Trabalho de Conclusão de Curso

Uso de Curvas Elípticas para Assinatura Digital: ECDSA e EdDSA

Arthur Bernardo Lisboa Webler

JUIZ DE FORA

JULHO, 2019

Uso de Curvas Elípticas para Assinatura Digital: ECDSA e EdDSA

Arthur Bernardo Lisboa Webler

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientadora: Profa. Beatriz Casulari da Motta Ribeiro

JUIZ DE FORA

JULHO, 2019

Uso de Curvas Elípticas para Assinatura Digital: ECDSA e EdDSA

Arthur Bernardo Lisboa Webler

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Profa. Beatriz Casulari da Motta Ribeiro
Doutora em Matemática

Prof. Marcos de Mendonça Passini
Doutor em Engenharia de Sistemas e Computação

Prof. Frederico Sercio Feitosa
Doutor em Matemática

JUIZ DE FORA

2 DE JULHO, 2019

Resumo

Assinaturas digitais são uma tecnologia importante que ajuda muitos serviços da Internet a funcionar. Com uma assinatura digital, alguém produz prova de que é quem diz ser usando a área da matemática conhecida como Álgebra Abstrata. Em particular, assinaturas digitais são implementadas em grupos finitos comutativos. O Teorema de Lagrange e tópicos associados são necessários para entender grupos finitos e portanto sistemas atuais de assinatura digital.

O problema de resolver $g^x = h$ num grupo, quando g e h são conhecidos, é chamado de Problema do Logaritmo Discreto (DLP). A segurança de um esquema de assinatura digital depende da dificuldade de resolver o DLP no grupo usado. Esquemas mais antigos, como ElGamal e DSA, são baseados na dificuldade de resolver o DLP em grupos multiplicativos primos $\mathbb{F}_p^*$. O cálculo de índice é um algoritmo subexponencial para o DLP em $\mathbb{F}_p^*$. A ideia ligeiramente mais nova da curva elíptica como um grupo, com algoritmos semelhantes para assinaturas digitais, parece oferecer um DLP mais difícil devido à falta de um algoritmo subexponencial para o seu DLP. Portanto, ela parece ser uma alternativa mais segura.

Palavras-chave: Curvas Elípticas, Assinaturas Digitais, Problema do Logaritmo Discreto, Problema do Logaritmo Discreto da Curva Elíptica, Gestão de Identidade, DSA, ECDSA, EdDSA

Abstract

Digital signatures are an important technology that helps many Internet services to work. With a digital signature, someone produces proof that they are who they claim to be through mathematical means. Abstract Algebra is the field that has allowed their construction. Particularly, digital signatures are implemented in finite commutative groups. Lagrange's Theorem and associated topics are needed to understand finite groups and therefore current digital signature algorithms.

The problem of solving $g^x = h$ in a group, when g and h are known, is known as the Discrete Logarithm Problem (DLP). The security of a digital signature scheme depends on the difficulty of solving the DLP in the used group. Older schemes, such as ElGamal and DSA, are based on the DLP in prime multiplicative groups $\mathbb{F}_p^*$. The index calculus is a subexponential algorithm for the DLP in $\mathbb{F}_p^*$. The slightly younger idea of the elliptic curve as a group, with similar algorithms for digital signatures, seems to offer a harder DLP because of the lack of a subexponential algorithm for its DLP. Therefore, it seems to be a safer alternative.

Keywords: Elliptic Curves, Digital Signatures, Discrete Logarithm Problem, Elliptic Curve Discrete Logarithm Problem, Identity Management, DSA, ECDSA, EdDSA

There is always room at the top.

Daniel Webster

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
AD	Assinatura Digital
CE	Curva Elíptica
ECC	Criptografia de Curva Elíptica
DLP	Problema do Logaritmo Discreto
ECDLP	Problema do Logaritmo Discreto na Curva Elíptica
ECDSA	Algoritmo de Assinaturas Digitais em Curvas Elípticas
EdDSA	Algoritmo de Assinaturas Digitais na Curva de Edwards

1 Introdução

1.1 Assinaturas Digitais

Frodo olhou cuidadosamente o lacre antes de rompê-lo.

Certamente, a carta parecia ser de Gandalf.

J. R. R. Tolkien, *A Sociedade do Anel*

Assinatura digital (AD) é a técnica de produzir um objeto digital que apenas uma certa entidade poderia gerar, de modo a servir como **prova da sua identidade**. Esses objetos funcionam, como o nome sugere, como assinaturas no formato digital.

Cada usuário do mecanismo de assinaturas digitais tem duas chaves:

- sua **chave pública**, que é divulgada para outros; e
- sua **chave privada**, que guarda-se para si.

Digamos que Samantha tem tais chaves e quer enviar uma assinatura digital para Victor. Para produzir uma assinatura digital, Samantha terá que usar duas coisas:

- Um documento D;
- e sua chave privada.

Abstraindo os detalhes matemáticos por ora, ela então "assina"o documento D com sua chave privada: o resultado disso é justamente a assinatura digital S que ela deseja.

Depois de assinar, Samantha envia a assinatura para Victor. Para comprovar que foi de fato Samantha quem "assinou"o documento, Victor precisará igualmente de duas coisas:

- A assinatura digital S, criada por Samantha;
- e a chave pública de Samantha.

A saída desta verificação será o documento D original assinado por Samantha.

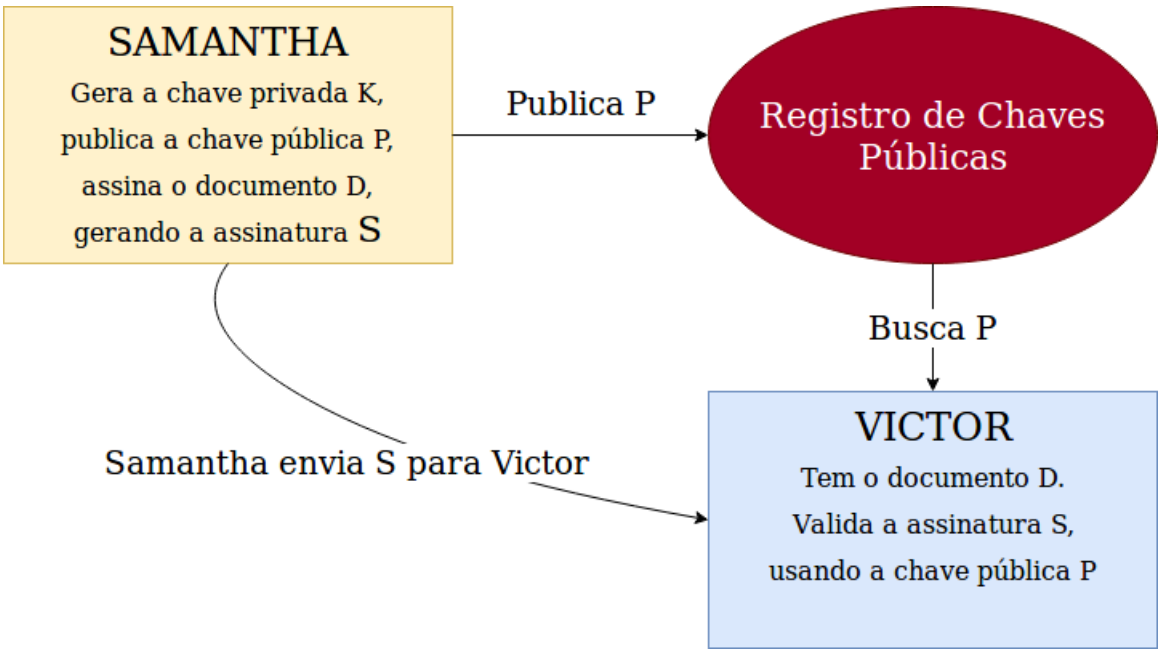

Figura 1.1: Esquema do funcionamento de assinaturas digitais.

1.2 Uso de Assinaturas Digitais

Assinaturas digitais são fundamentais para o funcionamento da Internet na atualidade, e particularmente do comércio virtual. É usando ADs que podemos confiar, por exemplo, num site que acessamos que diz ser `www.mercadolivre.com.br` (e não um atacante que está realizando um ataque *man-in-the-middle*). Na literatura, temos (Hoffstein_2008), p.439:

> Assinaturais digitais são pelo menos tão importantes quanto sistemas de criptografia de chave pública para a realização de comércio numa era digital, e de fato, *poderia-se argumentar que têm maior importância*. [grifo nosso.] [...] Como o seu computador consegue saber que uma atualização [do sistema] vem de uma fonte legítima? [...] A resposta é uma assinatura digital.

Além do uso na **autenticação de sites** na Internet, ADs também podem ser usadas:

- **Como opção a senhas** de caracteres em alguns protocolos (e sendo normalmente mais seguras que estas), como no protocolo SSH (1), fundamental para a administração de servidores Linux;

- **Como prova da posse de valores**, como é o caso nas criptomoedas (por exemplo, Bitcoin e Ethereum)

Além dos usos de comprovação da identidade vistos acima, um outro uso mais administrativo é o da **não-repudiação** (Ganley_1994): caso Samantha queira realizar uma ação no sistema de Victor, ela terá que prover uma AD juntamente com sua ação, de modo que ela não possa negar futuramente que realizou a ação.

Assinaturas digitais poderiam também ser usadas como prova da **integralidade dos dados**, mas no geral, funções de *hashing* são as preferidas para isso.

Em geral, quando falamos da Internet, as pessoas estão mais familiarizadas com o problema da criptografia: como transmitir mensagens de modo seguro, de modo que ninguém que as intercepte seja capaz de lê-las?

Em criptografias assimétricas, o remetente da mensagem a codifica de modo que só o destinatário possa decodificá-la. Numa assinatura digital, o assinante (que é o remetente) realiza um cálculo que só ele consegue fazer, mas que pode ser verificado por qualquer destinatário da assinatura. Então, do ponto de vista de implementação, podemos ver que existe uma analogia entre os dois problemas, e as mesmas técnicas de Matemática são usadas para solucionar ambos.

1.3 Assinaturas Digitais em Curvas Elípticas

O primeiro algoritmo de Assinatura Digital foi o Algoritmo RSA, publicado no mesmo artigo que propunha a mais famosa criptografia assimétrica de mesmo nome, em 1978 (Rivest_1978). A proposta de usar curvas elípticas para ADs veio depois, em 1985.

Este surgimento posterior teve um custo - a RSA já tinha sido mais estudada e se firmou como padrão na década de 1990. Mas as Assinaturas Digitas em Curvas Elípticas têm ganho relevância, particularmente pela suposta maior segurança e também pela sabida maior eficiência nos cálculos e no gasto de memória. Essa economia é importante para os dispositivos de pouca potência e dependentes de bateria da atual revolução da Computação Móvel.

O surgimento da moeda digital **Bitcoin**, na qual ADs em Curvas Elípticas ocupam o papel principal na posse dos valores, ajudou a difundir seu uso (sendo inclusive como o autor teve seu primeiro contato com a tecnologia). A grande maioria das criptomoedas posteriores também usam ADs em Curvas Elípticas para os mesmos fins, devido ao pequeno tamanho das chaves e das assinaturas, e à facilidade com que se geram novas chaves.

1.4 Objetivos

O objetivo deste principal deste trabalho é descrever a Matemática necessária por trás de um algoritmo de Assinatura Digital baseados em Curvas Elípticas - o Elliptic Curve Digital Signature Algorithm (ECDSA). De modo mais superficial, também veremos o Edwards Curve Digital Signature Algorithm (EdDSA).

Para entender seu funcionamento, será necessário passar por elementos da Matemática que os embasam - em especial, de Algebra Abstrata e Teoria dos Números, alguns dos primeiros algoritmos de Assinatura Digital e as próprias Curvas Elípticas. Veremos os assuntos nessa ordem.

1.5 Organização do Trabalho

O **capítulo 1**, que foi este, apresentou o trabalho.

O **capítulo 2** mostra os **conceitos matemáticos mais básicos**, tendo boa parte de seu conteúdo em comum com o curso de Teoria dos Números da UFJF.

