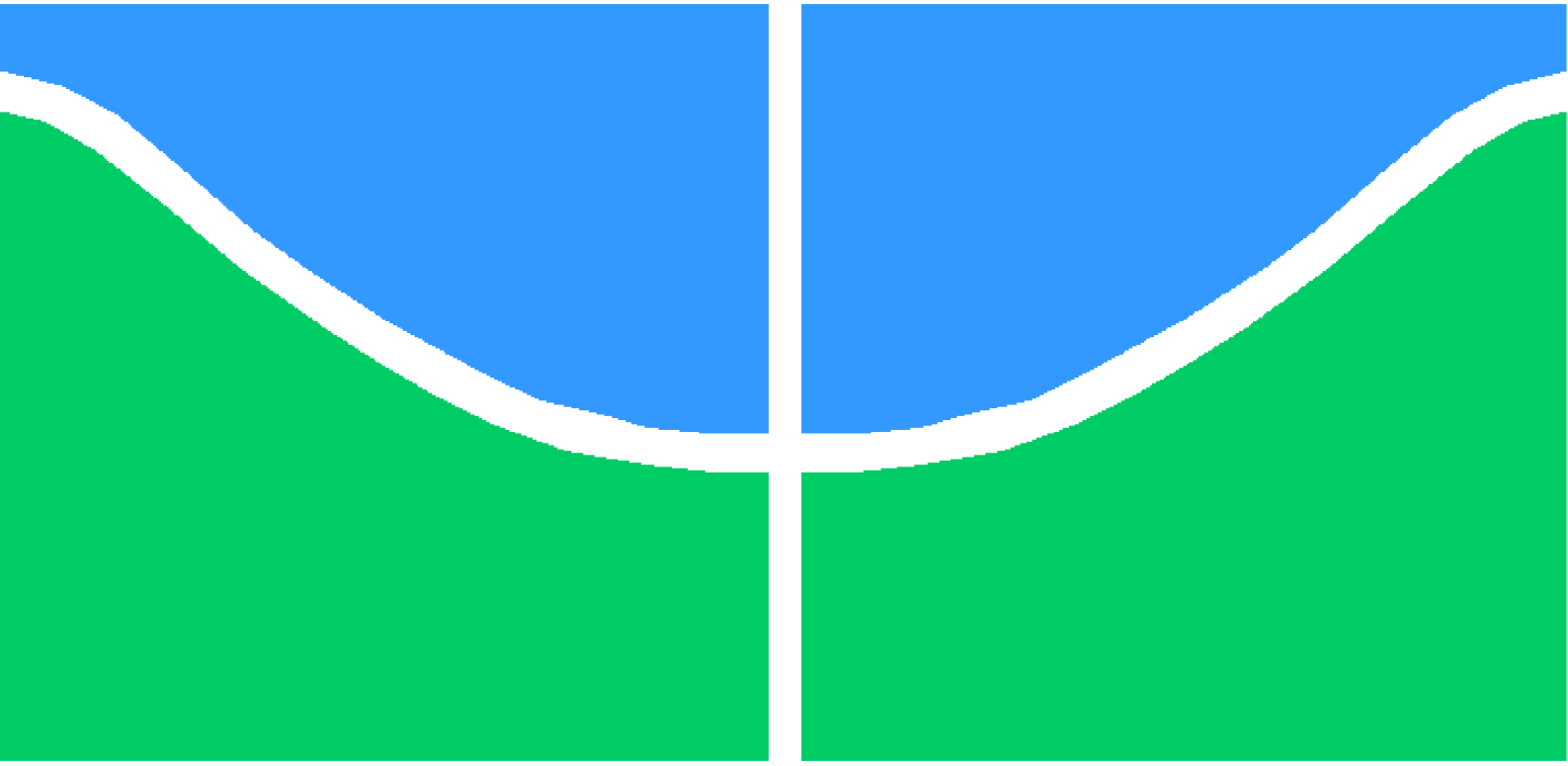

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Implementação e análise de perfomance do algoritmo Pollard-rho para ataque à segurança de curvas elípticas

Autores: Rodrigo Santana Gonçalves e Carlos Alberto Teixeira Junior

Orientador: Prof. Dr. Luiz Augusto Laranjeira

Brasília, DF

2015

Rodrigo Santana Gonçalves e Carlos Alberto Teixeira Junior

Implementação e análise de perfomance do algoritmo Pollard-rho para ataque à segurança de curvas elípticas

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Luiz Augusto Laranjeira

Coorientador: Prof. Dr. Edson Alves Costa Júnior

Brasília, DF

2015

Rodrigo Santana Gonçalves e Carlos Alberto Teixeira Junior

Implementação e análise de perfomance do algoritmo Pollard-rho para ataque à segurança de curvas elípticas/ Rodrigo Santana Gonçalves e Carlos Alberto Teixeira Junior. – Brasília, DF, 2015-

68 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Luiz Augusto Laranjeira

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2015.

1. Curva elíptica. 2. Pollard-rho. I. Prof. Dr. Luiz Augusto Laranjeira. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Implementação e análise de perfomance do algoritmo Pollard-rho para ataque à segurança de curvas elípticas

CDU 02:141:005.6

Rodrigo Santana Gonçalves e Carlos Alberto Teixeira Junior

Implementação e análise de perfomance do algoritmo Pollard-rho para ataque à segurança de curvas elípticas

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 16 de Dezembro de 2015:

Prof. Dr. Luiz Augusto Laranjeira
Orientador

Prof. Dr. Edson Alves Costa Júnior
Coorientador

Prof. Tiago Alves
Convidado 1

Prof. Vinícius Rispoli
Convidado 2

Brasília, DF
2015

Agradecimentos

Eu, Rodrigo Gonçalves, agradeço a Deus acima de tudo, que me deu a oportunidade de estudar em uma universidade federal, que me sustentou durante toda a minha caminhada, e me deu forças para vencer os desafios da vida. Agradeço à minha família que sempre apoiou meus sonhos e objetivos, proporcionando um suporte emocional indispensável, em especial aos meus pais Robertson Gonçalves e Claudevânia de Santana, pois eles foram fundamentais para minha formação acadêmica. Agradeço aos demais amigos e companheiros que também proveram um suporte emocional e intelectual importantes para que eu chegasse até aqui.

Eu, Carlos Alberto, agradeço a Deus acima de tudo e à minha família por todo o esforço que fizeram para me ajudar durante o curso e por todo o incentivo.

Agradecemos aos orientadores, professor Luiz Laranejeira e professor Edson Alves pelos seus suportes. Além de todo corpo acadêmico da Faculdade UnB Gama, em especial os professores de Engenharia de Software por terem contribuído com a nossa formação acadêmica ao longo dos cinco anos de curso nos tornando profissionais qualificados no mercado de trabalho.

Resumo

O avanço da tecnologia nos últimos anos tem causado um grande impacto para a humanidade. Hoje em dia é praticamente inconcebível que exista alguma área que não utilize a tecnologia de alguma forma. Com esse crescimento tecnológico, torna-se imprescindível a criação de meios para garantir a segurança das pessoas. A criptografia tem importância central nesse contexto, pois evita que dados confidenciais sejam descobertos por pessoas mal-intencionadas. Porém, com o avanço da tecnologia, os sistemas criptográficos existentes demandam cada vez mais processamento e complexidade para se manterem seguros. Daí, é necessário buscar alternativas que garantam a segurança dos indivíduos no meio digital e que utilizem menos recursos computacionais. Uma dessas alternativas é a utilização de curvas elípticas em criptografia. Essa alternativa oferece a mesma segurança que a criptografia baseada no RSA, porém, consumindo bem menos recursos computacionais. A segurança da criptografia de curvas elípticas é baseada na dificuldade em resolver o problema do logaritmo discreto. No entanto, existem alguns algoritmos computacionais que resolvem esse problema, podendo comprometer a segurança do sistema criptográfico. O objetivo deste trabalho é explorar os principais conceitos que são empregados na interessante estrutura matemática de curvas elípticas e implementar um desses algoritmos de ataque, a saber, o algoritmo de Pollard-rho.

Palavras-chaves: Criptografia. Criptoanálise. Curva elíptica. Problema do logaritmo discreto. Pollard-rho.

Abstract

The advancement of technology in recent years has caused a great impact on humanity. Nowadays it is almost inconceivable that there is any area that does not use the technology in some way. With this technological growth, it is essential to create means to ensure the safety of persons. Encryption has central importance in this context because it prevents confidential data from being discovered by malicious people. But with the advancement in technology, the existing cryptographic systems require increasingly complex processing and to stay safe. Hence, it is necessary to seek alternatives to ensure the safety of individuals in the digital environment and using less computing resources. One such alternative is the use of elliptic curves in cryptography. This alternative offers the same security that encryption based on RSA, however, consuming far less computing resources. The security of elliptic curve cryptography is based on the difficulty in solving the discrete logarithm problem. However, there are some computational algorithms that solve this problem, which may compromise the security of cryptographic system. The objective of this study is to explore the key concepts that are used in interesting mathematical structure of elliptic curves and implementation of these attack algorithms, namely the Pollard-rho algorithm.

Key-words: latex. abntex. text editoration.

Lista de ilustrações

Figura 1 – Exemplos de curvas elípticas 30
Figura 2 – Soma de pontos em curvas elípticas 31
Figura 3 – Curva elíptica $E_{23}(1, 1)$ 32
Figura 4 – Diagrama da sequência produzida pelo algoritmo Pollard-rho 45
Figura 5 – A sequência gerada pelo algoritmo Pollard-rho paralelizado. 49

Lista de tabelas

Tabela 1 – Número de processadores necessários para ataque de um ano 37
Tabela 2 – Configuração de hardware do computador de desenvolvimento. 42
Tabela 3 – Ferramentas de desenvolvimento empregadas no processo de experimentação. 42
Tabela 4 – Bibliotecas e ferramentas utilizadas no desenvolvimento. 42
Tabela 5 – Cronograma do TCC1 43
Tabela 6 – Cronograma do TCC2 43

Lista de abreviaturas e siglas

ECC	Elliptic Curve Criptography
EC	Elliptic Curve
NIST	National Institute for Standards and Technology
RSA	Rivest-Shamir-Adleman cryptosystem
AES	Advanced Encryption System
DSA	Digital Signature Algorithm
ECDLP	Elliptic Curve Discrete Logarithm Problem
GCD	Greatest Common Divisor

Lista de símbolos

$\in$	Pertence
$\mathcal{O}$	Ponto no infinito
λ	Letra grega lambda
μ	Letra grega mu
π	Letra grega pi
ψ	Letra grega psi
ρ	Letra grega rho
θ	Letra grega theta

Sumário

Introdução 21

1 FUNDAMENTAÇÃO TEÓRICA 23
1.1 Aritmética Modular 23
1.2 Estruturas Algébricas 24
1.2.1 Grupos 25
1.2.2 Subgrupos 25
1.2.3 Teorema de Lagrange 26
1.2.4 Subgrupos cíclicos 26
1.2.5 Homomorfismos de grupos 27
1.2.6 Corpos Finitos 27
1.3 Criptografia 27
1.3.1 Criptografia simétrica 28
1.3.2 Criptografia assimétrica 28
1.4 Curvas Elípticas 29
1.4.1 Definição 29
1.4.2 Leis de grupo para Curvas Elípticas 30
1.4.3 Curvas elípticas sobre corpo finito 31
1.4.4 Quantidade de pontos em uma curva elíptica 32
1.4.5 Criptografia de curvas elípticas 33

2 CRIPTOANÁLISE 35
2.1 Criptoanálise no contexto de curvas elípticas 35
2.1.1 O problema do logaritmo discreto sobre curvas elípticas 35
2.1.2 Algoritmos conhecidos para resolver ECDPL 35
2.1.3 Quebrando ECC em um ano 36

3 METODOLOGIA 39
3.1 Classificação da pesquisa 39
3.2 Metodologia de pesquisa 40
3.3 Atividades de projeto 41
3.4 Ferramentas 41
3.5 Cronograma 42
3.5.1 Prova de conceito 43

