding floyd, namun tidak ada jaminan akan hal tersebut.

V. KESIMPULAN

Dalam penelitian ini, telah diimplementasikan algoritma *pollard's rho* dengan dua buah metode, algoritma floyd dan algoritma brent. Pada umumnya, implementasi dengan algoritma brent memberikan hasil yang jauh lebih cepat dibanding dengan algoritma floyd. Akan tetapi, algoritma brent tidak dijamin lebih cepat untuk semua jenis kasus. Hal ini sesuai dengan sifat algoritma brent yang probabilistik.

REFERENCES

[1] N. Koblitz, A Course in Number Theory and Cryptography, Springer, May 1994.

[2] A Blumenfeld, "Pollard's rho algorithm for elliptic curves", 2015

[3] Wienardo and Yuliawan,Fajar and Muchtadi-Alamsyah,Intan and Rahardjo,Budi, "Implementation of Pollard Rho attack on elliptic curve cryptography over binary fields" ,2015

[4] Munir, Rinaldi. Elliptic Curve Cryptography. Oktober 2015. Presentasi PowerPoint

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, 25 Mei 2018

Chalvin