O **capítulo 3** parte disso e vai além, mostrando **estruturas algébricas e teoremas mais avançados** que serão usados depois.

O **capítulo 4** debate o **Problema do Logaritmo Discreto em** $\mathbb{F}_p^*$, sobre a dificuldade do qual muitas cifras assimétricas se baseiam.

O **capítulo 5** mostra os esquemas de Assinatura Digital **Elgamal** e **DSA** em $\mathbb{F}_p^*$, primeiros a surgir e que servem de inspiração para os esquemas de Assinatura Digital que funcionam em Curvas Elípticas.

O **capítulo 6** introduz a noção matemática das **Curvas Elípticas**, e como po-

demos definir uma operação entre seus pontos para criar um grupo comutativo.

O **capítulo 7** debate o algoritmo de Assinatura Digital em Curvas Elípticas **ECDSA**, primeiro a ser introduzido como padrão, o **EdDSA**, alternativa mais recente, e o Problema do Logaritmo Discreto em Curvas Elípticas, análogo ao de $\mathbb{F}_p^*$.

O **capítulo 8** são as **considerações finais** do que vimos e onde exploramos possíveis trabalhos futuros nessa linha.

Os gráficos foram gerados pelo autor usando a biblioteca Python `matplotlib`.

2 Base matemática para Assinaturas Digitais

ἀγεωμέτρητος μηδεὶς εἰσίτω.

Não entre nenhum iletrado em geometria.

Suposta frase na entrada da Academia de Platão

Os principais protocolos criptográficos no mundo digital foram baseados em Álgebra Abstrata e Teoria dos Números. Para entender assinaturas digitais, é preciso entender alguns conceitos básicos destas áreas.

No geral, o que vamos mostrar aqui corresponde à matéria do curso de Teoria dos Números (chamado de *Matemática Discreta* em algumas universidades) da graduação em Ciência da Computação. Sugerimos maior atenção do leitor nos trechos deste capítulo e, especialmente, do próximo, que vão além - em especial, nas descrições das estruturas algébricas (grupos e corpos) e no Teorema de Lagrange e suas consequências.

2.1 Grupos Comutativos

Exemplo 2.1.1. (Adição em $\mathbb{Z}$) Pensemos na soma entre elementos de $\mathbb{Z}$: os inteiros. Podemos observar que:

- Sempre que somamos dois números a e b inteiros, o resultado é outro inteiro.
- Existe um número, o zero, que se somado a outro número a tem como resultado o próprio a; isso é, somar zero a um número qualquer não resulta em um número diferente.
- A ordem da soma não altera o resultado: $a + b = b + a$.
- Se queremos somar números a, b e c distintos, podemos somá-los em qualquer ordem e obter o mesmo resultado, isto é: $(a + b) + c = a + (b + c)$.
- Para todo número $a \in \mathbb{Z}$, existe um outro número $-a \in \mathbb{Z}$ tal que $a + (-a) = 0$.

Exemplo 2.1.2. (Multiplicação em $\mathbb{Q}^*$) Agora, vamos para a multiplicação no conjunto dos racionais sem o número zero: $\mathbb{Q}^*$. Afirmações parecidas podem ser feitas:

- Sempre que multiplicamos dois racionais a e b, o resultado é outro racional.
- Existe um número, 1, para o qual $1 * a = a$ sempre.
- A ordem dos números não altera o resultado: $a * b = b * a$.
- Se queremos multiplicar números a, b e c distintos, a ordem dos fatores não altera o resultado: $ab(c) = a(bc)$.
- Para todo número a, existe um outro número a^{-1} tal que $a * a^{-1} = 1$.

Podemos ver que existem importantes paralelos entre as duas operações; por exemplo, ambas têm um elemento, dito **neutro**, que quando operado com outro número não o altera; e que o elemento neutro pode ser obtido com uma operação a partir de qualquer elemento. Essas mesmas propriedades ocorrem em vários contextos na Matemática; portanto é útil abstrair o contexto e descrever essas propriedades por si só.

Dado um conjunto S e uma operação $\star$ em S, essas propriedades são válidas para todos a, b e c em C:

1. **Fechamento**: Se $a \star b = c$ então $c \in C$
2. **Elemento neutro**: Existe um elemento e tal que $e \star a = a$
3. **Comutatividade**: $a \star b = b \star a$
4. **Associatividade**: $(a \star b) \star c = a \star (b \star c)$
5. **Elemento inverso**: Existe a^{-1} em C tal que $a \star a^{-1} = a^{-1} \star a = e$

Então, com relação a $\star$, S é dito um grupo comutativo (ou abeliano).

Outras propriedades adicionais podem deduzidas apenas sabendo que se trata de um grupo comutativo, como veremos em seguida.

2.2 Divisibilidade

Nesse trabalho, divisibilidade é um conceito relacionado à multiplicação de inteiros. Dados números inteiros, dizemos que a divide b, e escrevemsos $a|b$, se existe um inteiro q tal que $a = bq$. Em casos em que isso não é verdade, teremos que $a = bq + r$, sendo que $r \neq 0$ e $0 < r < b$. Esse resultado é bastante conhecido, sendo denominado Divisão Euclidiana.

A divisibilidade tem propriedades interessantes, que podem ser deduzidas diretamente da definição.

- Transitividade: se $a \mid b$ e $b \mid c$, então $a \mid c$.
- Anti-simetria: se $a \mid b$ e $b \mid a$, então ou $a = b$ ou $a = -b$.
- Se $a \mid b$ e $a \mid c$, então $a \mid (b \pm c)$.

2.3 Algoritmo de Euclides

O **maior divisor comum** m entre dois inteiros a e b, comumente abreviado para mdc, é o maior inteiro capaz de dividir ambos a e b.

Por exemplo:

- O mdc entre 30 e 24 é 6.
- O mdc entre 24 e 12 é o próprio 12.
- O mdc entre 13 e 12 é 1.

Quando o mdc entre dois números for igual a 1, dizemos que eles são **relativamente primos**.

O algoritmo de Euclides é usado para encontrar o maior divisor comum entre quaisquer dois números. Ele realiza divisões sucessivas e mantém em sua memória dois valores de restos de divisões. Podemos descrevê-lo com o seguinte pseudocódigo:

Algoritmo 1: Algoritmo de Euclides para encontrar o MDC.

Entrada: $m, n \in \mathbb{Z}$

Saída: mdc(m, n)

```
1  início
2      se m < n então
3          troca (m , n)
4      fim se
5      repita
6          r = m % n
7          m = n
8          n = r
9      até n ≠ 0;
10 fim
```

O passo mais importante se dá na linha $r = m \,\%\, n$. Sabemos, por definição, que o mdc m entre a e b divide ambos. Logo, $m|(a - qb)$, sendo que $(a - qb)$ é o primeiro resto. Recursivamente, os novos restos são também gerados por contas do tipo $r_x - q'r_{x-1}$, de modo que todos eles são divisíveis pelo mdc entre a e b.

Como um resto sempre é menor que o divisor, os restos vão diminuindo até que um seja igual a zero (linha 9), que é divisível por todos os números. Quando isso acontece, o outro resto é igual ao mdc(a, b).

Notamos do algoritmo de Euclides que todos os restos gerados podem ser representados como somas de múltiplos de a e b. Isso não é difícil de ver: se r_1 é o primeiro resto, então $r_1 = a - bq$. Como o próximo será definido em função de r_1 e b, também poderá ser expresso em função de a e b, e assim por diante. A consequência disso é que mdc(a, b) pode ser escrito na forma

$$mdc(a, b) = x * a + y * b$$

em que x e y são inteiros.

2.4 Aritmética Modular

Pensemos em dias da semana. Sabemos que existem sete, ordenados assim:

Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira	Sábado	Domingo

Tabela 2.1: Dias da Semana

Se estamos, digamos, numa quinta-feira, e queremos saber qual dia da semana será daqui a 10 dias, é provável que pensemos assim: "Daqui a 7 dias, será outra quinta-feira. Então será uma sexta-feira, depois sábado, e finalmente domingo. Então daqui a 10 dias será domingo."

Já se quisermos saber que dia da semana será daqui a 20 dias, o raciocínio provavelmente será: "Daqui a 21 dias seria outra quinta-feira, porque 21 é igual a três vezes sete, ou seja, três semanas. Então, daqui a 20 dias será outra quarta-feira."

Esses raciocínios usam implicitamente **Aritmética Modular**. Considerando um número k inteiro, estamos dizendo que:

- Todo dia que está $7k + 3$ dias distante de hoje é um domingo.
- Todo dia que está $7k - 1$ dias distante de hoje é uma quarta-feira.

Podemos usar a **notação modular** para descrever divisões. Dizer que

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

significa que a gera o mesmo resto que b quando dividido por m. Daí, podemos concluir também que $a - b$ é um múltiplo de m, ou seja, $a - b = q * m$.

No nosso exemplo anterior, sabemos que existem 7 dias da semana. Se associarmos um número a cada dia da semana, sendo a domingo o 0, a segunda-feira o 1 e assim por diante, podemos dizer as mesmas coisas que dissemos antes usando Aritmética Modular e considerando "+4" porque estamos considerando que estamos numa quinta-feira:

$$7k + 3\,(+4) \equiv 0 \pmod{7}$$

$$7k - 1\,(+4) \equiv 3 \pmod{7}$$

A notação modular é útil porque podemos definir operações análogas às de $\mathbb{Z}$. Para todos a, b, c e $d \in \mathbb{Z}$, se $a \equiv b \pmod{m}$ e $c \equiv d \pmod{m}$, temos:

- Produto: $ac \equiv bd \pmod{m}$;
- Soma: $a + c \equiv b + d \pmod{m}$.

Exemplo 2.4.1. Sabendo que $8 \equiv 1 \pmod{7}$ e $9 \equiv 2 \pmod{7}$:

$$8 * 9 \equiv 72 \equiv 1 * 2 \equiv 2 \pmod{7}$$

$$8 + 9 \equiv 17 \equiv 1 + 2 \equiv 3 \pmod{7}.$$

Exemplo 2.4.2. Em alguns casos na aritmética modular, é necessário lembrar da definição. Por exemplo,

$$2 - 5 \equiv -3 \pmod{7}$$

Mas, $4 - (-3) = 7 = 1 * 7$. Portanto,

$$2 - 5 \equiv -3 \equiv 4 \pmod{7}$$

2.4.1 Conjunto $\mathbb{Z}/n\mathbb{Z}$

Observamos que existe uma quantidade finita de valores possíveis para restos numa divisão inteira. Mais especificamente, existem n valores possíveis para resto numa divisão por n: de 0 a $n - 1$. Grosso modo, podemos dizer que *mod* n é um conjunto *finito*. Além disso, todo número em $\mathbb{Z}$ corresponderá a um desses valores possíveis ao ser dividido por n.

Somando e multiplicando números no intervalo $[0, n - 1] \cap \mathbb{Z}$, obteremos novos números neste intervalo. Para esse conjunto especial de elementos finitos, determinados entre 0 e $n-1$, usamos a notação $\mathbb{Z}/n\mathbb{Z}$. Como vimos na seção anterior, esse conjunto admite uma soma e um produto, dando-lhe uma estrutura algébrica importante que veremos no próximo capítulo.

3 Resultados em $\mathbb{F}_p^*$

The Starks can count past six. Unlike some princes I might name.

Tyrion Lannister (to Joffrey), *A Game of Thrones*

Caso p seja um número primo, podemos afirmar fatos úteis sobre operações módulo p. Nesse capítulo, vamos ver alguns desses resultados.

3.1 Corpos

$\mathbb{Z}/p\mathbb{Z}$ tem certas propriedades que o caracterizam como uma estrutura algébrica chamado **corpo**, nos casos em que p é primo.

Chamamos de **corpo** todo conjunto $\mathbb{K}$ munido de duas operações tais que:

- Haja uma operação que chamamos **soma**, sob a qual $\mathbb{K}$ é um grupo comutativo;
- Haja uma operação que chamamos **multiplicação**, sob a qual $\mathbb{K} - \{0\}$ (isto é, $\mathbb{K}$ sem o elemento neutro da soma) é um grupo comutativo;
- A **propriedade distributiva** entre soma e multiplicação vigore: denotando a soma por $+$ e a multiplicação por $*$,

$$a * (b + c) = (a * b) + (a * c)$$

para a, b, c que pertençam ao conjunto.