4 POLLARD-RHO PARA RESOLVER ECDLP 45

4.1 Pollard-rho original . . . 46
4.2 Pollard-rho com único processador . . . 47
4.2.1 Floyd's cycle-finding . . . 47
4.3 Pollard-rho multiprocessadores . . . 48
4.3.1 Propriedade distintiva . . . 49
4.4 Pollard-rho com automorfismo . . . 50

5 CONSIDERAÇÕES FINAIS . . . 51

Referências . . . 53

APÊNDICES 55

APÊNDICE A – OTIMIZANDO A MULTIPLICAÇÃO DE UM PONTO 57

APÊNDICE B – ALGORITMO DE SCHOOF . . . 59
B.0.0.1 Caso com $l = 2$ . . . 59
B.0.0.2 Caso com $l > 2$ . . . 59

ANEXOS 63

ANEXO A – PRIMEIRO ANEXO . . . 65

ANEXO B – SEGUNDO ANEXO . . . 67

Introdução

Contextualização

A maior parte dos produtos que utilizam a criptografia de chave pública para criptografia e assinaturas digitais utiliza RSA. O tamanho da chave para o uso seguro do RSA tem aumentado nos últimos anos, e isso gerou uma carga de processamento maior sobre as aplicações usando RSA. Esse peso tem ramificações, especialmente para sites de comércio eletrônico, que realizam grandes quantidades de transações seguras. Recentemente, um sistema concorrente começou a desafiar o RSA; criptografia de curva elíptica (ECC - *Elliptic Curve Cryptography*). O ECC já está aparecendo em esforços de padronização, como o IEEE P1363 *Standard for Public-Key Cryptography*. (LEE, 2011)

O atrativo principal do ECC, em comparação com o RSA, é que ele parece oferecer igual segurança com um tamanho de chave muito menor reduzindo assim o overhead do processamento. (STALLINGS, 2011) Em 2003, a principal empresa ligada à ECC (CERTICOM) promoveu um teste para verificação de segurança do criptossistema baseado em curvas elípticas, que foi atacado por 10.000 computadores do tipo Pentium durante 540 dias seguidos. Nesse episódio foi quebrado um sistema com chave de 109 bits, utilizando um ataque conhecido como ataque aniversário (*birthday attack*). Atualmente, o tamanho mínimo de chave recomendado pelo NIST para se obter um bom nível de segurança em ECC é de, pelo menos, 163 bits. (SANGALLI, 2011)

Por outro lado, embora a teoria do ECC já exista há algum tempo, só recentemente esses produtos começaram a aparecer, e tem havido interesse criptoanalítico contínuo em encontrar algum ponto fraco. Por conseguinte, o nível de confiança no ECC ainda não é tão alto quanto do RSA. (STALLINGS, 2011) Isso se deve ao fato de se definir alguns parâmetros antes de executar o algoritmo de cifragem/decifragem, sendo estes parâmetros não comuns em outros criptossistemas. Devemos definir inicialmente um corpo finito e, em seguida, definir a curva elíptica para que possamos gerar o grupo elíptico sobre o qual as operações serão definidas. (SANGALLI, 2011)

Objetivos

1. **Objetivo Geral**

   Objetivo deste trabalho consiste em realizar pesquisas acerca da criptografia de curvas elípticas e implementar o algoritmo Pollard-rho - que se propõe a resolver o problema matemático no qual está baseada a segurança deste sistema criptografico

- e suas variações, comparando o tempo de execução destas variações entre si.

2. **Objetivos Específicos**

   a) Apresentar conceitos matemáticos importantes sobre curvas elípticas e sua aplicação na criptografia.

   b) Demonstrar o problema do logaritmo discreto e algoritmos para resolvê-lo no contexto de ECC.

   c) Implementar o algoritmo Pollard-rho e suas variações.

   d) Utilizar os algoritmos implementados para solucionar o problema do logaritmo discreto para curvas elípticas mais simples, que não tenham utilização prática para criptografia de curvas elípticas. Serão utilizadas curvas elípticas mais simples por dois motivos:

      - Para validar o trabalho.
      - A literatura pesquisada cita a dificuldade e o poder computacional necessário para solucionar tal problema com curvas muito complexas.

   e) Apresentar as comparações de resultados práticos dos desempenhos dos algoritmos implementados.

1 Fundamentação Teórica

Neste capítulo serão mostrados os conceitos matemáticos necessários para o entendimento dos objetivos propostos.

1.1 Aritmética Modular

O **teorema da divisão** diz que: dados dois inteiros a, b, com $b > 0$, ao dividir a por b, obtém-se um único par de inteiros q, r que obedecem ao seguinte relacionamento:

$$a = qb + r \qquad 0 \leq r < b;\ q = \lfloor a/b \rfloor \tag{1}$$

onde $\lfloor x \rfloor$ é o maior inteiro menor ou igual a x e q, r são chamados respectivamente de *quociente* e *resto* ou *resíduo* da divisão de a por b (SANTOS, 2014). A demonstração desse teorema não será abordada nesse trabalho.

A aritmética modular trata da relação entre os inteiros e seus resíduos quando são divididos por algum inteiro positivo denominado **módulo** (LEWINTER; MAYER, 2015). Para exemplificar melhor, considere a sequência de números naturais $S = 3, 4, 5, \ldots$ e o módulo sendo $b = 3$, é possível reescrever qualquer número dessa sequência como $3q$, $3q + 1$ ou $3q + 2$. Dessa forma, qualquer número da sequência se encaixa em uma dessas três classes. Analogamente, se o módulo for k, então haverá k **classes de congruência mod k**. A aritmética modular estuda essa relação entre cada inteiro da sequência S e sua classe de congruência mod k (LEWINTER; MAYER, 2015).

Quando dois inteiros pertencem à mesma classe de congruência, diz-se que estes são *congruentes mod k*. Uma definição mais formal de congruência é: dados dois inteiros a e b, diz-se que a é *congruente* a b módulo m se m divide a diferença $(a - b)$, ou seja, $m \mid (a - b)$. A relação de congruência entre a e b é representada pela notação abaixo.

$$a \equiv b \pmod{m} \tag{2}$$

Consequentemente, se m não divide a diferença $(a - b)$, então a é *incongruente* a b *mod* m, e representa-se por $a \not\equiv b \pmod{m}$ (SANTOS, 2014).

Seja um inteiro positivo n e a, b, c inteiros quaisquer, a congruência tem as seguintes propriedades (STALLINGS, 2011):

1. $a \equiv b \pmod{n}$ se $n|(a - b)$.
2. $a \equiv b \pmod{n}$ implica $b \equiv a \pmod{n}$.

3. $a \equiv b \pmod{n}$ e $b \equiv c \pmod{n}$ implica $a \equiv c \pmod{n}$.

Por definição, o operador $\pmod{n}$ mapeia todos os inteiros para o conjunto de inteiros $\{0, 1, ..., (n-1)\}$. Daí pode-se realizar operações aritméticas dentro dos limites desse conjunto, técnica conhecida como aritmética modular. A aritmética modular exibe as seguintes propriedades (STALLINGS, 2011):

1. $[a \pmod{n} + b \pmod{n}] \pmod{n} = (a + b) \pmod{n}$
2. $[a \pmod{n} - b \pmod{n}] \pmod{n} = (a - b) \pmod{n}$
3. $[a \pmod{n} \times b \pmod{n}] \pmod{n} = (a \times b) \pmod{n}$

A divisão módulo n não está definida em todos os casos. Seja d é um número inteiro que divide a e b, verifica-se que a relação apresentada na Equação 3, onde $\gcd(n, d)$ é o máximo divisor comum entre n e d.

$$\frac{a}{d} \equiv \frac{b}{d} \left(\mod \frac{n}{\gcd(n, d)} \right) \tag{3}$$

O inverso multiplicativo de a módulo n é o número inteiro b que satisfaz a operação $a \times b = 1 \pmod{n}$, e somente pode ser obtido quando existe $(a, n) = 1$ (isto é, a e n são coprimos). O valor do inverso multiplicativo b pode ser obtido através do Algoritmo de Euclides Estendido (HALIM, 2013).

Outra operação que pode ser definida sobre corpos é a exponenciação modular, que calcula o resto de um número inteiro b quando elevado a um número inteiro k e é dividido por um inteiro positivo m. Assim, b é a base, k o expoente e m o módulo da exponenciação modular, que pode ser expressa como na equação 4.

$$c \equiv b^k \pmod{m} \tag{4}$$

1.2 Estruturas Algébricas

Antes de iniciar os estudos em aritmética de curvas elípticas, é importante fixar alguns conceitos de álgebra. De acordo com Hefez,

> "Estruturas algébricas são modelos abstratos para tratar em bloco várias situações matemáticas concretas em que em determinados conjuntos são definidas operações com propriedades semelhantes." (HEFEZ; VILLELA, 2008) (p. 12)

1.2.1 Grupos

Um **grupo** $(G, *)$ é composto por um conjunto G e uma operação binária $*$ sobre os elementos desse conjunto tal que os axiomas abaixo sejam satisfeitos (GILBERT; NICHOLSON, 2004)

1. O conjunto G é **fechado** para a operação $*$

   $a * b \in G$ para todo $a, b \in G$.

2. A operação $*$ é **associativa**

   $(a * b) * c = a * (b * c)$ para todo $a, b, c \in G$.

3. Existe um **elemento identidade** $e \in G$ tal que para todo elemento $a \in G, a*e = a$.

4. Existe um **elemento inverso** a' para todo elemento $a \in G$ tal que $a' * a = e$ (elemento identidade).

Se a operação é **comutativa**, ou seja, se $a * b = b * a$ para todo $a, b \in G$, então o grupo é denominado **abeliano** (ou **comutativo**) em homenagem ao matemático Niels Abel. (GILBERT; NICHOLSON, 2004)

Exemplos de grupos abelianos são $(Z, +)$, $(R, +)$, ambos com identidade 0 e com infinitos elementos. "O número de elementos de um grupo é a sua **ordem**" (COUTINHO, 2014) (pg. 134). Para este trabalho, os grupos de maior interesse são aqueles que possuem ordem finita.

1.2.2 Subgrupos

Seja $(G, *)$ um grupo, e H um subconjunto não vazio de G. Se $(H, *)$ satisfaz todos os axiomas de grupo, então diz-se que $(H, *)$ é um subgrupo de $(G, *)$(COUTINHO, 2014), ou seja

1. Para todo $a, b \in H$, $a * b \in H$.

2. O elemento neutro de G está em H e também é seu elemento neutro.

3. Existe um inverso a' para todo $a \in H$.

Todo grupo possui pelo menos dois subgrupos. Se e indica o elemento neutro de G, então obviamente e é um subgrupo de G. É imediato, também, que o próprio G é um subgrupo de si mesmo. Esses dois subgrupos, ou seja, e e G, são chamados de **subgrupos triviais** de G. (DOMINGUES; IEZZI, 2003) (pg. 154)

1.2.3 Teorema de Lagrange

Um teorema de grande importância para o estudo da criptografia é o teorema de Lagrange, no qual estabelece que, se G é um grupo finito e H é um subgrupo de G, então a ordem de H sempre divide a ordem de G (SHOUP, 2005). A demostração deste teorema não será abordada nesse trabalho, mas seu resultado será de grande utilidade mais à frente, no estudo da criptografia de curvas elípticas.

1.2.4 Subgrupos cíclicos

Seja o grupo finito $(G, *)$ e um elemento $a \in G$. É possível realizar a operação $*$ do grupo repetidas vezes sobre o elemento a, obtendo

$$a^k = a * a * \ldots * a \quad \text{(k vezes)}$$

Pode-se dizer que essa é a k-ésima *potência* de a (mesmo que a operação $*$ do grupo não seja equivalente à multiplicação). É possível criar um conjunto com as potências de a, obtendo

$$H = \{e, a, a^2, a^3, \ldots\}$$

que é um subconjunto de G. Como o conjunto G é um conjunto finito, então o conjunto H também é finito e existem inteiros positivos $n > m$, tais que $a^m = a^n$. Considere que o inverso de a seja a', multiplicando esta equação por $(a')^m$ obtem-se $a^{n-m} = e$, que é o elemento neutro.(COUTINHO, 2014) Daí pode-se afirmar que

1. Dado um elemento $a \in G$, existe um inteiro positivo k tal que $a^k = e$.
2. Se $a^k = e$, então o inverso de a é dado por a^{k-1}, pois $a * a' = a^k = e$. Por ser uma potência de a, o inverso de a também pertence ao conjunto H.