Por exemplo, $\mathbb{Z}$, o conjunto dos números inteiros, não é um corpo porque nem todos os números tem um inverso na operação da multiplicação, o que é necessário para que ele forme um grupo comutativo e portanto para que o conjunto seja um corpo: não existe um número i em $\mathbb{Z}$ tal que $3 * i = 1$.

Já $\mathbb{Q}$, o conjunto dos números racionais, é um corpo, porque satisfaz as propriedades; observamos que se $\frac{p}{q} \in \mathbb{Q} - \{0\}$, então seu inverso multiplicativo é $\frac{q}{p}$.

Para mais um exemplo de corpo, lembremos que uma das consequências do Algoritmo de Euclides é que sempre podemos escrever o mdc entre a e b inteiros como

$$au + bv = \text{mdc}(a, b)$$

para u, v inteiros. Se $b = p$ e p é primo, para qualquer a que não seja múltiplo de p, teremos então

$$au + pv = 1 \Leftrightarrow au - 1 = -pv\,,$$

ou seja,

$$au \equiv 1 \pmod{p}\,.$$

Isso significa dizer que, se n é primo, todo inteiro a diferente de zero tem um inverso multiplicativo u em $\mathbb{Z}/n\mathbb{Z} - \{0\}$. Ao mesmo tempo, se n for composto, ao menos seu divisor $1 < d < n$ não terá inverso multiplicativo, pois se tivesse, $\text{mdc}(n, d) = 1$, o que é uma contradição. Com isso, temos o seguinte resultado:

Lema 3.1.1. *$\mathbb{Z}/n\mathbb{Z}$ é um corpo se e somente se n é primo. Nesse caso, denotamos $\mathbb{Z}/n\mathbb{Z}$ como $\mathbb{F}_n$.*

Notamos que $\mathbb{Q}$ é um corpo, e é um conjunto com infinitos elementos. Já $\mathbb{F}_p$, para p primo, é um corpo com elementos finitos. Neste caso, dizemos que é um **corpo finito**.

Em geral, vamos usar a notação $\mathbb{F}_p^*$ para nos referir ao **grupo (multiplicativo)** $1, 2, ..., p-1$, isto é, sem incluir o zero e que é portanto distinto do corpo finito $\mathbb{Z}/p\mathbb{Z} = \mathbb{F}_p$.

3.2 Ordem em grupos

Agora, vamos sair de corpos e voltar aos grupos. Chamamos de **ordem de um grupo finito** a quantidade de elementos que esse grupo tem. Nosso principal exemplo é o grupo $\mathbb{F}_p^*$, que tem $p - 1$ elementos. Já um grupo com infinitos elementos tem **ordem infinita**.

Chamamos de **ordem** w do elemento a de um grupo multiplicativo o *expoente mínimo* pelo qual temos que elevá-lo para obter o elemento neutro e, isto é, o menor inteiro não negativo tal que $a^w = e$. No caso de $\mathbb{F}_p^*$, isso significa que $a^w \equiv 1 \pmod{p}$.

Caso um expoente assim não exista, dizemos que o elemento tem **ordem infinita**.

O termo "ordem" está aqui submetido a uma sobrecarga de operador, significando coisas distintas quando aplicado a um corpo finito ou a elementos de um corpo qualquer. Como ambas estão relacionadas, isso geralmente não causará problema.

Dada a definição de ordem, vejamos agora alguns teoremas relacionados a ela que serão de grande utilidade posteriormente.

Teorema 3.2.1. *(Teorema da Ordem Finita) Se G é um grupo finito, então todo elemento a de G tem **ordem finita** $w \in \mathbb{N}$. Além disso, se $a^x = e$, então $w|x$.*

Demonstração. Como G tem ordem finita, se calcularmos $a^1, a^2, a^3, ...$, pelo Princípio das Casas de Pombos[1], devem existir i e j tais que

$$a^i = a^j \,.$$

Multiplicando em ambos os lados por a^{-i} (que existe pela deifinição de grupo), temos

$$a^{j-i} = e \,.$$

Portanto, existe um expoente para a que o leva ao elemento neutro.

Suponhamos agora que w seja o menor expoente assim, e que exista outro expoente x tal que $a^x = e$. Dividindo x por w, temos que existem únicos q e r inteiros tais que:

$$x = wq + r, \quad 0 \leq r < w$$

Logo

$$e = a^x = a^{wq+r} = a^{wq} * a^r = (a^q)^w * a^r = e^q * a^r = a^r$$

Ou seja, $a^r = e$. Mas assumimos que w era o menor expoente de a que leva ao elemento neutro, assim, como $0 \leq r < w$, devemos ter $r = 0$ e $x = wq$, portanto $w|x$. □

[1] O Princípio das Casas de Pombos afirma que, se temos c casas para colocar p pombos, se $c < p$, então inevitavelmente uma das casas terá mais de um pombo.

3.3 Teorema de Lagrange

O Teorema de Lagrange é um resultado essencial em Álgebra Abstrata. Podemos enunciá-lo como: se G é um grupo e H é um subgrupo de G, então $\#H|\#G$.

Aqui, $\#G$ representa a ordem (cardinalidade), ou seja, a quantidade de elementos, do grupo G. Para alcançar este resultado, precisaremos dos conceitos de **subgrupo** e **coclasse**.

Definição 3.3.1. Se um subconjunto H de G tem ele mesmo todas as propriedades de um grupo (ou seja, fechamento, elemento neutro, elemento inverso e associatividade), ele é dito um **subgrupo**.

A partir dessa definição, será útil salientar os seguintes fatos:

- O elemento neutro e pertence a todos os subgrupos.
- Todo $a \in H$ tem um inverso $a^{-1} \in H$ tal que $aa^{-1} = a^{-1}a = e$.
- Para quaisquer $a, b \in H$, $ab = c \in H$.

Um exemplo de subgrupo é, por exemplo, o subconjunto $\{0, 3, 6\}$ no grupo comutativo aditivo $\mathbb{Z}/9\mathbb{Z} = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$.

Definição 3.3.2. Dados um grupo G e um subgrupo H de G, uma **coclasse** à esquerda de H é gerada quando multiplicamos todos os elementos de H por um $a \in G$, isto é, $aH = \{ah|\ h \in H\}$. Já uma coclasse à direita é obtida analogamente por $Ha = \{ha|\ h \in H\}$. Para grupos comutativos, podemos falar de coclasses apenas, sem esta distinção.

Observando certas propriedades deste curioso conceito, seremos capazes de demonstrar o Teorema de Lagrange. Nos restringiremos a coclasses à esquerda, que serão suficientes para nossos fins. Uma elaboração maior sobre coclasses no geral pode ser vista em (Gallian_2003).

Lema 3.3.1. *Sejam G um grupo, H um subgrupo e $a, b \in G$. Então, são válidas as seguintes afirmações:*

1. $\#(aH) = \#(bH)\ \forall a, b \in G$.

2. $a \in aH$.

3. $aH = H \Leftrightarrow a \in H$.

4. $aH = bH \Leftrightarrow a \in bH$.

5. $aH = bH$ *ou* $aH \cap bH = \emptyset$.

Demonstração. Vamos provar cada item.

1. Podemos fazer a bijeção $ah \longmapsto bh$ para cada h, simplesmente. O fato de ser uma bijeção segue do fato de que os elementos de aH e bH são todos distintos dentro de cada conjunto uns dos outros, porque $ah_1 = ah_2$ implicaria $a^{-1}ah_1 = a^{-1}ah_2 \Rightarrow h_1 = h_2$, sendo que H só contém elementos distintos.

2. Como todo subgrupo contém o elemento neutro e, $ae = a \in aH$.

3. ($\Rightarrow$) $aH = H \Rightarrow ae \in H \Rightarrow a \in H$.
   ($\Leftarrow$) Se $a \in H$, temos que mostrar que $aH = H$. Para isso, é necessário mostrar que $aH \subseteq H$ e que $H \subset aH$.
   $aH \subseteq H$ vêm imediatamente pela propriedade do fechamento de subgrupos. Tomemos depois $h \in H$. Como $a^{-1} \in H$, $a^{-1}h \in H$. Logo, $a(a^{-1}h) \in aH \Rightarrow h \in aH$.
   Essa propriedade equivale a dizer que a coclasse à esquerda aH de um subgrupo é igual ao próprio subgrupo, desde que a pertença ao subgrupo.

4. ($\Rightarrow$) $a = ae \in aH = bH$.
   ($\Leftarrow$) Se $a \in bH$, $a = bh, h \in H$. Portanto, $aH = bhH$ - mas pelo item 3, $hH = H$. Logo, $aH = bH$.

5. Pelo item anterior, se houvesse c tal que $c \in aH$ e $c \in bH$, então $cH = aH$ e $cH = bH$ e portanto teríamos $aH = bH$.

□

Armados com estes resultados, podemos agora provar o Teorema de Lagrange.

Teorema 3.3.1. *(Lagrange) Se G é um grupo e H um subgrupo de G, então $\#H|\#G$.*

Demonstração. O item 1 do Lema 3.3.1 afirma que todas as coclasses de H têm a mesma quantidade de elementos. Já o item 2 afirma que todo elemento $a \in G$ aparecerá em alguma coclasse; e o item 5 afirma que as coclasses são disjuntas e particionam G. Portanto, podemos escrever G utilizando elementos $c_1, \ldots, c_n$ como

$$G = c_1H \cup c_2H ... \cup c_nH$$

Ou seja, $\#(G) = n(\#H)$ e portanto $\#H|\#G$. □

3.4 Teorema de Fermat

O Teorema de Fermat é fundamental em várias aplicações de Teoria dos Números. Seu enunciado é:

Teorema 3.4.1. *(Fermat) Dados $p, a \in \mathbb{Z}$, sendo p primo, se $p \nmid a$, então*

$$a^{p-1} \equiv 1 \pmod{p}.$$

Para prová-lo, além do Teorema de Lagrange, precisamos de mais um resultado.

Teorema 3.4.2. *Sejam G um grupo finito e a um elemento de G de ordem w. O conjunto $H = \{a, a^2, \ldots, a^w\}$ é um subgrupo de G.*

Demonstração. Sendo a um elemento em G, já sabemos que em algum momento teremos w tal que $a^w = e$. Mostremos que $\{a^1, a^2, ...a^w\}$ é um subgrupo.

1. Associatividade. A associatividade do grupo G não é alterada por nada em nosso conjunto e continua valendo.

2. Inverso. O inverso de qualquer a^x será evidentemente a^{w-x}.

3. Fechamento. $a^x a^y = a^{x+y} = a^{wq+r}$, sendo $0 \leq r < w$. Logo, $a^{wq+r} = e * a^r = a^r$, que pertence a nosso conjunto.