A **ordem do elemento** $a \in G$ é o menor inteiro positivo k tal que $a^k = e$. Como o elemento a gera o conjunto H, diz-se que a é um **gerador** do conjunto H, e consequentemente, a ordem do conjunto H é igual à ordem de a. Quando um grupo é formado por um elemento gerador, diz-se que tal grupo é um *grupo cíclico*. (COUTINHO, 2014)

Pelo teorema de Lagrange, é possível concluir que a ordem de H divide a ordem de G. Uma importante consequência dessa afirmação é que, se um grupo G possui ordem p prima, então seus subgrupos devem ter ordem 1 ou p. Porém, como um subgrupo H formado pelas potências de um elemento a deve possuir pelo menos o elemento neutro e e um gerador a, ou seja, deve possuir pelo menos dois elementos, então todos os subgrupos de G, com exceção do subgrupo trivial formado pelo elemento neutro, devem ter ordem p. Em consequência disso, qualquer elemento $b \in G$ é um gerador do grupo e todo grupo

de ordem prima é um grupo cíclico, embora nem todo grupo cíclico possua ordem prima, p.e., o grupo formado formado pelo conjunto $G = \{1, 2, 3, 4\}(\text{mod } 5)$ e a operação de multiplicação, tem ordem $k = 4$, porém admite o elemento 2 como gerador. (COUTINHO, 2014)

1.2.5 Homomorfismos de grupos

Sejam os grupos $(G, \cdot)$ e $(H, \cdot)$, uma função $f : G \to H$ é chamada de *homomorfismo* se $f(a \cdot b) = f(a) \cdot f(b)$ onde $a, b \in G$. Se a operação dos grupos for diferente, por exemplo, $(G, \cdot)$ e $(H, \star)$, então a condição é dada por $f(a \cdot b) = f(a) \star f(b)$ (GILBERT; NICHOLSON, 2004).

Sendo $f : G \to H$ uma função de homomorfismo entre os grupos G e H, se a função for bijetora, então ela é denominada *isomorfismo*, e ainda, se os grupos G e H são iguais e a função é um isomorfismo, então ela é denominada *automorfismo* (SHOKRANIAN, 2010).

1.2.6 Corpos Finitos

Corpos finitos são estruturas algébricas compostas por um conjunto numérico $\mathbb{F}$ juntamente com duas operações, sendo uma delas a operação de soma (denotada por $+$) e a outra a operação de multiplicação (denotada por $\cdot$) onde são satisfeitas as seguintes propriedades: (HANKERSON; MENEZES; VANSTONE, 2004)

1. $(\mathbb{F}, +)$ é um grupo abeliano com elemento identidade sendo 0.
2. $(\mathbb{F}\backslash\{0\}, \cdot)$ é um grupo abeliano com elemento identidade sendo 1.
3. Para todo $a, b, c \in \mathbb{F}$, é válida a propriedade distributiva $(a + b) \cdot c = a \cdot c + b \cdot c$.

Se o conjunto $\mathbb{F}$ é finito, então o corpo é denominado *corpo finito*. A *ordem* de um corpo finito indica a quantidade de elementos que constituem o corpo finito. Existe um corpo finito de ordem q se e somente se q é igual a uma potência prima, ou seja, $q = p^m$, onde p é um número primo. O número p é chamado de característica do corpo finito. Se $m = 1$ então o corpo é denominado *corpo primo*, caso $m \geq 2$, o corpo é chamado de *corpo extenso*. (HANKERSON; MENEZES; VANSTONE, 2004).

1.3 Criptografia

Criptografia é a ciência que trata de cifrar a escrita, de modo a torná-la ininteligível para os que não tenham os métodos convencionados para ter acesso a ela. Em Tecnologia da Informação, esta definição é ampliada a fim de englobar não só a escrita, mas qualquer

tipo de documento ou dados que devam ser tratados secretamente. Um Sistema de Criptografia define um sistema em que um texto (ou documento, ou qualquer tipo de dado) é transformado através da criptografia em um texto cifrado (*ciphertext*) ou o texto cifrado é transformado de volta à informação original (*plaintext*), através de algoritmos. A primeira ação é denominada cifragem ou criptografar, e a segunda é chamada de decifração ou decriptação. (PORTNOI, 2005)

Com a criptografia pretende-se garantir que uma mensagem só será lida e compreendida pelo destinatário autorizado, e para isso, é necessário que se cumpram quatro requisitos:

- *confidencialidade*: obtida pela encriptação dos dados, assegura que só os receptores autorizados terão acesso às informações da mensagem.

- *integridade*: assegura que a informação não será alterada durante o processo de transporte da informação. É obtida por meio da assinatura digital.

- *autenticação*: o remetente e o receptor podem confirmar as identidades uns dos outros assim como a origem e o destino da informação. É obtida por meio da assinatura digital e dos certificados.

- *irretratabilidade*: ou não-repúdio é obtida por meio da assinatura digital e certificados, o remetente pode assiná-lo de forma digital, limitando legalmente a responsabilidade.

1.3.1 Criptografia simétrica

A criptografia simétrica foi o primeiro tipo de criptografia criado. Funciona transformando um texto em uma mensagem cifrada por meio da definição de uma chave secreta, que será utilizada posteriormente para decriptar a mensagem, tornando novamente um texto simples. (CAVALCANTE, 2015)

A criptografia simétrica utiliza apenas uma chave para codificar e decodificar uma mensagem. É usada em transmissões de dados em que não é necessário um grande nível de segurança como mensagens enviadas de um computador para outro, nas comunicações entre duas máquinas, no armazenamento da informação em um disco rígido.

A criptografia simétrica é relativamente rápida, contudo como desvantagem, não só o transmissor deve conhecer a chave como também o receptor. Além disso, o volume total dos dados transmitidos é limitado pelo tamanho da chave.

1.3.2 Criptografia assimétrica

Utiliza duas chaves, uma para criptografar os dados (chamada de chave pública) e outra para decifrar os dados (chamada de chave privada). Neste caso a chave pública é divulgada enquanto que a chave privada permanece secreta. Esta técnica é muito utilizada para o envio de mensagens onde se deseja que somente o destinatário, portador da chave privada, consiga ler a mensagem. O emissor da mensagem utiliza a chave pública para criptografar a mensagem e a envia. Quando o receptor receber a mensagem, ele então utilizará a chave privada para decifrar a mensagem. Este esquema provê a confidencialidade dos dados e também autenticação pois somente o proprietário da chave privada será capaz de decifrar a mensagem. Em algoritmos dessa natureza, um dado que é criptografado com uma chave só poderá ser decifrado com a utilização da outra chave e vice-versa.

1.4 Curvas Elípticas

1.4.1 Definição

"Curvas elípticas não são elipses. Elas têm esse nome porque são descritas por equações cúbicas, semelhantes às usadas para calcular a circunferência de uma elipse" (STALLINGS, 2011). Uma curva elíptica E definida sobre um corpo $\mathbb{F}$ é descrita pela equação 5

$$y^2 + axy + by = x^3 + cx^2 + dx + e \tag{5}$$

onde a, b, c, d e e são elementos do corpo $\mathbb{F}$ e x e y são as variáveis da equação e assumem valores dentro de $\mathbb{F}$. A notação utilizada para indicar que uma curva elíptica E está definida sobre um corpo $\mathbb{F}$ é $E(\mathbb{F})$, e o corpo $\mathbb{F}$ é chamado de corpo subjacente da curva (HANKERSON; MENEZES; VANSTONE, 2004). Equações deste tipo são conhecidas como *equações de Weierstrass*. Com frequência, no estudo de criptografia de curvas elípticas, é costuma-se utilizar a forma reduzida de Weierstrass, descrita pela equação abaixo, obtida através de uma mudança de variáveis.

$$y^2 = x^3 + ax + b \tag{6}$$

É imprescindível que os valores de a e b sejam escolhidos tais que $4a^3 + 27b^2 \neq 0$ (SEET, 2007). Essa condição garante que não existe ponto na curva que possua mais de uma reta tangente (HANKERSON; MENEZES; VANSTONE, 2004).

O **conjunto dos pontos** que pertencem à curva elíptica E definida sobre um corpo $\mathbb{F}$ é descrito por:

$$E(\mathbb{F}) = \{(x, y) \in \mathbb{F} \times \mathbb{F} \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\} \tag{7}$$

onde o ponto $\mathcal{O}$ é chamado de *ponto no infinito*. (SEET, 2007). Será visto na próxima seção que, juntamente com uma operação de soma dos pontos da curva elíptica, o conjunto $E(\mathbb{F})$ forma um grupo abeliano com o ponto no infinito sendo $\mathcal{O}$ o elemento identidade (HANKERSON; MENEZES; VANSTONE, 2004).

A Figura 3 apresenta alguns exemplos de curvas elípticas usando a forma normal da equação de Weierstrass.

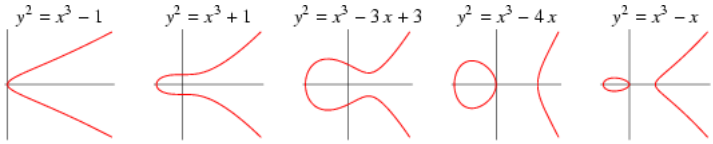

Figura 1 – Exemplos de curvas elípticas

1.4.2 Leis de grupo para Curvas Elípticas

Seja o conjunto de pontos $E(\mathbb{F})$ da curva elíptica E definida sobre um corpo $\mathbb{F}$, é possível definir uma operação sobre esse conjunto de pontos, que será chamada de adição e indicada por $+$. Essa operação de adição e o conjunto de pontos $E(\mathbb{F})$ da curva elíptica formam um grupo abeliano (STALLINGS, 2011).

A forma mais clara de entender a operação de adição sobre curvas elípticas é geometricamente (ver figura 2). Para somar dois pontos P e Q com coordenadas x diferentes, basta ligar uma linha reta entre eles e encontrar o terceiro ponto de interseção R. Pela natureza das curvas elípticas, pode-se observar que existe um único ponto R que é o ponto de interseção (a menos que os pontos P, Q possuam a mesma coordenada x e coordenadas y diferentes). Para formar uma estrutura de grupo, é preciso definir a adição sobre três pontos da seguinte forma: $P + Q = -R$. (STALLINGS, 2011)

Se as coordenadas x dos pontos P e Q são iguais e as coordenadas y são diferentes, então a reta que passa pelos dois pontos não intersecta a curva elíptica em nenhum outro ponto. Para definir a estrutura de grupo, considera-se que essa reta vertical intersecta a curva no ponto no infinito $\mathcal{O}$. (STALLINGS, 2011)

É possível calcular o dobro de um ponto P, ou seja, calcular a soma do ponto com ele mesmo. Para isso desenha-se a reta tangente à curva no ponto P. Essa linha intersecta a curva elíptica num segundo ponto, que será refletido para obter-se o ponto R, que é o resultado da soma. (HANKERSON; MENEZES; VANSTONE, 2004) Observe que esse caso se aplica quando se deseja somar os pontos P e Q onde $P = Q$.

Os axiomas de grupo são satisfeitos da seguinte forma:

1. Para quaisquer pontos $P, Q \in E(\mathbb{F})$, o ponto $P + Q \in E(\mathbb{F})$.

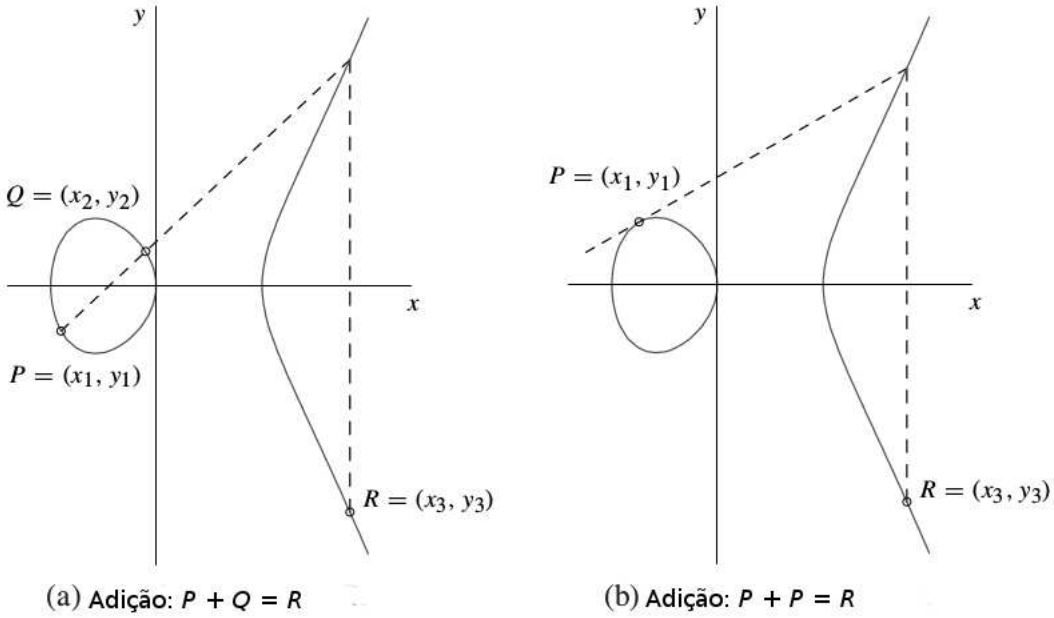

Figura 2 – Soma de pontos em curvas elípticas

2. $\mathcal{O}$ é o elemento identidade. Logo, $P + \mathcal{O} = P$ para todo $P \in E(\mathbb{F})$.

3. Para qualquer ponto P, existe um ponto P' tal que $P + P' = \mathcal{O}$. O ponto P' é denominado *inverso aditivo* de P.

4. A adição é associativa. Logo, $P, Q, R \in E(\mathbb{F})$, observa-se que $(P + Q) + R = P + (Q + R)$

5. A adição é comutativa. Logo, $P, Q \in E(\mathbb{F})$, observa-se que $P + Q = Q + P$.

1.4.3 Curvas elípticas sobre corpo finito

"A criptografia de curva elíptica utiliza curvas elípticas em que as variáveis e coeficientes são todos restritos a elementos de um corpo finito"(STALLINGS, 2011), ou seja, sendo a curva elíptica E definida sobre o corpo finito $\mathbb{F}_p$, seus coeficientes assumem valores inteiros entre 0 e $p-1$, pois os cálculos são realizados módulo p. Quando definida sobre um corpo finito, a equação da curva deve indicar qual é esse corpo através da notação *mod*, como ocorre na equação 8, onde a curva está definida sobre um corpo finito $\mathbb{F}_p$.

$$y^2 \mod p = (x^3 + ax + b) \mod p \tag{8}$$

Pode-se mostrar que um grupo abeliano finito é definido com base no conjunto $E_p(a, b)$, desde que $(x^3 + ax + b) \mod p$ não tenha fatores repetidos. Isso é equivalente à condição

$$(4a^3 + 27b^2) \mod p \neq 0 \tag{9}$$

A Equação 6 tem a mesma forma da Equação 8. As regras para adição sobre $E_p(a, b)$ correspondem à técnica algébrica descrita para as curvas elípticas definidas sobre números reais. Para todos os pontos $P, Q \in E_p(a, b)$.

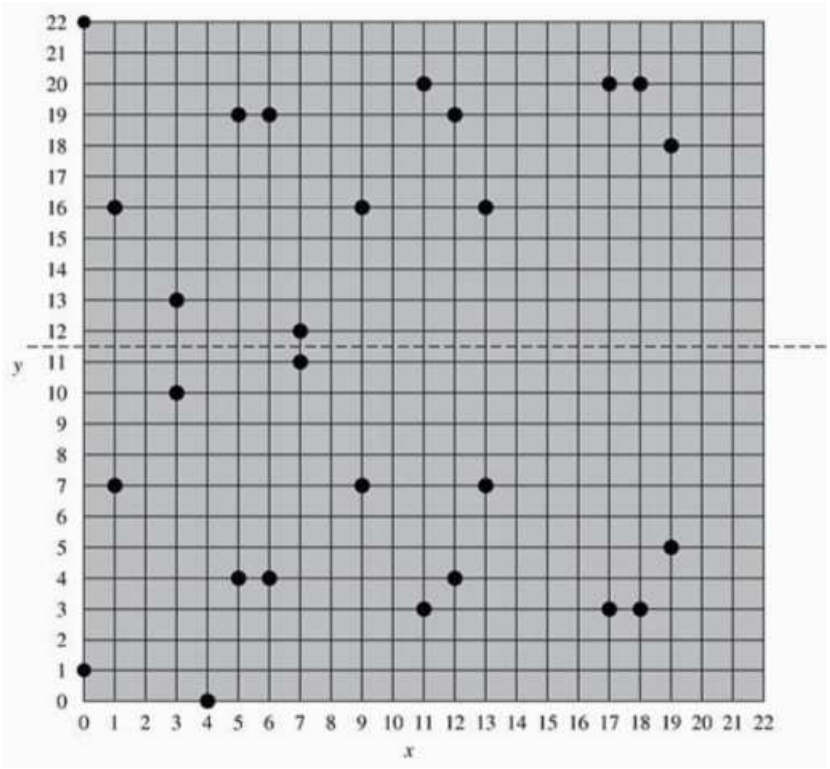

Figura 3 – Curva elíptica $E_{23}(1, 1)$

1. $P + \mathcal{O} = P$.

2. Se $P = (x_P, y_P)$, então $P + (x_P, -y_P) = \mathcal{O}$. O ponto $(x_P, -y_P)$ é o negativo de P, indicado como $-P$.

3. Se $P = (x_P, y_P)$ e $Q = (x_Q, y_Q)$ com $P \neq -Q$, então $R = P + Q = (x_R, y_R)$ é determinado pelas seguintes regras:

$$\begin{aligned} x_R &= (\lambda^2 - x_P - x_Q) \mod p \\ y_R &= (\lambda(x_P - x_R) - y_P) \mod p \end{aligned}$$

onde

$$\lambda = \begin{cases} \left(\frac{y_Q - y_P}{x_Q - x_P}\right) & \text{mod } p, \text{ se } P \neq Q \\ \\ \left(\frac{3x_P^2 + a}{2y_P}\right) & \text{mod } p, \text{ se } P = Q \end{cases}$$

4. A multiplicação é definida como adição repetida; por exemplo, $4P = P+P+P+P$.

1.4.4 Quantidade de pontos em uma curva elíptica

Seja a curva elíptica E definida sobre o corpo finito $\mathbb{F}_q$. O conjunto de todos os pontos que pertencem à curva elíptica é representado por $E(\mathbb{F}_p)$ (ver 7). A quantidade de pontos que compõe esse conjunto é a **ordem de** E **sobre** $\mathbb{F}_q$ e será representada pela notação $\#E(\mathbb{F}_q)$. (HANKERSON; MENEZES; VANSTONE, 2004)

A ordem da curva elíptica é importante para o sistema criptográfico baseado em curvas elípticas, pois sua segurança depende dos fatores primos da ordem (ALVARADO, 2005). Se a ordem da curva elíptica pode ser fatorada em números primos pequenos, a sua segurança pode ser facilmente quebrada utilizando o algoritmo de Pohlig-Hellman, que não será abordado em profundidade nesse trabalho.

O teorema de Hasse fornece limites para a ordem de uma curva elíptica. Esse teorema determina que, sendo a curva elíptica E definida sobre $\mathbb{F}_q$, então

$$q + 1 - 2\sqrt{q} \leq \#E(\mathbb{F}_q) \leq q + 1 + 2\sqrt{q} \tag{10}$$

O **traço** de E sobre $\mathbb{F}_q$ é $t = 2\sqrt{q}$. Sabendo-se o traço da curva definida sobre $\mathbb{F}_q$, é possível calcular a sua ordem. O algoritmo de Schoof(ver B) calcula o traço de uma curva, permitindo saber a sua ordem.

1.4.5 Criptografia de curvas elípticas

Primeiro, selecione um inteiro grande p que seja primo e parâmetros da curva elíptica a e b de acordo com a equação 5 ou 6. Isso define o grupo elíptico de pontos $E_p(a, b)$. Em seguida escolha um ponto base $G \in E_p(a, b)$, cuja a ordem seja um valor muito grande n. $E_p(a, b)$ e G são os parâmetros do criptossistema, conhecidos por todos os participantes.

Um acordo de chaves entre os usuários **A** e **B** pode ser realizado da seguinte maneira:

1. **A** seleciona um inteiro $n_A < n$. Essa é chave privada de **A**, então **A** gera sua chave pública $P_A = n_A \times G$

2. Do mesmo modo, **B** seleciona um inteiro $n_B < n$, sendo essa a chave privada de **B**. E então **B** calcula e divulga sua chave pública $P_B = n_B \times G$.

3. **A** gera a chave secreta $k = n_A \times P_B$. **B** gera a chave secreta $k = n_B \times P_A$

Os dois cálculos na etapa 3 produzem o mesmo resultado, porque

$$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A$$

Para quebrar esse esquema, um atacante teria de ser capaz de calcular k a partir de kG, o que é considerado difícil. Este problema é conhecido como **problema do logaritmo discreto sobre curvas elípticas**.

2 Criptoanálise

Enquanto o campo da Criptografia procura desenvolver um sistema ou um protocolo que permita a privacidade na troca de mensagens, o campo da Criptoanálise é o estudo de métodos para quebrar sistemas de criptografia.

Existem dois lados da Criptoanálise: a que procura criar uma criptografia segura e a que se propõe a quebrar a segurança de um criptossistema. Embora possam ser vistos como antagonistas, o conhecimento de que um criptossistema é suscetível à ataques pode ser uma vantagem. O princípio de Kerchoff's diz que a segurança de um criptossistema deve depender apenas do sigilo do espaço da chave K [1], e não do sigilo do algoritmo de criptografia. (SEET, 2007)

2.1 Criptoanálise no contexto de curvas elípticas

No caso de um criptossistema de curva elíptica, ela deverá depender do sigilo dos inteiros n_A e n_B, que são as chaves privadas de **A** e **B** mencionados na seção 1.4.5. Se estes inteiros são facilmente derivados, então o criptossistema não é mais seguro. Esse tipo de descoberta é classificado como ruptura total, no qual o atacante deduz as chaves secretas. (KNUDSEN, 1998)

2.1.1 O problema do logaritmo discreto sobre curvas elípticas

Seja E uma curva elíptica sobre o corpo finito $\mathbb{F}_p$ e seja P e Q pontos em $E(\mathbb{F}_p)$. O problema do logaritmo discreto sobre curvas elípticas (ECDLP - *Elliptic Curve Discrete Logarithm Problem*) é o problema de encontrar um inteiro x tal que $Q = xP$. Pela analogia com o problema do logaritmo para $\mathbb{F}_p$, denotamos esse inteiro x por

$$xP = Q \tag{1}$$
$$x = \log_P(Q) \tag{2}$$

e chamamos x como logaritmo discreto elíptico de Q em relação a P. (HOFFSTEIN, 2008)

[1] Uma chave de tamanho K oferece um espaço de 2^K.

2.1.2 Algoritmos conhecidos para resolver ECDPL

Ataques mais conhecidos sobre ECC têm complexidade exponencial. Essa afirmação é válida para curvas genéricas e exclui ataques em subclasses especiais, como curvas super-singular e anômalo. A solução para a Equação 2 pode ser calculada usando as seguintes técnicas: (PELZL, 2006)

- Força-bruta: Este método adiciona sequencialmente o ponto $P \in E(\mathbb{F}_p)$ a ele mesmo. A cadeia de adição $P, 2P, 3P, 4P, ...$ acabará por chegar em Q descobrindo assim o valor de n, de acordo com a Equação 1. No pior caso, este cálculo pode levar a $n-1$ passos onde n é da ordem de P. Desta forma, o ataque pode se tornar inviável para prática quando n é muito grande.