4. Elemento neutro. Continua havendo um elemento neutro, que é $a^w = e$.

□

Temos então o seguinte resultado:

Corolário 3.4.1. *A ordem de um elemento de um grupo finito divide a ordem (ou cardinalidade) do grupo.*

Estamos em condições de demonstrar o Teorema de Fermat (Teorema 3.4.1):

Demonstração. Segue do Teorema de Lagrange e do teorema 3.4.2. De fato, a ordem de qualquer elemento de $\mathbb{F}_p^*$ divide $p-1$, porque todas são também a ordem de um subgrupo H em $\mathbb{F}_p^*$. □

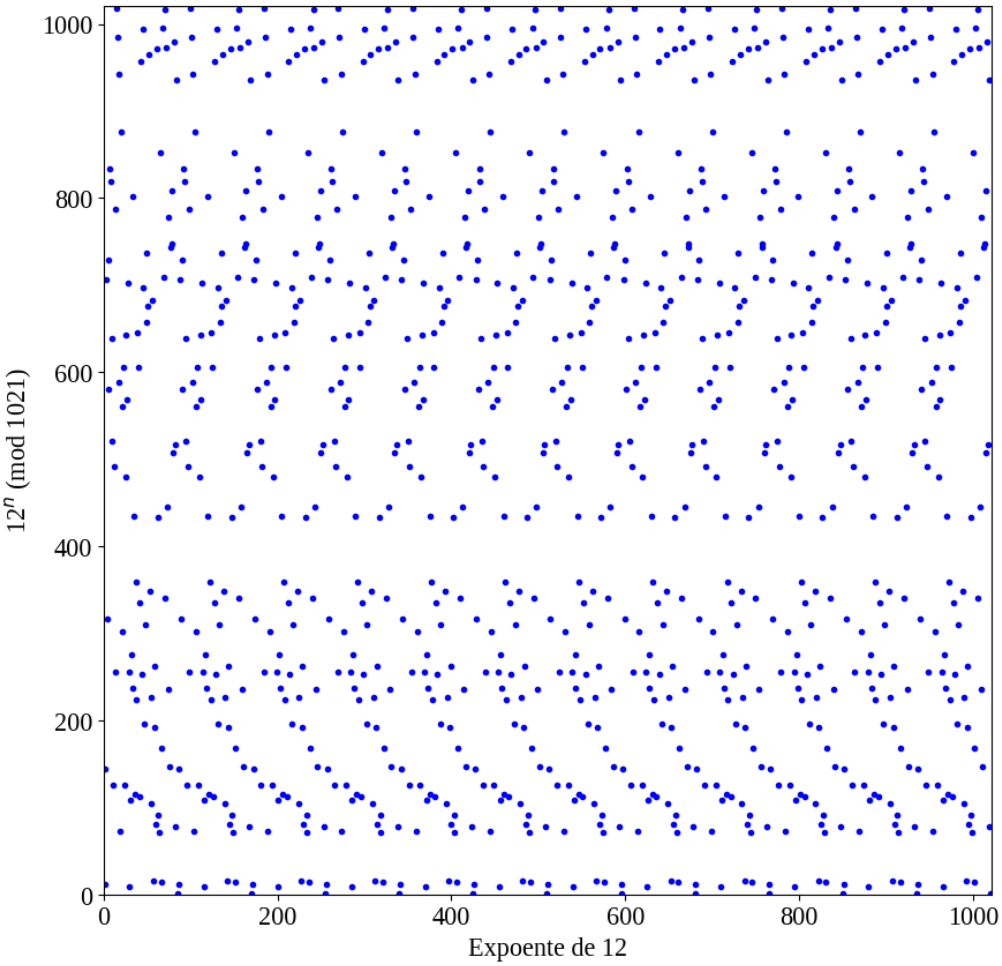

Figura 3.1: A ordem de 12 (primo) em F_{1021}^* é 85, um divisor de 1020. A repetição é evidente.

O Teorema de Fermat não exclui a possibilidade de a ordem de um elemento a ser menor que $p-1$ em $\mathbb{F}_p^*$, como de fato acontece às vezes - como podemos ver na figura 3.1 - como $12^{85} \equiv 1 \pmod{1021}$, os valores alcançados antes se repetem a partir de 12^{86}.

Outra consequência do Teorema de Lagrange é que, se a ordem de um elemento em $\mathbb{F}_p^*$ não é $p-1$, ela divide $p-1$ mesmo assim.

Por exemplo, em F_7^*, $2^3 \equiv 1 \pmod 7$ - portanto, a ordem de 2 em F_7^* é 3 e não $\#(F_7^*) = 6$; mas 3|6. Outro exemplo, em F_{11}^*, é $3^5 \equiv 243 \equiv 242 - 1 \equiv 2 * (11^2) - 1 \equiv 1 \pmod{11}$. A ordem de 3 em F_{11}^* é 5 e $\#(F_{11}^*) = 10$ e 5|10.

Definição 3.4.1. Elementos em G cuja ordem seja igual a $\#G$ são chamados de **raízes primitivas** de G.

Por exemplo, em F_7^*, 3 é uma raiz primitiva, pois sua ordem é 6. De fato:

$3^1 \equiv 3 \equiv 3 \pmod 7$ $\quad$ $3^2 \equiv 9 \equiv 2 \pmod 7$ $\quad$ $3^3 \equiv 6 \equiv 6 \pmod 7$

$3^4 \equiv 18 \equiv 4 \pmod 7$ $\quad$ $3^5 \equiv 12 \equiv 5 \pmod 7$ $\quad$ $3^6 \equiv 15 \equiv 1 \pmod 7$

Raízes primitivas são assim chamadas pois suas potências geram **todos** os elementos de G (se não o fizessem, teriam ordem menor que $p-1$). No caso acima, vemos que todos os elementos de $F_7^* = \{1, 2, 3, 4, 5, 6\}$, aparecem como resultados da exponenciação de 3, que é uma raiz primitiva.

4 Logaritmo Discreto em $\mathbb{F}_p^*$

> *[This paper] introduces a new digital signature scheme that depends on the difficulty of computing discrete logarithms over finite fields.*
>
> Taher ElGamal, (ElGamal_1985)

Podemos agora avançar para a prática. O problema do Logaritmo Discreto (*DLP*, na sigla em inglês) é de simples descrição. Queremos saber, dados um grupo $\mathbb{F}_p$, um elemento $h \neq 0$ desse grupo e um inteiro g que não seja múltiplo de p, se existe um inteiro positivo x tal que $g^x \equiv h \pmod{p}$.

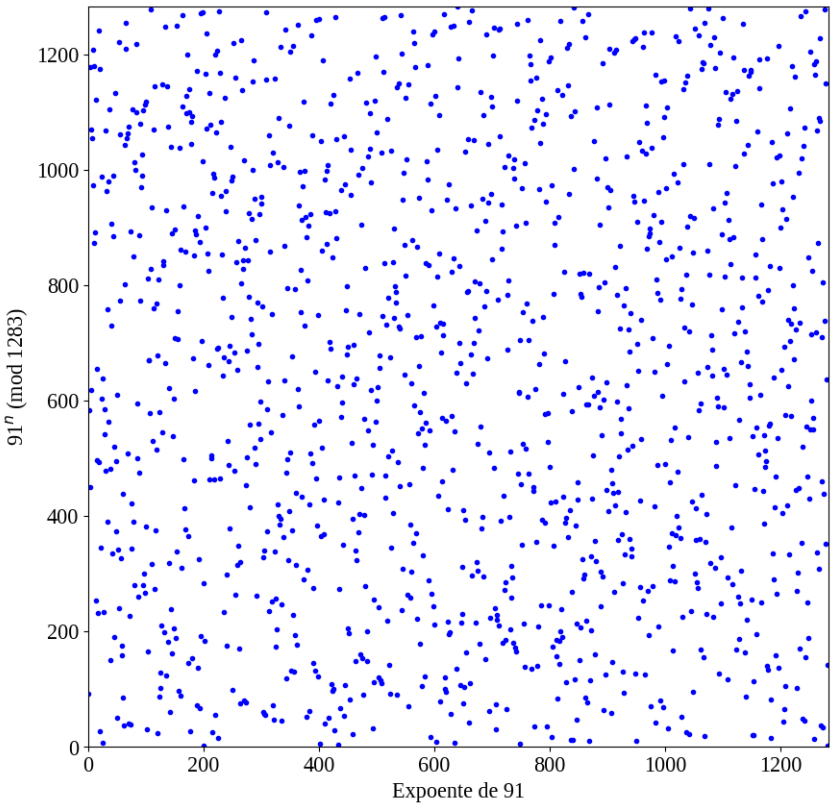

Figura 4.1: Potências $91 \pmod{1283}$.

No DLP, todos os números, exceto x, são conhecidos de quem deseja resolvê-lo. O cálculo de $g^x \pmod{p}$ variando x nos inteiros positivos gera resultados aparentemente aleatórios, como pode ser visto na figura 4.1. A dificuldade em resolver o problema vem justamente dessa aparente falta de estrutura.

A segurança de algumas criptografias assimétricas baseadas em Matemática Discreta e Teoria dos Números está asssociada à dificuldade de resolução do problema do logaritmo discreto em $\mathbb{F}_p^*$.

4.1 Ataque de Força Bruta

O método óbvio de resolução do DLP é começar da base e tentar todos os expoentes até $p-1$. Se todos os expoentes em $\{1, \ldots, p-1\}$ têm a mesma chance de serem a solução, o valor esperado de operações de multiplicação que teremos que fazer é de $\frac{1}{2}(p-1)$ (se considerarmos uma base g que seja uma raiz primitiva em F_p^*).

Dissemos acima que a complexidade do Algoritmo de Força Bruta tem uma relação linear com a ordem N da base g (sua complexidade é $O(N)$, ou $O(p)$, caso g seja uma raiz primitiva). Mas, como para cada bit `n` que o número primo p tiver, o dobro de expoentes terá que ser tentado, o total de testes que um ataque de força bruta deve fazer aumenta, em média, num fator de 2 por bit adicionado. Assim, é neste sentido que dizemos que o ataque de força bruta tem complexidade exponencial - a cada acréscimo de x bits na entrada, o problema se torna 2^x vezes mais caro de se resolver.

Números escolhidos para propósitos criptográficos têm, geralmente, centenas, se não milhares, de dígitos binários. Portanto, este método é impossível em um computador contemporâneo, que poderia levar bilhões de anos para atingir o expoente correto. Por exemplo, se o primo tivesse 1000 dígitos binários e o computador fosse capaz de testar 1.000.000 expoentes por segundo, levaria aproximadamente $1.7 * 10^{278}$ bilhões de anos (já que $2^{1000} \simeq 1.0715 * 10^{301}$, e 1 ano $\simeq 3.154 * 10^7$ segundos, e na média, ter-se-á que verificar metade dos expoentes possíveis).

4.2 Algoritmo de Colisão

O Algoritmo de Colisão de Shanks, em inglês também comumente chamado de **Baby-Step Giant-Step Algorithm**, é um algoritmo genérico, que resolve problemas do tipo $g^x \equiv h \pmod{p}$ em qualquer grupo, não só em $\mathbb{F}_p^*$. Isso é interessante, porque ele também pode ser aplicado ao Problema do Logaritmo Discreto numa Curva Elíptica

(ECDLP), que veremos futuramente. Sua complexidade é significativamente menor que a de um Ataque de Força Bruta, e é determinada pela ordem N do elemento g no grupo.

Nossa apresentação dele está baseada em (Hoffstein_2008). Podemos enunciar o Algoritmo de Colisão da seguinte forma:

Etapa 1

Faça $n = 1 + \lfloor \sqrt{N} \rfloor$, de modo que $n > \sqrt{N}$.

Etapa 2

Crie duas listas, a saber:

(Baby-Step) Lista 1: e, g, g^2, g^3, . . ., g^n

(Giant-Step) Lista 2: h, $h * g^{-n}$, $h * g^{-2n}$, $h * g^{-3n}$, . . ., $h * g^{-n^2}$

Etapa 3

Procure um elemento comum entre as duas listas, tal que $g^i = h * g^{-jn}$.

Etapa 4

Tendo i e j, calcule $x = i + jn$. Este x é uma solução para $g^x \equiv h \pmod{p}$.

Demonstração. Sabemos que $1 \leq x < N$, já que g tem ordem N. Podemos reescrever x como $x = nq + r$. Como n é conhecido e q e r de qualquer divisão são únicos, a colisão será única e se conseguirmos um modo de descobri-los, conseguimos também descobrir x.

Como $x < N$, temos que $nq < N$; e como $n > \sqrt{N}$, $q < n$. Igualmente $r < n$, pela definição de divisibilidade. Agora, podemos reescrever a equação $g^x \equiv h \pmod{p}$ como

$$g^{qn} * g^r \equiv h \pmod{p}$$

e, portanto,

$$g^r = h * g^{-qn}$$

que é a "colisão" que queremos alcançar. □

Ou seja, o algoritmo usa a técnica de divisão e conquista: tenta-se encontrar x de um modo indireto, reescrevendo-o como $x = nq + r$ e tentando então determinar q e r através de uma ideia astuta, que nos indica que duas funções geradas a partir de possíveis q e r terão que coincidir, mostrando assim seus valores apropriados.