- *Baby Step Giant Step* (BSGS): O algoritmo BSGS é uma melhoria à busca por força-bruta. Para n na ordem de P, é necessário que o algoritmo utilize um quantidade de memória temporária por cerca de $\sqrt{n}$ e aproximadamente um adicional de $\sqrt{n}$ passos. No entanto, devido à sua complexidade de memória alta, BSGS acaba não sendo muito interessante.

- Pohlig-Hellman: trabalha com a fatoração do valor n da ordem da curva e utiliza o Teorema do Resto Chinês para o cálculo da série de equações lineares gerados com os fatores de n. Esta abordagem pode ser implementada em conjunto com Pollard's rho. No entanto, a escolha de uma curva cuja ordem é um número primo, pode inibir esse tipo de ataque, pois não é possível fatorar números primos.

- Pollard's rho: O ataque Pollard's rho ou Pollard-rho foi proposto por J. Pollard em 1978. Este ataque consiste em um algoritmo baseado em colisão com base em um percurso aleatório num grupo cíclico, assim, pode ser aplicado ao grupo de pontos gerados por P numa curva elíptica. O percurso aleatório calcula uma trilha de pontos numa curva elíptica e eventualmente termina em um ciclo, revelando a solução para ECDLP. (POLLARD, 1978) Embora tendo um tempo de complexidade similar de $\sqrt{\pi n/2}$ comparado ao BSGS, Pollard-rho é superior devido aos seus desprezíveis requisitos de memória. Em combinação com a paralelização adequada, o método Pollard-rho é o mais rápido ataque conhecido contra ECC. (PELZL, 2006)

  Neste trabalho serão apresentados os métodos Pollard-rho original e suas variantes Pollard-rho com único processador (SPPR), Pollard-rho com multiprocessadores (MPPR) e Pollard-rho com automorfismo.

2.1.3 Quebrando ECC em um ano

De acordo com uma pesquisa (PELZL, 2006), definindo um limite de tempo específico, pode-se calcular o número de processadores necessários. Seguindo a tentativa de

k	**Pentium M**	**XC3S1000**	**ASIC**
80	1	1	-
96	56	21	-
128	$4.86 \cdot 10^6$	$2.55 \cdot 10^6$	$2.05 \cdot 10^4$
160	$3.78 \cdot 10^{11}$	$3.1 \cdot 10^{11}$	$2.48 \cdot 10^9$

Tabela 1 – Número de processadores necessários para ataque de um ano

quebrar ECC com uma chave de tamanho k de bits em um tempo máximo de um ano (365 dias), é possível encontrar a quantidade de processadores necessários representadas pela Tabela 1.

Levaria cerca de 378 bilhões de processadores de propósito geral para completar um ataque de sucesso em um ano em um ECC com chave de $k = 160$ bits. Com hardwares mais preparados utilizando diferentes arquiteturas, levaria menos tempo. No caso do XC3S1000 não se nota grandes diferenças, no entanto, utilizando ASIC é preciso cerca de 100 vezes menos processadores para completar a tarefa. Obviamente, a construção de um cenário tão grande é inviável atualmente.

3 Metodologia

Metodologias específicas colaboram para estabelecer diretrizes e boas práticas na condução do trabalho, conferindo padronização, noções de pesquisa científica, dentre ou tras contribuições. (WOHLIN, 2000)

Com o intuito de guiar a pesquisa de forma adequada, esse capítulo aborda sobre os diversos tipos de metodologias de pesquisa, de modo a definir qual se adequa melhor ao projeto.

3.1 Classificação da pesquisa

A seguir serão apresentados os grupos de classificação de pesquisa, quanto à natureza da pesquisa, abordagem do problema, objetivos de uma pesquisa e procedimentos técnicos.

- Natureza da pesquisa:
  - **Pesquisa básica:** Possui o objetivo de gerar novos conhecimentos para ciência. Neste tipo de pesquisa não é obrigatório que o conhecimento gere um uso prático (TAFNER; SILVA, 2007).
  - **Pesquisa Aplicada:** Visa gerar uma maior compreensão para assuntos práticos dirigidos à solução de problemas específicos (TAFNER; SILVA, 2007).
- Abordagem do problema:
  - **Pesquisa Quantitativa:** O estudo quantitativo considera que tudo pode ser quantificável, ou seja, o estudo que pode ser traduzido em númerose requer uso de técnicas estatísticas para sua análise (TRAVASSOS, 2002).
  - **Pesquisa Qualitativa:** O estudo qualitativo é descritivo, sendo assim, nao pode ser analisada de forma mensurável e sim indutivamente (TRAVASSOS, 2002).
- Objetivos de uma pesquisa:
  - **Pesquisa Exploratória:** é utilizada pelo pesquisador para se familiarizar com um assunto pouco explorado. Ao decorrer ou no final da pesquisa exploratória, o pesquisador poderá estar apto para formular hipóteses (GIL, 2008).

  - **Pesquisa Descritiva:** é usada quando se tem um conhecimento do assunto e se quer descrever um fenômeno. Hipóteses podem ser formuladas com base em conhecimentos prévios, procurando confirmá-las ou negá-las (GIL, 2008).
  - **Pesquisa Explicativa:** é classificada com base em procedimentos técnicos, podendo ser quantitativa ou qualitativa. A pesquisa quantitativa traduz em números os estudos realizados e se utiliza técnicas estatísticas para comprovar os fatos (GIL, 2008).

- Procedimentos técnicos:

  - **Pesquisa bibliográfica:** é realizada a partir do levantamento de referências teóricas sobre o tema. De forma geral, toda pesquisa inicia-se como uma pesquisa bibliográfica, mas há aquelas que dependem exclusivamente desse tipo de pesquisa. Em essência, a conclusão desse tipo de pesquisa é uma compilação das publicações referentes ao tema (TAFNER; SILVA, 2007).
  - **Pesquisa documental:** semelhante à pesquisa bibliográfica, diferencia-se pela natureza das fontes. Tem como base documentos sem tratamento analítico, como tabelas, cartas, fotos, pinturas, dentre outros (GIL, 2008).
  - **Pesquisa experimental:** seleciona grupos de assuntos coincidentes e submete-os a tratamentos diferentes, com isso, ocorre uma verificação se há variáveis estranhas e checa-se as diferenças observadas nas respostas e se são estatisticamente significantes (TAFNER; SILVA, 2007).
  - **Pesquisa *ex-post facto*:** a tradução literal é "de um fato passado", ou seja, a pesquisa *ex-post facto* é realizada após a ocorrência de variáveis no objeto de estudo. A principal característica deste tipo de pesquisa é o fato de os dados serem coletados após a ocorrência dos eventos. Assim, investiga possíveis relações de causa e efeito entre um determinado fato identificado pelo pesquisador e um fenômeno que ocorre postreriormente (GIL, 2008).
  - **Levantamento (*survey*):** é uma investigação realizada em retrospecto, que em seguida, mediante análise quantitativa, chega às conclusões correspondentes aos dados coletados. O levantamento feito com informações de todos os integrantes do universo da pesquisa origina um censo (TRAVASSOS, 2002).
  - **Estudo de caso:** o estudo de caso busca o aprofundamento nas questões propostas, estudando um único grupo ou comunidade, utilizando mais a observação direta do que a interrogação, captando as explicações e interpretações do que ocorre no grupo. No estudo de campo, o pesquisador está inserido no grupo, para poder entender melhor as regras, os costumes e as convenções (TRAVASSOS, 2002).

3.2 Metodologia de pesquisa

Analisando-se o tema proposto, observa-se uma natureza de pesquisa aplicada. Este trabalho de conclusão de curso visa a implementação do algoritmo Pollard-rho e suas variações, carregando consigo diversos conhecimentos de assuntos práticos e gerando uma solução para um problema específico, no caso, o problema do logaritmo discreto no contexto de curvas elípticas.

Considerando os objetivos de estudo deste trabalho, foi incorporada uma abordagem explicativa afim de apresentar valores quantitativos que demonstrem a relação entre os algoritmos implementados e suas diferenças, além de uma análise dos dados e resultados utilizando de técnicas com o intuito de comprovar fatos inerentes à segurança de curvas elípticas.

3.3 Atividades de projeto

Os seguintes pacotes de trabalho, não necessariamente nessa ordem, são fundamentais para compor este projeto de monografia:

- **Pesquisa:** Levantamento e pesquisa acerca das propriedades matemáticas da curva elíptica e potenciais algoritmos que desafiam sua segurança.
- **Levantamento Bibliográfico:** Levantamento bibliográfico sobre os aspectos teóricos e práticos que envolvem a segurança das curvas elípticas.
- **Implementação:** Implementação do algoritmo Pollard-rho e suas variações.
- **Execução:** Execução e ajustes dos algoritmos, assim como os testes de suas execuções.
- **Parte escrita:** Escrita da parte textual do trabalho.

3.4 Ferramentas

A seguir serão apresentadas as ferramentas utilizadas e o ambiente de desenvolvimento, desde as configurações físicas do computador ao software utilizado.

As especificações técnicas do computador utilizado no desenvolvimento são descritas na Tabela 2, a listagem dos software empregados no ambiente de desenvolvimento são descritas na Tabela 3 e, por fim, as bibliotecas e ferramentas de apoio são descritas na Tabela 4.

Ainda com a definição dessas especificações, elas podem sofrer alterações. Por se tratar de uma pesquisa dependente das configurações de hardware, as mudanças que

ocorrerem serão para que o desempenho seja melhorado. Todas as alterações que houverem serão devidamente documentadas na parte escrita do próximo trabalho.

Item	Especificação
Placa Mãe	Dell Inc. 0K5C1C
Processador	Intel Core i7-4510U 2.0 GHz
Memória	16 GiB SODIMM DDR3 Síncrono 1600 MHz (0,6 ns)
Armazenamento	ATA Disk 1 TB
Adaptador de Vídeo	AMD Radeon 2 GB

Tabela 2 – Configuração de hardware do computador de desenvolvimento.

Ferramenta	Nome	Versão	Comentário
Sistema Operacional	Ubuntu Linux	14.04	-
Linguagem	C++	11	O padrão C++11 foi escolhido por possuir mecanismos na própria linguagem que facilitam o desenvolvimento.
Compilador	GCC	3.6.1	A saída de compilação é melhor estruturada e é compatível com os argumentos das ferramentas GNU.
Builder	GNU Make	4.1	-
Editores de Texto	VIM, Sublime Text	7.4, 3083	-

Tabela 3 – Ferramentas de desenvolvimento empregadas no processo de experimentação.

Nome	Versão	Comentário
GNU Multiple Precision v Arithmetic Library	6.0.0	Biblioteca para tratar números com tamanho arbitrário e realização de operações algébricas diversas.
Python	3.4	Usada principalmente para prototipações rápidas.

Tabela 4 – Bibliotecas e ferramentas utilizadas no desenvolvimento.

3.5 Cronograma

O cronograma de execução deste trabalho é apresentado nas Tabelas 5 e 6. Objetivo é conferir uma noção temporal acerca das atividades definidas. Trata-se de uma

visão preliminar, logo, pode haver modificações, ao longo do projeto de acordo com as necessidades.

Tabela 5 – Cronograma do TCC1

	Agosto	**Setembro**	**Outubro**	**Novembro**	**Dezembro**
Elaborar Proposta Inicial	X	X			
Definir Escopo	X	X			
Definir Metodologia		X	X		
Realizar Levantamento Bibliográfico		X	X	X	
Estabelecer Proposta		X	X	X	
Desenvolver Prova de Conceito		X	X	X	
Apresentar TCC1					X
Refinar TCC1					X

Tabela 6 – Cronograma do TCC2

	Março	**Abril**	**Maio**	**Junho**	**Julho**
Implementação PR único processador	X	X	X		
Implementação PR paralelizado	X	X	X		
Implementação PR com automorfismo		X	X	X	
Coletar e Avaliar os Resultados			X	X	
Apresentar TCC2				X	
Refinar TCC2					X

3.5.1 Prova de conceito

Foi desenvolvida a implementação do algoritmo Pollard-rho original em C++ afim de realizar testes de alguns conceitos discutidos neste trabalho. O algoritmo em si será explicitado na Seção 4, a implentação da prova de conceito pode ser acessada através do código fonte no repositório disponível em <https://github.com/rodrigoncalves/pollard-rho>.