4.2.1 Complexidade de Tempo e Espaço

Para fins práticos, a etapa 1 é de ordem $O(1)$. Do mesmo modo, a Etapa 4 é visivelmente $O(1)$, faltando então determinar o trecho mais complexo das etapas 2 e 3.

Etapa 2

Podemos observar que, como $r < n$ e $q < n$, que tanto a Lista 1, gerada calculando g^i para encontrar r, como a Lista 2, gerada calculando $h * g^{-jn}$ para encontrar q, têm n elementos. Como $n = \lfloor\sqrt{N}\rfloor$, fazemos aproximadamente $\sqrt{N}$ multiplicações para cada lista, dando uma ordem de complexidade de tempo (e de espaço) de $O(2\sqrt{N}) = O(\sqrt{N})$ para esta etapa.

Etapa 3

A etapa 3, de busca, é mais eficiente utilizando busca binária, que aqui levará $log(\sqrt{N})$ operações. Mas é necessário lembrar a natureza do Logaritmo Discreto, cujos valores gerados não ficam em ordem crescente como sua contraparte contínua, fato que podemos constatar ao reexaminar a figura 4.1. Por isso temos que ordenar cada lista antes de usar a busca binária, e como é sabido, os algoritmos de ordenação mais eficientes tem complexidade $O(n \log n)$. Assim, esta etapa terá complexidade $O(\sqrt{N} \log \sqrt{N})$. Esta etapa não aumenta a complexidade de espaço do algoritmo, visto que só usa dados que já foram criados.

4.2.2 Complexidade

Assim, temos que a complexidade final do Algoritmo de Colisão é

$$O(\sqrt{N}) + O(\sqrt{N} * \log \sqrt{N}) = O(\sqrt{N} * \log \sqrt{N})\,.$$

Por exemplo, se $N = 10^{100}$, um valor razoável para aplicações reais, a complexidade de tempo do Algoritmo de Colisão seria

$$O(\sqrt{10^{100}} * \log \sqrt{10^{100}}) = 10^{50} * 166 = 1,66 * 10^{52}\,.$$

O valor esperado para a complexidade do Algoritmo de Força Bruta seria $10^{100}/2 = 5{*}10^{99}$, que, estimando pela ordem de grandeza, levaria mais de 10^{47} vezes o tempo do Algoritmo de Colisão.

Mesmo assim, em nosso exemplo, $1,66 * 10^{52}$ operações está longe de ser um valor razoável para resolver o DLP. Sua complexidade de espaço enorme também chama a atenção, particularmente porque, na prática, a memória é o elemento mais lento do computador. A dependência de muitos acessos de leitura/escrita reduzem dramaticamente a performance, e a alternativa, que é paralelizar, tem inconvenientes próprios.

De qualquer forma, o Algoritmo de Colisão cumpre nosso objetivo de mostrar uma abordagem mais eficiente que o mero Ataque de Força Bruta, e tem a vantagem de funcionar em qualquer grupo.

4.3 Algoritmo Pohlig-Hellman

O algoritmo Pohlig-Hellman é um algoritmo ainda mais eficiente para a resolução do Problema do Logaritmo Discreto, explorando a estrutura do grupo. Seu conceito é de divisão e conquista: o DLP original é reduzido a outros DLPs de mais fácil resolução, que podem depois ser combinados para alcançar a resposta do original. Seu raciocínio não muito evidente à primeira vista exemplifica que o tamanho do grupo não é o único critério que devemos levar em conta ao escolhê-lo para aplicações de segurança.

Etapa 1

A partir da descrição do DLP original, $g^x \equiv h \pmod{p}$, a primeira etapa de Pohlig-Hellman é reescrever o número $(p-1)$ em sua fatoração de primos única:

$$(p-1) = q_1^{e1} * q_2^{e2} * \cdots * q_n^{e_n}$$

Etapa 2

Para cada q_i, geramos um novo DLP, da seguinte forma:

Calculamos $g_i = g^{N/q_i^{e_i}}$ e $h_i = h^{N/q_i^{e_i}}$. O novo DLP é

$$g_i^y \equiv h_i \pmod{p}$$

É útil observar que g_i **tem ordem** e_i, pois $g_i^{q_i^{e_i}} = g^{(N/q_i^{e_i})*q_i^{e_i}} = g^N = e$.

Algoritmos para bases de ordem q^e

A parte mais difícil do Algoritmo Pohlig-Hellman é esta etapa de resolução dos DLPs separados (que, em todo caso, mesmo juntos, são mais simples que o problema original). Poderíamos, é claro, resolvê-lo usando o Algoritmo de Colisão. Por outro lado, o algoritmo que veremos agora realiza outra simplificação e converte o problema numa base que tenha ordem q^e num outro que tem ordem de complexidade de $O(q)$.

Para que este algoritmo para o problema $g^x \equiv h \pmod{p}$ funcione, é necessário que g **tenha uma ordem que pode ser expressa como** $w = q^e, e \geq 1$.

A primeira etapa é usar o primo q, base única da fatoração da ordem de g, **como base numérica do expoente** x **de** g. Como g tem ordem q^e, $x < q^e$; e podemos reescrevê-lo como $x = x_0 + x_1 q + x_2 q^2 + \cdots + x_{e-1} q^{e-1}$, sendo que $0 \leq x_i < q$, pelo funcionamento das bases numéricas.

Feito isso, criaremos equações separadas, que permitem determinar cada um dos x_i por vez e finalmente o x original.

$$g^x \equiv g^{x_0 + x_1 q + x_2 q^2 + \cdots + x_{e-1} q^{e-1}} \equiv h \pmod{p}$$

$$g^{(x_0 + x_1 q + x_2 q^2 + \cdots + x_{e-1} q^{e-1}) * q^{e-1}} \equiv g^{x_0 q^{e-1}} * (g^{q^e})^{x_1 + x_2 q + \cdots + x_{e-1} q^{e-2}} \equiv h^{q^{e-1}} \pmod{p}$$

Como g tem ordem q^e, $g^{q^e} = 1$ e:

$$(g^{q^{e-1}})^{x_0} \equiv h^{q^{e-1}} \pmod{p}$$

Resolvemos este DLP em x_0, aqui finalmente com o Algoritmo de Colisão ou outro. Com

o valor dele em mãos, vamos elevar a equação anterior agora a q^{e-2}:

$$\begin{aligned}(g^{(x_0+x_1q+x_2q^2+\cdots+x_{e-1}q^{e-1})*q^{e-1}})^{q^{e-2}} &\equiv h^{q^{e-2}} \equiv g^{x_0q^{e-2}}g^{x_1q^{e-1}}(g^{q^e})^{x_2+x_3q+\cdots+x_{e-1}q^{e-3}} \\ &\equiv g^{x_0q^{e-2}}g^{x_1q^{e-1}} \pmod{p}\end{aligned}$$

Reescrevendo a equação, obtemos:

$$(g^{q^{e-1}})^{x_1} \equiv (h * g^{-x_0})^{q^{e-2}} \pmod{p}$$

Todos os valores, exceto x_1, são conhecidos. Novamente um DLP, e assim recursivamente até que todos os coeficientes x_i tenham sido determinados.

Com relação à complexidade do algoritmo, lembremos que g tem ordem q^e. Já $g^{q^{e-1}}$ tem ordem q, pois $(g^{q^{e-1}})^q = g^{q^{e-1}}g^q = g^{q^{e-1+1}} = g^{q^e} \equiv 1 \pmod{p}$.

De modo análogo, $g^{q^{e-i}}$ tem ordem iq, e então é linearmente, e não exponencialmente, relacionado ao primo q. Portanto, se usarmos o Algoritmo de Colisão no passo final, a ordem de complexidade deste algoritmo é $O(\sqrt{iq}) = O(\sqrt{i}\sqrt{q}) = O(\sqrt{q})$, e não $O(\sqrt{q^e})$ como seria se após montar os logaritmos discretos já usássemos imediatamente o Algoritmo de Colisão. No caso médio, isso constitui uma excelente melhora para a resolução do DLP.

Etapa 3

Agora que temos os valores de cada y_i para cada equação $g_i^{y_i} \equiv h_i \pmod{p}$ como vistas anteriormente, falta a etapa de reunir os valores encontrados. Fazemos isso resolvendo o sistema de congruências

$$x \equiv y_1 \pmod{q_1^{e_1}}, x \equiv y_2 \pmod{q_2^{e_2}}, \ldots x \equiv y_n \pmod{q_n^{e_n}}.$$

Este sistema é resolvido (por exemplo, com o Teorema do Resto Chinês) e então x é uma solução para o DLP original.

Teorema do Resto Chinês

O Teorema do Resto Chinês é um teorema e um método para resolver sistemas de equações em módulo. Seu enunciado é:

Teorema 4.3.1. *Teorema do Resto Chinês. Sejam $n_1, n_2, n_3, ...n_k$ inteiros dois a dois primos entre si. Então, um sistema de equações*

$$\begin{cases} x & = r_1 \pmod{n_1} \\ x & = r_2 \pmod{n_2} \\ & \dots \\ x & = r_k \pmod{n_k} \end{cases}$$

tem solução única $\pmod{N}$*, onde* $N = n_1 * n_2 * \cdots * n_i$*.*

Demonstração. Para cada n_i façamos N_i tal que $N_i = N/n_i$. Evidentemente, $mdc(N_i, n_i) = 1$. Portanto, como vimos, podemos escrever $1 \pmod{n_i}$ como $1 \equiv N_i * x_i \pmod{n_i}$, sendo x único em F_{n_i}.

Com isso, podemos escrever um x tal que

$$x = \sum_{i=1}^{k} N_i x_i r_i$$

Este x é uma solução para o sistema inicial, pois $\forall i$, $N_i x_i r_i \equiv r_i \pmod{n_i}$ e n_i divide todos os outros termos da soma.

Além disso, se houvesse uma segunda solução x', é evidente que $N|x - x'$. Portanto, $x' \equiv x \pmod{N}$, o que mostra que a solução é única $\pmod{n_1 n_2 \dots n_k}$. □

A prova do Teorema do Resto Chinês é construtiva, no sentido em que apresenta o método para resolver um sistema de equações em módulo em que os n_i são coprimos dois a dois. Primeiro resolvemos

$$x_1 \equiv r_1 \pmod{n_1}$$

(aqui, o simples $x_1 = r_1$ serve). Então vamos à próxima equação, que era $x_2 \equiv r_2$

(mod n_2). Agora queremos que ela satisfaça a equação anterior também, então temos essa limitação. Mas, se escrevermos o problema como

$$x_1 + n_1 * x_2 \equiv \quad (\text{mod } n_2),$$

a solução dessa equação, x_2 satisfará ambas as equações. Agindo assim recursivamente, por exemplo, com as próximas equações sendo

$$x_2 + (n_1 * n_2)x_3 \equiv \quad (\text{mod } n_3), \quad x_3 + (n_1 n_2 n_3)x_4 \equiv \quad (\text{mod } n_4)$$

até k, conseguiremos encontrar uma solução simultânea para todas as equações.

4.3.1 Complexidade

Geralmente, o custo do Teorema do Resto Chinês na Etapa 3 é desprezível, sendo o trecho mais caro a resolução dos logaritmos discretos na Etapa 2 (a fatoração de $(p-1)$ também pode ser custosa). Desse modo, sua ordem de complexidade é a mesma do algoritmo usado nessa etapa (como vimos, se usamos o Algoritmo de Colisão com a estratégia de reduzir a ordem dos grupos dos DLPs, o problema tem complexidade $O(q)$, sendo q o maior fator primo de p).

Pohlig-Hellman mostra que a fatoração de $(p-1)$ para ser usado em cifras que contam com a dificuldade do DLP é um fator de grande relevância para a escolha de $\mathbb{F}_p^*$. Quanto mais primos estiverem na fatoração e menores eles forem, mais eficiente é o algoritmo. Inevitavelmente, $(p-1)$ será par, portanto a melhor forma possível para um p contra Pohlig-Hellman seria $p = 2q + 1$, com q primo.

4.4 Logaritmo Discreto pelo Cálculo de Índice

Índice é um nome antigo para se referir a logaritmos discretos, que foi preservado no nome desse algoritmo. Antes de seguir com ele, precisamos do conceito de B-lisura.

Definição 4.4.1. Números B-lisos. Chamamos um número de **B-liso** quando ele pode ser completamente fatorado por números iguais ou menores que B.

Por exemplo: 144 é um número 3-liso, porque $144 = 12^2 = 2^4 3^2$; e 51 é um número 17-liso, porque $51 = 17^1 3^1$.

Etapa 1

Escolhemos um B tal que a probabilidade de um número n em $1 < n < p$ ser B-liso nos seja satisfatória. Temos aqui um *trade-off*: como veremos, gostaríamos de ter a maior probabilidade possível, mas teremos que calcular o logaritmo discreto de todos os primos menores que B. Foge ao escopo deste trabalho debater como realizar essa escolha, mas (Hoffstein_2008) introduz o assunto e indica outros trabalhos.

Etapa 2

Querendo novamente encontrar x em $g^x \equiv h \pmod{p}$, calculamos o logaritmo discreto de todos os primos iguais ou menores a B, da seguinte forma:

1. Escolhemos aleatoriamente um número i tal que $1 < i < p$, e calculamos $g^i \pmod{p}$.

2. Fatoramos $g^i \pmod{p}$ para verificar se ele é B-liso. Se não, voltamos ao passo 1.

3. Se g^i é B-liso, então podemos expressar o próprio i como uma combinação linear dos expoentes dos logaritmos discretos dos primos $\rho \leq B$. Portanto,

$$i \equiv \sum_{\rho \leq B} e_\rho * \log_g \rho \pmod{(p-1)}$$

   sendo u_ρ o expoente de cada primo. O sistema é $\pmod{(p-1)}$ porque estamos lidando com expoentes. Lembrando o Teorema de Fermat, fica fácil ver isso: $g^{(p-1)+b} \equiv g^{p-1} * g^b \equiv g^b \pmod{p}$.

4. Caso necessitemos de mais equações, voltamos para o passo 1. Caso contrário, resolvemos o sistema linear composto pelas equações geradas até agora e obtemos os logaritmos discretos dos primos $\rho \leq B$.

Etapa 3

Usando um contador k, vamos calculando $h * g^{-k} \pmod{p}$ para $k = 1, 2, 3, \ldots$. Para cada $h*g^{-k}$, testamos se ele é B-liso, ou seja, se ele pode ser fatorado como $\prod_{\rho \leq B} \rho^{e_\rho}$. Se sim, calculamos $x \equiv \log_g h \equiv k + \sum_{\rho \leq B}(e_\rho * \log_g \rho) \pmod{(p-1)}$.

4.4.1 Complexidade

Algumas coisas afetam a ordem de complexidade exata do Cálculo de Índice, como a escolha de B e o modo de se encontrar os números primos menores que B. Partindo de certas suposições, (Hoffstein_2008) diz que o trecho mais custoso teria complexidade $O(e^{\sqrt{(\ln p)*\ln(\ln p)}*\sqrt{2}})$, que equivale à ordem do total de i's que teriam que ser testados. (A forma da fórmula nos parece sugerir por que o livro não a demonstra).

Em todo caso, é importante notar que o Cálculo de Índice é um algoritmo sub-exponencial para o Problema do Logaritmo Discreto em $\mathbb{F}_p^*$, com menor complexidade de tempo e de espaço que o Algoritmo de Colisão, que é exponencial (mas funciona em qualquer grupo, não só em $\mathbb{F}_p^*$).

5 Esquemas de Assinatura Digital em $\mathbb{F}_p^*$

Veremos alguns esquemas de Assinatura Digital mais antigos, que são a inspiração dos esquemas de AD em curvas elípticas e tem análogos nelas: ElGamal, e o DSA (Digital Signature Algorithm).

5.1 Assinatura RSA

O primeiro esquema de AD foi introduzido juntamente com a criptografia assimétrica RSA, em 1978. Ele é muito distinto das ADs com curvas elípticas e não ajuda a entendê-las; por isso não vamos examiná-lo aqui. Além disso, ele enfrenta críticas de ordem técnica:

> A RSA oferece verificação de assinaturas ainda mais rápida que a ECC. Mas a RSA demora muito mais para assinar, gerar chaves, e tem assinaturas e chaves públicas muito maiores. É difícil encontrar aplicações onde isso é uma boa troca, e é fácil encontrar aplicações onde a performance ruim da RSA comprometeu a segurança. A RSA também tem muitas armadilhas de implementação, e mesmo uma implementação da RSA muito em dia é mais preocupante do ponto de vista de segurança do que a ECC[...].(2)

5.2 Assinaturas Digitais ElGamal

O esquema de assinatura digital ElGamal foi concebido em 1984 e publicado em 1985 pelo egípcio Taher ElGamal, no mesmo artigo em que apresentou seu esquema de criptografia de chave pública (ElGamal_1985). Ambos usam as mesmas idéias básicas e se apoiam na dificuldade de resolução do Problema do Logaritmo Discreto em F_p^*.

Nela, haverá um primo p que define o corpo finito $\mathbb{F}_p$ e uma base g (geralmente uma raiz primitiva em $\mathbb{F}_p^*$).

Samantha, que deseja assinar um documento, escolhe um número secreto c entre

0 e $p-1$. c é efetivamente sua chave privada, que ela deve guardar para si. A partir dela, Samantha calcula sua chave pública C:

$$C \equiv g^c \pmod{p}$$

Feito isso, Samantha assinará o documento D, conhecido de ambos. Para que as contas funcionem, D tem que ser representado como um número entre 2 e $p-1$, inclusive (por exemplo, X bits de uma determinada função de hashing aplicada sobre o documento).

Além disso, Samantha precisa escolher um valor k aleatório relativamente primo a $p-1$, isto é, tal que $\text{mdc}(k, p-1) = 1$. Este valor é chamado de **nonce**, e só deve usado uma única vez com a mesma chave.

Depois disso, Samantha calcula dois valores. Esses dois valores juntos constituem a assinatura.

O primeiro é

$$S_1 \equiv g^k \pmod{p}$$

Ele serve para adicionar um elemento aleatório nos cálculos que não é revelado: sua remoção terá que passar forçosamente pelos cálculos de processo de verificação.

Para calcular o segundo número da assinatura, Samantha primeiro tira o elemento i, o inverso multiplicativo de k em $\mathbb{F}_{p-1}$.

$$ik \equiv 1 \pmod{p-1}$$

Com i, a segunda parte da assinatura pode ser calculada.

$$S_2 \equiv (D - cS_1)i \pmod{p-1}$$

Samantha então assume que sua chave pública C e o documento D são de conhecimento público e publica o par (S_1, S_2) como sua assinatura.

5.2.1 Verificando a assinatura

Para verificar a assinatura de Samantha, Victor calculará $A^{S_1}S_1^{S_2}$. Se

$$A^{S_1}S_1^{S_2} \equiv g^D \pmod{p},$$

então a assinatura é legítima, o que provaremos em seguida.

Cabe agora provar por que a verificação funciona. Primeiramente, faremos duas observações, usando o Teorema de Fermat.

Lema 5.2.1. *Se* $a \equiv b \pmod{p-1}$*, então* $x^a \equiv x^b \pmod{p}$*.*

Demonstração. Já que $a \equiv b \pmod{p-1}$, então $a = (p-1)q + b$ para algum inteiro q. Logo,

$$x^a \equiv x^{(p-1)q+b} \equiv x^b \pmod{p},$$

pelo Teorema de Fermat. □

Lema 5.2.2. *Se* $ab \equiv 1 \pmod{p-1}$*, então* $x^{abc} \equiv x^c \pmod{p}$*.*

Demonstração. $ab \equiv 1\, mod \pmod{p-1}$, logo $ab = (p-1)q + 1$ para algum $q \in \mathbb{Z}$. Daí,

$$x^{abc} \equiv x^{((p-1)q+1)c} \equiv x^{(p-1)qc}x^c \pmod{p}.$$

Mas, pelo Teorema de Fermat, $x^{(p-1)qc} \equiv 1^{qc} \equiv 1 \pmod{p}$, logo

$$x^{abc} \equiv x^{(p-1)q+c}x^c \equiv x^c \pmod{p}.$$

□

Com estes resultados em mãos, podemos provar o funcionamento da AD ElGamal. Lembrando, queremos provar que

$$A^{S_1}S_1^{S_2} \equiv g^D \pmod{p}.$$

Substituindo as definições de S_1 e S_2 na equação aos poucos, temos que

$$A^{S_1} * S_1^{S_2} \equiv g^{cS_1} * g^{kS_2} \equiv g^{cS_1+kS_2} \equiv g^{cS_1+k(D-cS_1)i} \pmod{p},$$

o que podemos fazer graças ao lema 5.2.1, e

$$g^{cS_1+k(D-cS_1)i} \equiv g^{cS_1+(D-cS_1)} \equiv g^D \pmod{p},$$

graças ao lema 5.2.2, que é o que queríamos mostrar.

5.3 DSA

5.3.1 Introdução

O esquema de AD DSA, do inglês *Digital Signature Algorithm*, é essencialmente o mesmo de ElGamal, mas funcionando em um subgrupo de $\mathbb{F}_p^*$. Foi definido em setembro de 1994 pela National Security Agency (NSA) dos Estados Unidos (3). Sua vantagem sobre o sistema ElGamal é ter uma assinatura muito menor em termos de espaço, tendo aproximadamente o mesmo nível de segurança. Ele é a inspiração direta do esquema ECDSA de Assinatura Digital em Curvas Elípticas, que também foi definido pela NSA.

Sua diferença fundamental está em usar um subgrupo de $\mathbb{F}_p^*$, ao invés de $\mathbb{F}_p^*$ inteiro. Como vimos nos algoritmos de resolução do Problema do Logaritmo Discreto, a segurança de cifras baseadas nele está intimamente ligada com o maior fator primo de $p-1$ - ou ao menos para os algoritmos que já foram desenvolvidos até agora. Por isso, o DSA seleciona um fator primo q de $p-1$ e um elemento que tenha ordem q em $\mathbb{F}_p^*$. Na prática, valores típicos para p e q parecem ser $2^{1000} < p < 2^{2000}$ e $2^{160} < q < 2^{320}$ (vide (Hoffstein_2008)).

Ao longo da assinatura e verificação, calculando como na AD ElGamal, em alguns o DSA também uma operação $\pmod{q}$. De certa forma, o que ele faz é compactar $\mathbb{F}_p^*$ em $\mathbb{F}_q^*$ com uma função $F : \mathbb{F}_p^* \to \mathbb{F}_q^*$. Relembrando o Teorema de Lagrange, $q|(p-1)$ e portanto $qm = (p-1)$; cada ponto em $\mathbb{F}_q^*$ será a imagem de m pontos em $\mathbb{F}_p^*$.

5.3.2 Parâmetros do DSA

Como na AD ElGamal, temos que escolher um primo p. Além disso, escolhemos:

- Um primo q que é um fator de $p - 1$;
- **Uma base g que tenha ordem q em $\mathbb{F}_p^*$.** Este detalhe é necessário para que as contas funcionem. Isso difere de ElGamal, em que uma raiz primitiva em $\mathbb{F}_p^*$ provavelmente seria a melhor escolha.

Estes três números, p, q e g compõem os parâmetros públicos do DSA.

5.3.3 Assinatura no DSA

Samantha escolherá um expoente a novamente, que guardará para si, e calculará sua chave pública a partir dele:

$$A \equiv g^a \pmod{p}$$

- Chave privada: a, $1 < a < p$
- Chave pública: $A \equiv g^a \pmod{p}$

Para criptografar uma mensagem, ela precisa escolher novamente um nonce k, $1 < k < q$. Para que as contas funcionem, k precisa ter um inverso multiplicativo em $\mathbb{F}_q^*$, tal qual precisava também com relação a $p - 1$ em ElGamal; mas como q é primo, qualquer inteiro no intervalo $[2, q - 1]$ satisfaz essa condição.

O documento D a ser assinado precisa também ser representado por um inteiro no intervalo $1 \leq D < q$.