4 Pollard-rho para resolver ECDLP

Em 1978, Pollard veio com o método Monte-Carlo[1] para resolver o problema do logaritmo discreto. Desde então, o método foi modificado para resolver o ECDLP. Como o algoritmo Pollard-rho é atualmente o algoritmo mais rápido para resolver o ECDLP, então a segurança do ECC depende da eficiência desse algoritmo. Teoricamente, se o algoritmo Pollard-rho é capaz de resolver o ECDLP eficientemente e em um tempo relativamente curto, então o criptossistema estará inseguro. (SEET, 2007)

A estratégia do algoritmo é produzir uma sequência de termos gerados randomicamente (a_k, b_k, R_k), onde R_k é um ponto na curva E e a_k e b_k estão em $\mathbb{F}_p$ sobre a qual a curva elíptica E está definida. Como $E(\mathbb{F}_p)$ é um grupo finito, a sequência eventualmente irá torna-se periódica e voltará para um termo anterior da sequência – tal ocorrência é chamada de *colisão*. Usa-se essa periodicidade para resolver ECDLP. Como nem sempre a sequência volta para o primeiro termo, um diagrama da sequência parecerá com a letra grega ρ (Ver figura 4). Por este motivo esse método é chamado de Pollard-rho. Essa sequência de termos gerados é chamada de "percurso".

Seja μ o tamanho da calda e λ o tamanho do ciclo. Após um número finito de iterações, obtêm-se os termos $R_k = R_{k+\lambda}$, onde $k > \mu$ e $\lambda > 1$. Neste ponto, já deverá ter sido encontrado uma correspondência entre os termos e poderá aplicar matemáticas discretas para resolver os problemas de logaritmo discreto gerado pelos elementos do conjunto finito.

A seguir serão apresentados o algoritmo de Pollard-rho e suas variações.

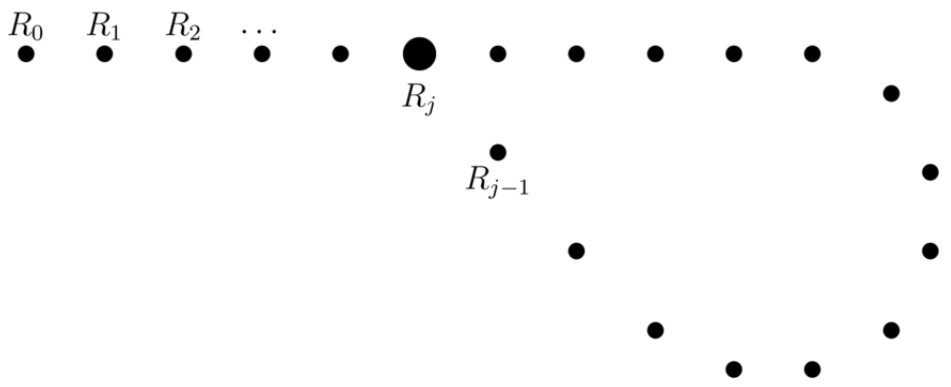

Figura 4 – Diagrama da sequência produzida pelo algoritmo Pollard-rho

[1] Monte-Carlo é um método estatístico que se baseiam em amostragens aleatórias massivas para obter resultados numéricos, isto é, repetindo sucessivas simulações um elevado número de vezes, para calcular probabilidades de forma heurística.

4.1 Pollard-rho original

Seja um grupo $G = E(\mathbb{F}_p)$, tal que a ordem de $|G| = n$, e P e $Q \in E(\mathbb{F}_p)$ tal que $Q = xP$ em G. O objetivo é calcular x. Segue os passos:

1. G é particionado em 3 conjuntos S_1, S_2, S_3 de aproximadamente do mesmo tamanho, em que $\mathcal{O} \notin S_2$.

2. Definir uma função de iteração $f : R \to R$ de um percurso aleatório:

$$R_{k+1} = f(R_k) = \begin{cases} Q + R_k, & R_k \in S_1 \\ 2R_k, & R_k \in S_2 \\ P + R_k, & R_k \in S_3 \end{cases} \tag{1}$$

3. Seja $R_k = a_k P + b_k Q$, e portanto

$$a_{k+1} = \begin{cases} a_k, & R_k \in S_1 \\ 2a_k \pmod{n}, & R_k \in S_2 \\ a_k + 1, & R_k \in S_3 \end{cases} \tag{2}$$

   e

$$b_{k+1} = \begin{cases} b_k + 1, & R_k \in S_1 \\ 2b_k \pmod{n}, & R_k \in S_2 \\ b_k, & R_k \in S_3 \end{cases} \tag{3}$$

4. Os termos iniciais são: $R_0 = P, a_0 = 1, b_0 = 0$ e os pares gerados (R_k, R_{2k}) até encontrar uma correspondência $R_m = R_{2m}$, para qualquer m.

5. Uma vez encontrados, calcule:

$$R_m = a_m P + b_m Q$$
$$R_{2_m} = a_{2_m} P + b_{2m} Q$$

6. Com isso, é possível calcular x:

$$x = (a_{2m} - a_m)(b_m - b_{2m})^{-1} \pmod{n} \tag{4}$$

O termo $(b_m - b_{2m})^{-1}$ em 4 somente é possível calcular quando o $\gcd(b_m - b_{2m}, n) = 1$, ou seja, quando b_m e b_{2m} são coprimos entre si. Caso contrário, o algoritmo retorna um valor indefinido.

4.2 Pollard-rho com único processador

O método original seleciona inteiros aleatórios $a, b \in [0, n-1]$ e armazena as triplas $(a, b, aP+bQ)$ em uma tabela ordenada pelo terceiro componente até que um ponto $aP+bQ$ seja obtida pela segunda vez. Pelo paradoxo do aniversário [2], o número de iterações esperado até que se obtenha uma colisão é de aproximadamente $\sqrt{\pi n/2} \approx 1.2533\sqrt{n}$. A desvantagem desse algoritmo é que a capacidade de armazenamento necessária para o cálculo é de $\sqrt{\pi n/2}$ triplas.

A modificação desse algoritmo encontra os pares (a_m, b_m) e (a_{2m}, b_{2m}) aproximadamente no mesmo tempo que o método original, mas tem desprezíveis requisitos de armazenamento. A ideia é definir uma função de iteração $f : \langle P \rangle \to \langle P \rangle$ de modo que $X \in \langle P \rangle$ e $a, b \in [0, n-1]$ com $X = aP + bQ$. Além disso, f deve ter características de uma função aleatória.

Seja $\{S_1, S_2, ..., S_L\}$ uma partição aleatória de $\langle P \rangle$ em uma quantidade L de conjuntos de aproximadamente do mesmo tamanho. Então um ponto $X \in \langle P \rangle$ pode ser designado a S_j se os cinco *bits* menos significantes da coordenada x de X representam o inteiro $j-1$. Desta forma, define-se $j = H(X)$ se $X \in S_j$, no qual H é uma função de partição. Finalmente, seja $c_j, d_j \in [0, n-1]$ para $1 \leq j \leq L$. Então $f : \langle P \rangle \to \langle P \rangle$ é definido por

$$f(X) = X + c_j P + d_j Q, \text{ onde } j = H(X)$$

O pseudo-algoritmo está descrito em anexo (A).

4.2.1 Floyd's cycle-finding

Pollard-rho é uma variante do algoritmo *Floyd's cycle-finding* (ou busca-ciclos de Floyd), que é importante para detectar a ocorrência de um ciclo em sequências arbitrárias, seja em estruturas de dados ou geradas por sequências pseudo-aleatórias e grafos. Este

[2] Suponha que uma urna tenha n bolas numeradas de 1 a n. As bolas são aleatoriamente retiradas uma de cada vez e colocadas de volta na urna. Portanto, o número esperado para que as bolas se repitam é de aproximadamente de $\sqrt{\pi n/2}$. Se $n = 365$ e as bolas representam diferentes dias do ano, então pode-se dizer que o número esperado de pessoas que tem de ser reunidas em uma sala para que pelo menos duas delas tenham nascido no mesmo dia é de aproximadamente $\sqrt{\pi 365/2} \approx 24$. Esse número é surpreendentemente pequeno e, consequentemente, daí vem a nomenclatura "paradoxo do aniversário". (HANKERSON; MENEZES; VANSTONE, 2004)

algoritmo, também é chamado de algoritmo-"lebre e tartaruga", no qual utiliza apenas dois ponteiros que se movem através da sequência com diferentes velocidades.

Para encontrar $R_i = R_j$, o algoritmo calcula as triplas (a_i, b_i, R_i) e (a_{2i}, b_{2i}, R_{2i}) até que $R_i = R_{2i}$. Para cada iteração, calcula-se $R_{i+1} = f(R_i)$ e $R_{2(i+1)} = f(f(R_{2i}))$, o que significa que este algoritmo utiliza uma quantidade mínima de armazenamento. O algoritmo *Floyd's cycle-finding* é baseado na seguinte ideia. (WANG; ZHANG, 2011) (BAI; BRENT, 2008)

Teorema 1 ((KNUTH, 1997)). Para uma sequência periódica $\{R_0, R_1, R_2, ...\}$, existe um $i > 0$ tal que $R_i = R_{2i}$ e o menor valor de i que pertença ao intervalo $\mu \leq i \leq \mu + \lambda$. μ e λ são o pré-período (calda) e o período (ciclo) da sequência R_i, respectivamente.

O melhor caso desse algoritmo são necessários μ iterações e o pior caso $\mu + \lambda$ iterações. Sob a suposição de que $f : G \to G$ se comporta como uma função verdadeiramente aleatória, o número esperado de iterações antes que se encontre uma colisão é de $\sqrt{\pi^5|G|/288} \approx 1.03\sqrt{|G|}$ (BAI; BRENT, 2008). O ponto chave para esse algoritmo é que precisa-se de três operações de grupo e uma comparação para cada iteração, o que o torna ineficiente.

4.3 Pollard-rho multiprocessadores

Suponha agora que M processadores estão disponíveis para resolver uma instância de ECDLP. Uma abordagem normal seria de executar o algoritmo Pollard-rho independente para cada processador (com diferentes pontos X_0 aleatórios escolhidos inicialmente) até que algum dos processadores finalize. Uma análise cuidadosa mostra que o número esperado de operações de curva elíptica executadas por cada processador até que algum finalize é de $3\sqrt{n/M}$. Assim a aceleração esperada é dada pelo fator $\sqrt{M}$. (HANKERSON; MENEZES; VANSTONE, 2004)

Van Oorschot e Wiener propuseram uma variante do algoritmo Pollard-rho que produz um fator de aceleração M quando M processadores são empregados. A ideia é permitir as sequências $\{X_i\}_{i \geq 0}$ geradas por um processador para colidir com outro. Ou seja, cada processador escolhe randomicamente seu próprio ponto inicial X_0, mas todos processadores utilizam a mesma função de iteração f para calcular os pontos subsequentes X_i. Desta forma, se a sequência de dois diferentes processadores sempre se colidem, então, como ilustrados na Figura 5, as duas sequências serão idênticas daquele ponto em diante. A sequência gerada pelos processadores 3 e 4 se colidem em X. O algoritmo informa a colisão em Y, o primeiro ponto distinto subsequente (OORSCHOT; WIENER, 1996).