Depois disso, Samantha calcula S_1 e S_2 como a seguir:

$$S_1 \equiv (g^k \pmod{p}) \pmod{q}$$

$$S_2 \equiv (D + aS_1)k^{-1} \pmod{q}$$

O par (S_1, S_2) é sua assinatura.

5.3.4 Verificação e prova

Para verificar a assinatura, Victor calcula primeiro

$$V_1 \equiv DS_2^{-1} \pmod{q} \quad \text{e} \quad V_2 \equiv S_1 S_2^{-1} \pmod{q}$$

e munido destes valores, verifica que

$$(g^{V_1} A^{V_2} \pmod{p}) \pmod{q} \equiv (g^k \pmod{p}) \pmod{q} \equiv S_1 \pmod{q}$$

Antes de verificar esta conta, precisamos do seguinte resultado.

Lema 5.3.1. *Dados:*

- *um primo p;*
- *um primo q que divide $p-1$;*
- *um número g de ordem q em $\mathbb{F}_p^*$*

Se $a \equiv b \pmod{q}$, então $g^a \equiv g^b \pmod{p}$.

Demonstração. Podemos reescrever a como $a = nq + b$. Logo,

$$g^a \equiv g^{nq+b} \equiv g^1 gb \equiv g^b \pmod{p},$$

pelo Teorema de Lagrange, já que a ordem de g em $\mathbb{F}_p^*$ é q. □

Este resultado nos dá grande liberdade no DSA quando trabalhamos com a base g em $\mathbb{F}_p^*$: ele nos garante que reduzir um expoente e de g com uma operação $\pmod{q}$ não altera o valor de g^e.

Retomando a demonstração de que $(g^{V_1} A^{V_2} \pmod{p}) \pmod{q} = S_1$, substituamos V_1, V_2 e A por suas definições na equação:

$$g^{V_1} A^{V_2} \equiv g^{DS_2^{-1}} g a S_1 S_2^{-1} \pmod{p}$$
$$g^{DS_2^{-1}} g^{aS_1 S_2^{-1}} \equiv g^{(D+aS_1)S_2^{-1}} \pmod{p}$$

Como $S_2 \equiv (D + aS_1)k^{-1} \pmod{q}$,

$$g^{(D+aS_1)S_2} \equiv g^{(D+aS_1)(D+aS_1)^{-1}k} \pmod{p}$$

E pela observação acima, $g^{(D+aS_1)(D+aS_1)^{-1}} \equiv g^1 \pmod{p}$, logo

$$g^{(D+aS_1)(D+aS_1)^{-1}}k \equiv g^{1*k} \equiv g^k \pmod{p}$$

e

$$(g^{V_1}A^{V_2} \pmod{p}) \pmod{q} = (g^k \pmod{p}) \pmod{q} = S_1,$$

que é o que queríamos mostrar.

6 Curvas Elípticas

Elliptic curves have been objects of intense study in Number Theory for the last 90 years.

Victor S. Miller (em 1985), (Miller_1985)

Curvas elípticas são uma área de estudo ativo na Matemática. Foram usadas, por exemplo, na prova do Último Teorema de Fermat por Andrew Wiles em 1993. Seu potencial para a criptografia foi percebido pela primeira vez independentemente em 1985 por Neil Koblitz (Koblitz_1987) e Victor S. Miller (Miller_1985). Apesar do nome, curvas elípticas não estão relacionadas diretamente com elipses, exceto pelo fato que elas surgem no cálculo das áreas de elipses; o nome vem das funções elípticas.

Nesse trabalho, consideramos curvas elípticas com a definição na forma de Weierstrass: dado um corpo $\mathbb{K}$, uma curva elíptica é o conjunto de pontos $(x, y) \in \mathbb{K}^2$ satisfazendo $y^2 = x^3 + ax + b$, sendo que a, b estão em $\mathbb{K}$ e $4a^3 + 27b^2 \neq 0$.

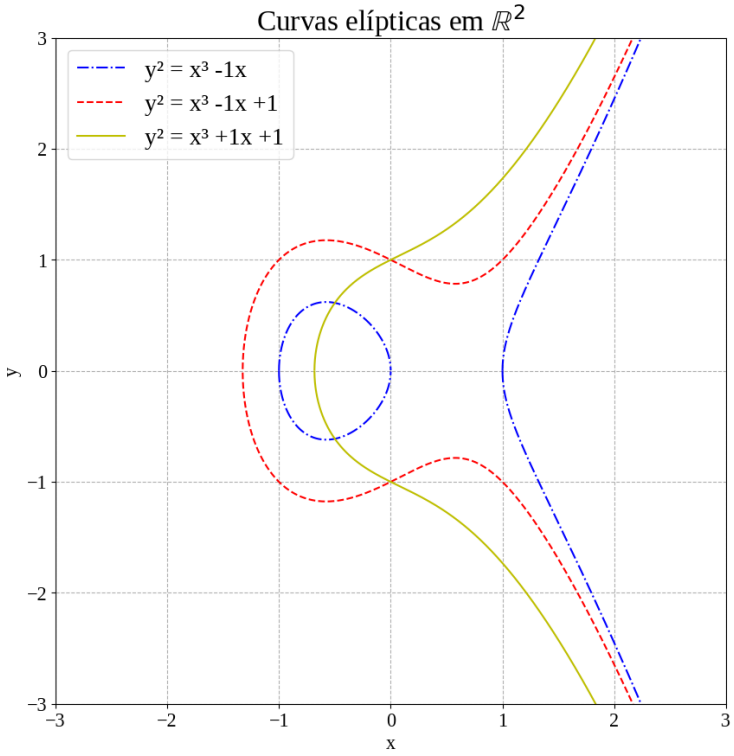

Figura 6.1: Exemplos de curvas elípticas no $\mathbb{R}^2$.

6.1 Adição na Curva Elíptica

Nós vamos tirar proveito de uma propriedade de curvas cúbicas: traçando uma reta por dois pontos da curva, a reta sempre encontrará um e somente um outro ponto da curva. Podemos usar essa propriedade para definir uma **operação de soma de pontos** na curva elíptica, que caracterizará um grupo comutativo como visto no Capítulo 2.

Geometricamente, a operação básica pode ser descrita da seguinte forma:

Definição 6.1.1. (Soma de pontos.) Dados os pontos P e Q que pertencem à curva elíptica C, traçamos uma reta. Devido à propriedade acima, haverá um ponto R com coordenadas (x, y) que pertencerá à reta $\overline{PQ}$ e à curva C. O ponto R', de coordenadas $(x, -y)$ é o resultado da soma entre os pontos P e Q.

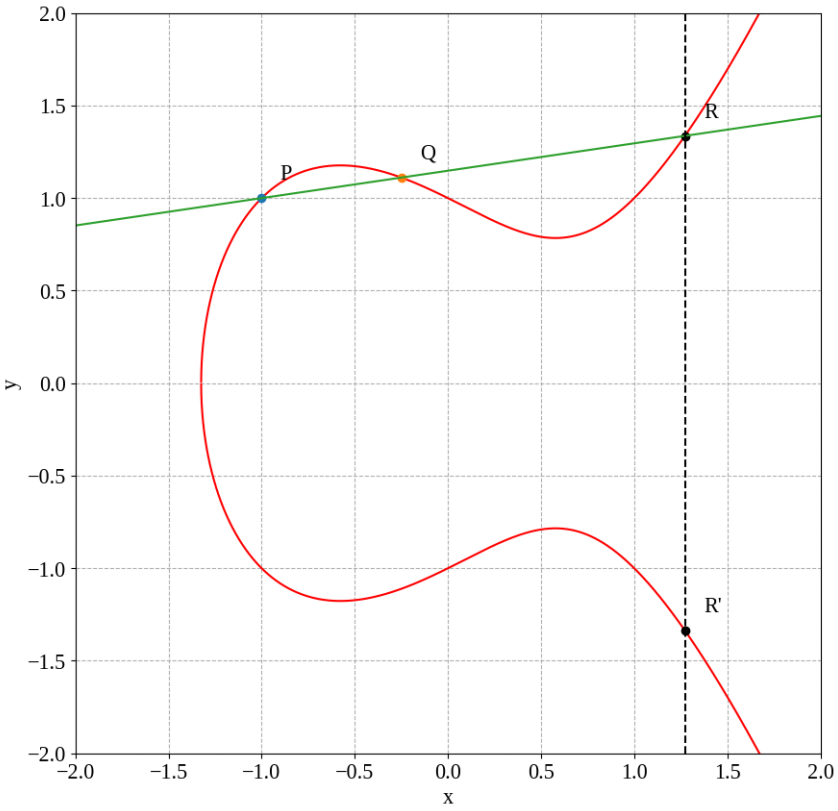

Figura 6.2: Visão geométrica da soma de pontos numa curva elíptica.

Observamos que, como toda curva elíptica na forma de Weierstrass é evidentemente simétrica em relação ao eixo x, o ponto $(x, -y)$ também pertencerá à curva.

6.1.1 Casos especiais

O resultado acima não é suficiente para estabelecer um grupo comutativo como queremos. Temos ainda que responder às perguntas:

1. Como somamos um ponto a ele mesmo?
2. Quem é o elemento neutro?
3. Quem é o inverso de cada elemento?

Somando um ponto a si mesmo

Nos remetemos à ideia geométrica da definição de derivada. Observamos que se a distância entre P e Q vai se tornando cada vez menor, o resultado da soma se aproxima cada vez mais do que seria somar o próprio P duas vezes. Por isso, definimos a soma de um ponto consigo mesmo como **o ponto que é atingido pela reta tangente à curva no ponto** P. A figura 6.3 mostra a soma de um ponto a si mesmo.

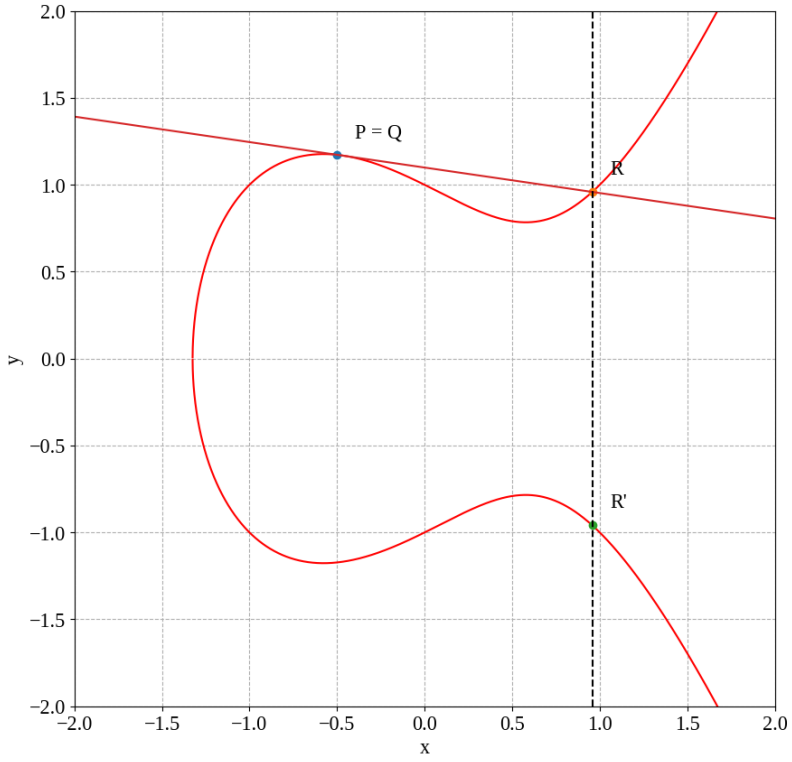

Figura 6.3: Somando um ponto a si mesmo.

Elemento neutro

A definição do elemento neutro na curva elíptica requer um pouco mais de imaginação: o ponto O, que é o elemento neutro, é definido como o ponto "no infinito", que é a interseção de todas as retas paralelas ao eixo y.