O algoritmo *Floyd's cycle-finding* – descrita na seção 4.2.1 – encontra uma colisão na sequência gerada por um único processador. A seguinte estratégia possibilita uma

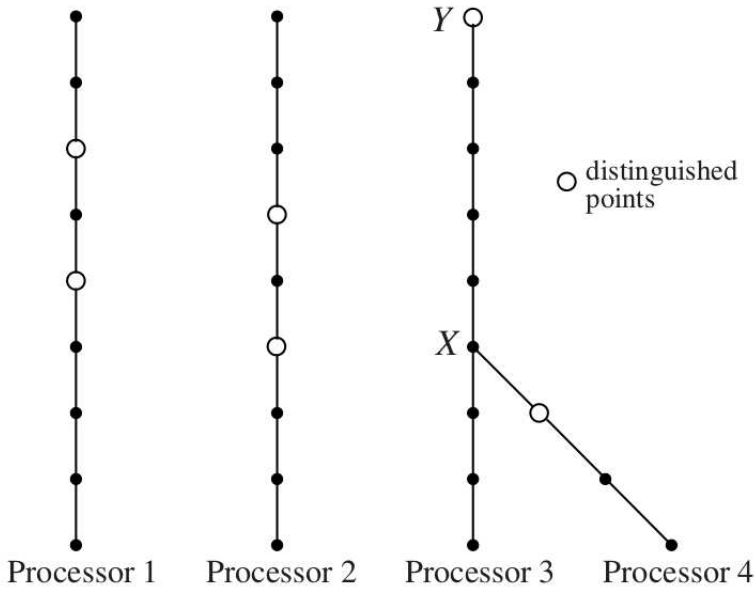

Figura 5 – A sequência gerada pelo algoritmo Pollard-rho paralelizado.

procura eficiente de uma colisão nas sequências geradas por diferentes processadores. Uma *propriedade distintiva* facilmente testável de pontos é selecionada. Por exemplo, um ponto deve ser *distinto* se os primeiros t *bits* de sua coordenada x são iguais a zero. Seja θ a proporção em $\langle P \rangle$ tendo essa propriedade distintiva. Sempre que um processador encontra um ponto distinto, ele transmite o ponto a um servidor central que o armazena em uma lista ordenada. Quando o servidor recebe o mesmo ponto distinto pela segunda vez, ele calcula o logaritmo discreto desejado pela Equação 4 e termina a execução de todos os processadores. O número esperado de passos por processador antes de uma colisão ocorrer é de $(\sqrt{\pi n/2})/M$. Um ponto distinto subsequente é esperado para após $1/\theta$ passos. Consequentemente o número esperado de operações de curva elíptica desempenhadas por cada processador antes de uma colisão de pontos distintos é de

$$\frac{1}{M}\sqrt{\frac{\pi n}{2}} + \frac{1}{\theta} \tag{5}$$

A sua versão paralelizada do algoritmo Pollard-rho obtém um aumento de velocidade que é linear em relação à quantidade de processadores empregados. Uma observação que deve ser feita é que os processadores não tem de comunicar-se entre si, e além disso tem uma comunicação limitada com o servidor central. Portanto, o espaço total necessário no servidor pode ser controlado com uma seleção cautelosa da propriedade distintiva.

O pseudo-algoritmo está descrito em anexo (B).

4.3.1 Propriedade distintiva

Atualmente, o método do ponto distinto é o algoritmo mais eficiente para detectar um ciclo pseudo-aleatório quando n é grande. Para quebrar ECC2K-130, por exemplo, (BAILEY L. BATINA, 2009) define a propriedade distintiva como o *Hamming weight*[3] de uma representação-base normal da coordenada x do ponto que seja menor ou igual a 34. Note que este tipo de definição permite uma rápida verificação para a propriedade distintiva.

4.4 Pollard-rho com automorfismo

Uma maneira descrita por Hankerson, Menezes e Vanstone para acelerar o algoritmo Pollard-rho é fazendo uso de automorfismo.(HANKERSON; MENEZES; VANSTONE, 2004)

Seja o grupo de pontos da curva elíptica $E(\mathbb{F}_q)$. Considere o ponto P de ordem n e o subgrupo gerado por esse ponto, ou seja, $\langle P\rangle$. Seja o automorfismo $\psi : \langle P\rangle \to \langle P\rangle$. A ordem da função ψ é o menor inteiro positivo t tal que $\psi^t(R) = R$ para todo ponto $R \in \langle P\rangle$. Por exemplo, a função $f(x) = -x$ tem ordem $t = 2$, pois $f(f(x)) = x$.

É possível definir uma relação de equivalência $R_1 \sim R_2$ se e somente se $R_1 = \psi^j(R_2)$ para algum $j \in [0, t-1]$. Daí cria-se a classe de equivalência $[R]$, que é descrita por

$$[R] = \{R, \psi(R), \psi^2(R), \ldots, \psi^{l-1}(R)\}$$

onde l é o menor divisor positivo da ordem t da função ψ, tal que $\psi^l(R) = R$.

O método de Pollard-rho com automorfismo consiste em utilizar uma função iterativa f que seja definida nas classes de equivalência. Para isso, será definido um representante $\overline{R}$ para cada classe de equivalência $[R]$ e uma função g tal que

$$g(R) = \overline{f(R)}$$

Sendo conhecido um inteiro $\lambda \in [0, n-1]$ tal que $\psi(P) = \lambda P$ e os inteiros a, b tal que $X = aP + bQ$, então pode-se calcular $\overline{X} = \overline{a}P + \overline{b}Q$ eficientemente por $\overline{a} = \lambda^j a \bmod n$ e $\overline{b} = \lambda^j b \bmod n$.

A função $g(R)$ é utilizada como função de iteração para o algoritmo Pollard-rho paralelizado. Essa modificação garante ao algoritmo uma melhoria no tempo de execução do Pollard-rho paralelizado, dessa forma, a equação 5 passa a ser

$$\frac{1}{M}\sqrt{\frac{\pi n}{2t}} + \frac{1}{\theta}$$

[3] *Hamming weight* de uma *string* é o número de símbolos que são diferentes do símbolo-zero do alfabeto utilizado. No caso de uma *string* binária, é a quantidade de *bits* que são iguais a 1.

ou seja, o uso da função $g(R)$ acarreta uma melhoria no tempo de execução do algoritmo por um fator de $\sqrt{t}$. (HANKERSON; MENEZES; VANSTONE, 2004)

5 Considerações finais

Nesta primeira etapa do trabalho, foram demonstrados os principais pontos quanto ao tema proposto. Em um contexto mais amplo, esta visão abrange aspectos matemáticos e conceitos relacionados à criptografia e a segurança de curvas elípticas, no qual sua característica mais marcante é a dificuldade de se resolver o problema do logaritmo discreto. Em um contexto mais técnico, foi discutida também a implementação dos algoritmos Pollard-rho e suas variações, no que se propõe a resolver o ECDLP definidas sobre corpos finitos.

Espera-se, como resultado deste trabalho, demonstrar a efetividade do algoritmo Pollard-rho no contexto descrito anteriormente. Além disso, é esperado que a versão paralelizada (com multiprocessadores) do algoritmo, seja mais rápida do que a versão original, de forma linearmente proporcional à quantidade de processadores utilizados, ou seja, apresentar valores equivalentes ao descrito na teoria, com $(\sqrt{\pi n/2})/M$ operações.

Outro fato esperado é que, com o uso de funções de automorfismo como função de iteração do algoritmo paralelizado, o tempo de execução e o número de passos necessários para solucionar o problema são reduzidos, conforme indica a literatura referenciada em 5.

Referências

ALVARADO, A. An exposition of schoof's algorithm. Arizona State University, 2005. Citado 2 vezes nas páginas 33 e 59.

BAI, S.; BRENT, R. P. On the efficiency of pollard's rho method for discrete logarithm. Department of Computer Science, Australian Computer Society, 2008. Citado na página 48.

BAILEY L. BATINA, D. J. B. D. V. Breaking ecc2k-130. Cryptology ePrint Archive, 2009. Citado na página 50.

CAVALCANTE, A. L. B. Teoria dos números e criptografia. Departamento de Sistemas de Informação da UPIS, 2015. Citado na página 28.

COUTINHO, S. C. *Números Inteiros e Criptografia RSA*. 2. ed. [S.l.]: IMPA, 2014. Citado 3 vezes nas páginas 25, 26 e 27.

DOMINGUES, H. H.; IEZZI, G. *Álgebra Moderna*. 4. ed. [S.l.]: Atual Editora, 2003. Citado na página 25.

GIL, A. C. Como elaborar projetos de pesquisa - edição 4. 2008. Citado 2 vezes nas páginas 39 e 40.

GILBERT, W. J.; NICHOLSON, W. K. *Modern Algebra with Applications*. 2. ed. [S.l.]: Wiley-Interscience, 2004. Citado 2 vezes nas páginas 25 e 27.

HALIM, S. *Competitive Programming*. 3. ed. [S.l.]: paperback, 2013. Citado na página 24.

HANKERSON, D.; MENEZES, A.; VANSTONE, S. *Guide to Elliptic Curve Cryptography*. [S.l.]: Springer, 2004. Citado 7 vezes nas páginas 27, 29, 30, 33, 47, 48 e 50.

HEFEZ, A.; VILLELA, M. L. T. *Códigos Corretores de Erros*. 2. ed. [S.l.]: IMPA, 2008. Citado na página 24.

HOFFSTEIN, J. *An Introduction to Mathematical Cryptograpfy*. [S.l.]: Springer, 2008. Citado na página 35.

KNUDSEN, L. R. T. Twofish. 1998. Citado na página 35.

KNUTH, D. E. *The Art of Computer Programming 3rd Edition*. [S.l.: s.n.], 1997. Citado na página 48.

LEE, T. C. An implementation of elliptic curve cryptosystem. Department of Computer Science and Information Systems Saginaw Valley State University, 2011. Citado na página 21.

LEWINTER, M.; MAYER, J. *Elementary Number Theory with Programming*. [S.l.]: Wiley, 2015. Citado na página 23.

MCGEE, J. J. René schoof's algorithm for determining the order of the group of points on an elliptic curve over a finite field. Virginia Polytechnic Institute and State University, 2006. Citado 2 vezes nas páginas 59 e 60.

OORSCHOT, P. V.; WIENER, M. On diffie-hellman key agreement with short exponents. 1996. Citado na página 48.

PELZL, J. On the security of elliptic cure cryptosystem against attacks with special-purpose hardware. Departamento de Sistemas de Informação da UPIS, 2006. Citado na página 36.

POLLARD, J. Monte carlo methods of index computation (mod p). 1978. Citado na página 36.

PORTNOI, M. Criptografia com curvas elípticas. Universidade Salvador - UNIFACS, 2005. Citado na página 28.

SANGALLI, L. A. Criptossistemas baseados em curvas elípticas e seus desafios. Campinas, SP, Brasil, 2011. Citado na página 21.

SANTOS, J. P. d. O. *Introdução à Teoria dos Números*. [S.l.]: IMPA, 2014. Citado na página 23.

SEET, M. Z. Elliptic curve cryptography - improving the pollard-rho algorithm. 2007. Citado 4 vezes nas páginas 29, 35, 45 e 60.

SHOKRANIAN, S. *Algebra 1*. 1. ed. [S.l.]: Ciência Moderna, 2010. Citado na página 27.

SHOUP, V. *A Computational Introduction to Number Theory and Algebra*. [S.l.]: Cambrigde University Press, 2005. Citado na página 26.

SILVERMAN, J. H. *The Arithmetic of Elliptic Curves*. 2. ed. [S.l.]: Springer, 2009. Citado 2 vezes nas páginas 57 e 59.

STALLINGS, W. *Cryptography and Network Security Principles and Pratice 5th Edition*. [S.l.]: Prentice Hall, 2011. ISBN 10: 0-13-609704-9, 13: 978-0-13-609704-4. Citado 6 vezes nas páginas 21, 23, 24, 29, 30 e 31.

TAFNER, E. P.; SILVA, R. Apostila de metodologia científica. 2007. Citado 2 vezes nas páginas 39 e 40.

TRAVASSOS, G. H. *Introdução à engenharia de software experimental*. UFRJ, 2002. (RT-ES-590/02). Disponível em: <http://www.ufpa.br/cdesouza/teaching/topes/4-ES-Experimental.pdf>. Citado 2 vezes nas páginas 39 e 40.

WANG, P.; ZHANG, F. An efficient collision detection method fo computing discrete logarithms with pollard's rho. School of Information Science and Technology, Sun Yat-sen University, Guangzhou 510006, China, 2011. Citado na página 48.

WOHLIN, C. *Experimentation in Software Engineering: An Introduction*. [S.l.]: Norwell, MA, USA: Kluwer Academic Publishers, 2000. ISBN 0-7923-8682-5. Citado na página 39.

Apêndices

APÊNDICE A – Otimizando a multiplicação de um ponto

A multiplicação de um ponto da curva elíptica é de extrema importância para a criptografia de curvas elípticas, pois, como foi visto, sua segurança depende da dificuldade em resolver o Problema do Logaritmo Discreto para Curvas Elípticas. Como a multiplicação de um ponto é definida em termos de sucessivas somas desse ponto com ele mesmo, então a multiplicação de um ponto se torna onerosa ao multiplicá-lo por um número muito grande. Para exemplificar, considere uma curva E/K e o ponto $P \in E(K)$, para calcular a multiplicação desse ponto por um número n seria necessário fazer uma soma recursiva para calcular os pontos

$$P, \quad [2]P = P + P, \quad [3]P = [2]P + P, \quad \ldots, \quad [n]P = [n-1]P + P$$

dessa forma, seria necessário realizar $n-1$ somas para encontrar o valor de $[n]P$. (SILVERMAN, 2009)

Um algoritmo mais eficiente para multiplicar um ponto $P \in E(K)$ por um número n é descrito abaixo (retirado de (SILVERMAN, 2009))

1. Escreva n como uma expansão binária

   $$n = \epsilon_0 + \epsilon_1 \cdot 2 + \epsilon_2 \cdot 2^2 + \cdots + \epsilon_t \cdot 2^t$$

   com $\epsilon_0, \ldots, \epsilon_t \in \{0, 1\}$ e $\epsilon_t = 1$.

2. Atribua $Q = P$ e $R = \begin{cases} O, & \text{se} \quad \epsilon_0 = 0 \\ P, & \text{se} \quad \epsilon_0 = 1 \end{cases}$

3. Repita para $i = 1, 2, \ldots, t$

   Atribua $Q = [2]Q$

   Se $\epsilon_i = 1$ então $R = R + Q$

4. Retornar R

APÊNDICE B – Algoritmo de Schoof

O algoritmo de Schoof é um algoritmo para calcular a ordem de uma curva elíptica em um tempo polinômial, que é dado por $O((\log n)^8)$ (SILVERMAN, 2009).

Considere uma curva elíptica definida sobre um corpo finito E/F_q e expressa pela equação $y^2 = x^3 + Ax + B$. Considere ainda que $q = p^n$ e que $p \neq 2, 3$. Pelo teorema de Hasse é dado que

$$\#E(F_q) = q + 1 - t, \qquad \text{com } |\mathrm{a}| \leq 2\sqrt{q}$$

e considere um conjunto de números primos $S = \{2, 3, 5, 7, \ldots, L\}$ tal que

$$\prod_{l \in S} l \geq 4\sqrt{q}$$

e a característica p do corpo finito não pertença a S. O algoritmo consiste em encontrar t mod l para cada primo $l \in S$, e enfim descobrir o valor de t utilizando o teorema do resto chinês.(ALVARADO, 2005)

B.0.0.1 Caso com $l = 2$

Para o caso em que $l = 2$, suponha que a equação $x^3 + Ax + B$ tenha uma raiz $\alpha \in \mathbb{F}_q$, então existe um ponto $(\alpha, 0) \in E(\mathbb{F}_q)$. A soma desse ponto com ele mesmo é o ponto no infinito $\mathcal{O}$, logo esse ponto possui ordem dois e dizemos que ele pertence ao grupo de pontos da curva elíptica com ordem dois, que é representado por $(\alpha, 0) \in E[2]$. O teorema de Lagrange diz que a ordem do subgrupo deve dividir a ordem do grupo, logo, se existe um subgrupo de ordem dois então a ordem do grupo será par, ou seja, $\#E(\mathbb{F}_q) = q + 1 - t \equiv 0 \pmod 2$, e por q ser ímpar, conclui-se que $t \equiv 0 \pmod 2$. (ALVARADO, 2005)

Dessa forma, se for provado que a equação $x^3 + Ax + B$ possui uma raíz em $\mathbb{F}_q$, então $t \equiv 0 \pmod 2$, caso contrário, se a raíz não existe, então $t \equiv 1 \pmod 2$.(MCGEE, 2006)

Uma forma verificar se a equação $x^3 + Ax + B$ possui uma raíz em $\mathbb{F}_q$ é fazendo o cálculo do seu máximo divisor comum com o polinômio $x^q - x$. Não será demonstrado nesse trabalho que o polinômio $x^q - x$ possui q raízes distintas em $\mathbb{F}_q$, logo calculando-se $\mathrm{mdc}(x^3 + Ax + B, x^q - x)$ será possível saber se a equação possui uma raiz em $\mathbb{F}_q$. Caso o mdc entre os polinômios seja igual a um, então a equação $x^3 + Ax + B$ não possui raiz em $\mathbb{F}_q$ e consequentemente $t \equiv 1 (\mathrm{mod}\ 2)$.(MCGEE, 2006)

B.0.0.2 Caso com $l > 2$

Para calcular t mod l para $l > 2$, é necessário fazer uso de uma matématica mais rebuscada. Para o propósito desse trabalho, serão dadas algumas definições e o algoritmo para realizar o calculo.

Dada uma curva elíptica E, o seu **j-invariante** é dado por

$$j = j(E) = 1728\frac{4A^3}{4A^3 + 27B^2} \tag{1}$$

onde $A^3 + 27B^2 \neq 0$. (SEET, 2007)

Outro conceito importante para o algoritmo de Schoof é o de polinômios de divisão. Tais polinômios são definidos sobre as coordenadas de um ponto da curva elíptica e seu resultado é zero para pontos de determinada ordem. Explicando melhor, define-se $E[n]$ como o conjunto dos pontos de E que possuem ordem n, a saber

$$E[n] = \{P \in E(\mathbb{F}_q) \quad | \quad nP = \mathcal{O}\}$$

Assim, o polinômio de divisão ψ_n de uma curva E possui a propriedade $\psi_n(x, y) = 0$ se e somente se o ponto com coordenadas $(x, y) \in E(\mathbb{F}_q$ possui ordem n, ou seja, $(x, y) \in E[n]$.(MCGEE, 2006)

Esses polinômios $\psi_m(x, y)$ podem ser obtidos recursivamente por

$$\begin{aligned}
&\psi_0 = 0\\
&\psi_1 = 1\\
&\psi_2 = 2y\\
&\psi_3 = 3x^4 + 6Ax^2 + 12Bx - A^2\\
&\psi_4 = 4y(x^6 + 5Ax^4 + 20Bx^3 - 5A^2x^2 - 4ABx - 8B^2 - A^3)\\
&\psi_{2m+1} = \psi_{m+2}\psi_m^3 - \psi_{m-1}\psi_{m+1}^3\\
&\psi_{2m} = \frac{\psi_m(\psi_{m+2}\psi_{m-1}^2 - \psi_{m-2}\psi_{m+1}^2)}{2y}
\end{aligned}$$

com $m > 2$. (SEET, 2007)

Agora, o algoritmo(SEET, 2007) para calcular t mod l para $l > 2$

1. Calcular $p_l \equiv p \bmod l$ com $|p_l| < l/2$.
2. Calcular a coordenada x' de

$$(x', y') = (x^{p^2}, y^{p^2}) + p_l(x, y) \bmod \psi_l$$

3. Para cada $h = 1, 2, \cdots, \frac{l-1}{2}$, calcular a coordenada x_h de $(x_h, y_h) = j(x, y)$

(a) Se $x' - x_h^p \equiv 0(\text{mod } \psi_l)$, ir para (b). Senão, tentar o próximo valor de h.

(b) Calcular y' e y_h. Caso

$$\frac{y' - y_h}{y} \equiv 0(\text{mod } \psi_l)$$

então $t \equiv h(\text{mod } l)$. Caso contrário, $t \equiv -h(\text{mod } l)$.

4. Caso o item (a) da etapa anterior não encontre valor de h que satisfaça a equivalência, então fazer $\omega^2 \equiv p(\text{mod } l)$. Caso ω não exista, então $t \equiv 0(\text{mod } l)$.

5. Se mdc(numerador$(x^p - x_\omega), \psi_l) = 1$, então $t \equiv 0(\text{mod } l)$. Senão, calcular

$$\text{mdc}(\text{numerador}(\frac{y^p - y_\omega}{y}), \psi_l)$$

Se mdc $\neq 1$, então $t \equiv 2\omega(\text{mod } l)$, caso contrário $a \equiv -2\omega(\text{mod } l)$.

Anexos

ANEXO A – Primeiro Anexo

Este anexo se refere à Seção 4.2.

O algoritmo recebe como parâmetro uma curva elíptica E e dois pontos P e Q pertencentes a E e calcula o valor de x tal que $x = \log_P Q$.

Código A.1 – Algoritmo Pollard-rho com único processador.

```
1  pollardRho_singleProcessor(E: Curva, P: Ponto, Q: Ponto): inteiro
2      seja L, k, inteiro
3      seja an, bn, inteiro
4      seja am, bm, inteiro
5      seja Xn, Xm, Ponto
6      seja c, d, array inteiro
7      seja R, array Ponto
8
9      k ← E.order()
10     para j ← 1, enquanto j ≤ L, faca
11         c[j] ← random() % k
12         d[j] ← random() % k
13         R[j] ← c[j]*P + d[j]*Q
14
15     an ← random() % k
16     bn ← random() % k
17     Xn ← an*P + bn*Q
18     am ← an
19     bm ← bn
20     Xm ← Xn
21
22     enquanto Xn != Xm faca
23         j ← H(Xn, L)
24         Xn ← Xn + R[j]
25         an ← an + c[j]
26         bn ← bn + b[j]
27
28     se bn = bm entao
29         retorne "falha"
30
31     seja x, inteiro
32     x ← (an - am)/(bm - bn) % k
33     retorne x
34
35 H(P: Ponto, L: inteiro): inteiro
36     retorne P.x % L + 1
```

ANEXO B – Segundo Anexo

Este anexo se refere à Seção 4.3.

O algoritmo recebe como parâmetro uma curva elíptica E e dois pontos P e Q pertencentes a E e calcula o valor de x tal que $x = \log_P Q$.

Código B.1 – Algoritmo Pollard-rho paralelizado.

```
1  pollardRho_parallelized(E: Curva, P: Ponto, Q: Ponto): inteiro
2      seja L, k, inteiro
3      seja an, bn, inteiro
4      seja am, bm, inteiro
5      seja Xn, Xm, Ponto
6      seja c, d, array inteiro
7      seja R, array Ponto
8
9      seja a, b, inteiro
10     seja X, ponto
11     seja Y, Point
12
13     k ← E.order()
14     para j ← 1, enquanto j ≤ L, faca
15         c[j] ← random() % k
16         d[j] ← random() % k
17         R[j] ← c[j]*P + d[j]*Q
18
19     # para cada processador M faca
20     a ← random() % k
21     b ← random() % k
22     X ← an*P + bn*Q
23
24     # repita ate que o servidor receba um ponto Y distinto pela 2a vez
25     enquanto duplicatedDistinguishedPointNotReceived() faca
26         se X = distinguishedPoint(X)
27             sendToServer(a, b, X)
28             j ← H(X)
29             X ← X + R[j]
30             a ← a + c[j] % k
31             b ← b + d[j] % k
32
33     an ← firstTripleCollided_a()
34     bn ← firstTripleCollided_b()
35     am ← secondTripleCollided_a()
36     bm ← secondTripleCollided_b()
37
```

```
38      se bn = bm entao
39          retorne "falha"
40
41      seja x, inteiro
42      x ← (an - am)/(bm - bn) % k
43      retorne x
44
45  H(P: Ponto, L: inteiro): inteiro
46      retorne P.x % L + 1
```