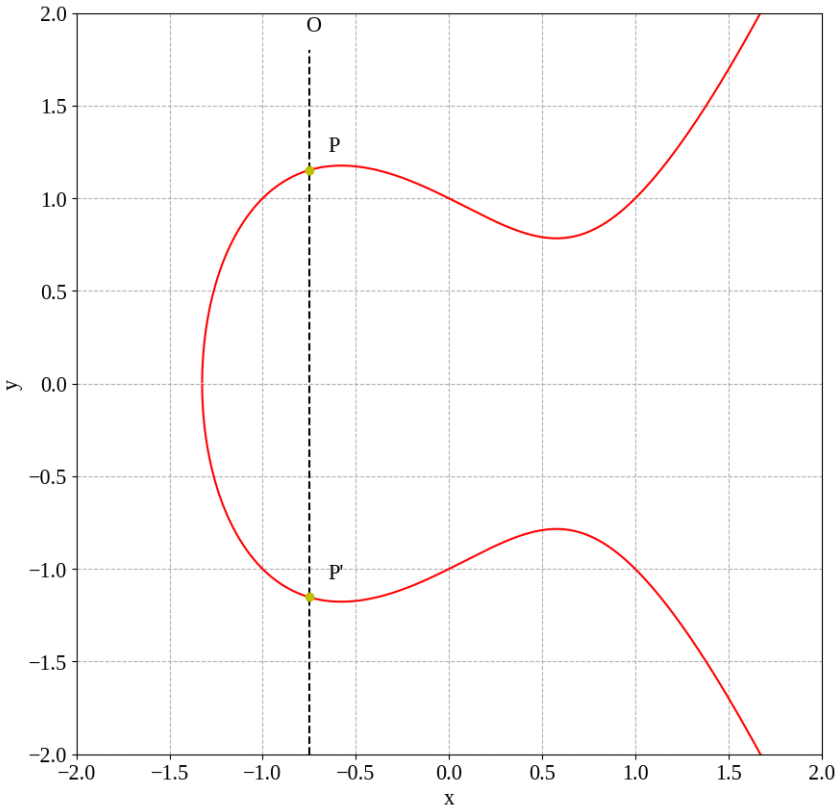

Figura 6.4: Elemento neutro na soma de pontos na curva elíptica.

Com essa definição, podemos completar os requisitos para dizer que a soma de pontos na curva elíptica constitui um grupo comutativo:

- Pela propriedade curvas cúbicas e pela definição da soma, o resultado da soma será um ponto que pertence à curva (fechamento);
- A comutatividade também é evidente, pois $\overline{PQ} = \overline{QP}$.
- Foge ao escopo do que estudamos a prova da associatividade, que está em (Silverman_2015).
- Se somamos um ponto P a O, a reta entre ambos atinge o ponto P', e após realizar a troca do sinal da coordenada y para atingir o resultado da soma, voltamos ao ponto P original para todo P. Portanto, O é o elemento neutro da soma.

- Se somamos um ponto P ao seu espelhado P', a reta atinge o ponto O - portanto, todo ponto passa a ter também um elemento inverso.

6.1.2 Polinômios com raízes repetidas

Trabalhar com polinômios de grau 3 têm certas sutilezas. Este parágrafo serve para exemplificar uma delas, que apareceu ao autor enquanto este estava experimentando com a operação da soma.

Somando geometricamente $(-1, -1)$ com $(1, 1)$ na curva $y^2 = x^3 - x + 1$, a reta $\overline{PQ}$ é evidentemente $x = y$. Substituindo na equação da curva, obtemos $x^3 - x^2 - x + 1 = 0$, isto é, $(x - 1)(x - 1)(x + 1) = 0$, ou seja, duas das raízes são -1 e 1. Assim, o terceiro x, que seria o x de R, é o próprio -1. Portanto, $R = P$. Em certo sentido, a reta passa três vezes pela curva: duas em P e uma em Q.

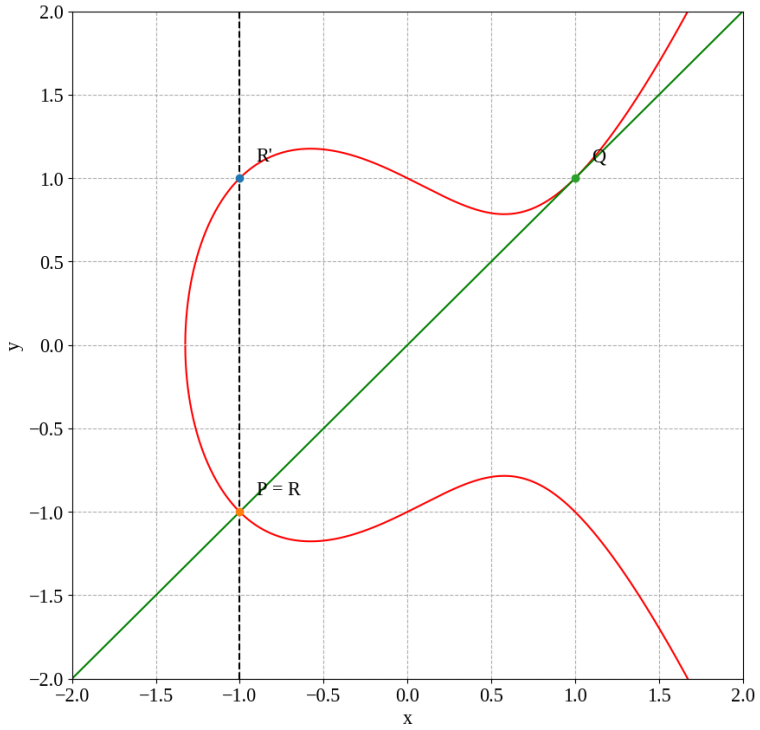

Figura 6.5: O terceiro elemento de $\overline{PQ}$ na curva é o próprio P.

6.2 Algoritmo da Adição

Passemos agora à análise algébrica da soma na curva elíptica. O algoritmo da soma de pontos na curva elíptica pode ser resumido assim:

Entrada:

- Curva elíptica C, tal que $y^2 = x^3 + ax + b$
- pontos $P, Q \in C$

Saída:

- Ponto $R = P + Q$.

1. Se $P = O$, retorne Q.
2. Se $Q = O$, retorne P.
3. Se $y_P = -y_Q$, retorne O.
4. Se $P = Q$, $\lambda = (3x^2 + A)/(2y)$.
   Senão, $\lambda = (y_Q - y_P)/(x_Q - x_P)$.
5. $x_R = \lambda^2 - x_P - x_Q$
6. $y_R = \lambda(x_P - x_R) - y_P$.
7. Retorne $R = (x_R, y_R)$.

Os passos 1, 2 e 3 seguem de como definimos a soma. Vejamos o raciocínio dos demais.

Determinando λ: passo 4

λ é o coeficiente da reta que passará por R. No caso $P = Q$, derivamos implicitamente a equação de Weierstrass $y^2 = x^3 + ax + b$, obtendo:

$$2y\frac{dy}{dx} = 3x^2 + a \qquad \Rightarrow \qquad \frac{dy}{dx} = \frac{3x^2 + a}{2y}$$

No caso dos pontos serem distintos, usamos a conhecida fórmula de Geometria Analítica para determinar o coeficiente da reta que passa por P e Q.

Determinando x_R e y_R: passos 5 e 6

As fórmulas de x_R e y_R seguem de substituir a fórmula da reta obtida, $y = \lambda x + \mu$, na fórmula da curva elíptica $y^2 = x^3 + ax + b$, e manipulá-las algebricamente.

A reta onde estão os três pontos tem a fórmula $y = \lambda x + \mu$. Uma vez obtido o λ e um ponto (x_1, y_1) da reta, é fácil ver que $\mu = y_1 - \lambda x_1$, e $y = \lambda x + \mu$.

Substituindo na equação da curva elíptica, temos que

$$y^2 = (\lambda x + \mu)^2 = x^3 + ax + b$$

Passando tudo para um lado, obtemos

$$0 = x^3 + (-\lambda^2)x^2 + (a - 2\lambda\mu)x + (b - \mu^2)$$

Sabemos que todo polinômio pode ser fatorado em termos que são da forma $(x - a_i)$, sendo os a_i's suas raízes. Logo,

$$x^3 + (-\lambda^2)x^2 + (a - 2\lambda\mu)x + (b - \mu^2) = (x - x_1)(x - x_2)(x - x_3)$$

Já conhecemos x_1 e x_2, que são x_P e x_Q. Abrindo a equação dos dois lados e igualando os coeficientes de x^2, chegamos em:

$$-\lambda^2 = -x^1 - x^2 - x^3 \quad \Rightarrow \quad x_R = \lambda^2 - x_P - x_Q \,.$$

Para y_R, temos aqui uma sutileza. Lembrando da fórmula da reta, poderíamos agora usar

$$y_R = \lambda x_R + \mu = \lambda x_R + y_P - \lambda x_P = \lambda(x_R - x_P) + y_P \,.$$

Mas, este y é o y do ponto colinear, e pela definição de adição que criamos, temos que inverter esse valor. Portanto, na verdade,

$$y_R = \lambda(x_P - x_R) - y_P \,.$$

6.3 Curvas elípticas em $\mathbb{F}_p$

Na definição que vimos, as curvas elípticas estão em $\mathbb{R}^2$. Por outro lado, para uso em redes de computadores, precisamos que os dados transmitidos sejam inteiros. Por isso, o uso de curvas elípticas em criptografia é feito considerando a curva sobre um corpo finito $\mathbb{F}_p$, onde p é um número primo fixado. Curvas elípticas em $\mathbb{R}^2$ não tem relação com a Matemática que tinha sido vista nos capítulos anteriores. Mas inserindo-as em $\mathbb{F}_p$, a teoria que vimos se torna relevante. É finalmente aqui que passamos a encarar $\mathbb{F}_p$ como um corpo finito tal qual definido anteriormente, que nos permite ter inversos da soma e multplicação (exceto zero) para todos os membros.

É interessante lembrar que números como -5 e $\frac{1}{4}$ tem significado distinto em $\mathbb{F}_p$. Por exemplo, em $\pmod 7$, $-5 \equiv 2 \pmod 7$, pois $2-(-5) \equiv 2+5 \equiv 7 \equiv 0 \pmod 7$; e $\frac{1}{4} \equiv 2 \pmod 7$, pois $\frac{1}{4} \equiv 4^{-1} \pmod 7$, e $4 * 2 \equiv 8 \equiv 1 \pmod 7$.

Além disso, a forma de Weierstrass deixa claro que teremos que tirar uma raiz quadrada em $\mathbb{F}_p$. O lema abaixo mostra que nem todos os números terão uma raiz quadrada:

Lema 6.3.1. *Se* $a^2 \equiv n \pmod p$*, então* $(p-a)^2 \equiv n \pmod p$*.*

Demonstração. Basta notar que: $(p-a)^2 \equiv (p^2 + 2pa + a^2) \equiv a^2 \equiv n \pmod p$. □

Dessa forma, os quadrados dos números $\pmod p$ serão "espelhados" quando ordenados. Por exemplo, os quadrados dos números $\pmod{17}$ são:

n	$n^2 \pmod{17}$	n	$n^2 \pmod{17}$
1	1	9	13
2	4	10	15
3	9	11	2
4	16	12	8
5	8	13	16
6	2	14	9
7	15	15	4
8	13	16	1

Tabela 6.1: Quadrados $\pmod{17}$.

Assim, nem todo valor de $\mathbb{F}_p$ será um quadrado - apenas $(p/2)+1$ valores serão quadrados. Além disso, pode ser que nem todos os quadrados válidos sejam alcançados

calculando $x^3 + ax + b$. A quantidade de pontos válidos que a combinação de curva e corpo gerará é um dos critérios para a seleção de ambos.

Também é interessante constatar que, caso existam soluções para $a \equiv \sqrt{b} \pmod{p}$, existirão sempre exatamente duas soluções - justamente os valores y e $-y$ necessários para haver (x, y) e $(x, -y)$ tal que as propriedades de grupo dos pontos da curva sob a adição definida seja válida.

Exemplifiquemos agora com a curva

$$y^2 = x^3 - 5x + 8 \qquad \text{sobre} \quad \mathbb{F}_{17}.$$

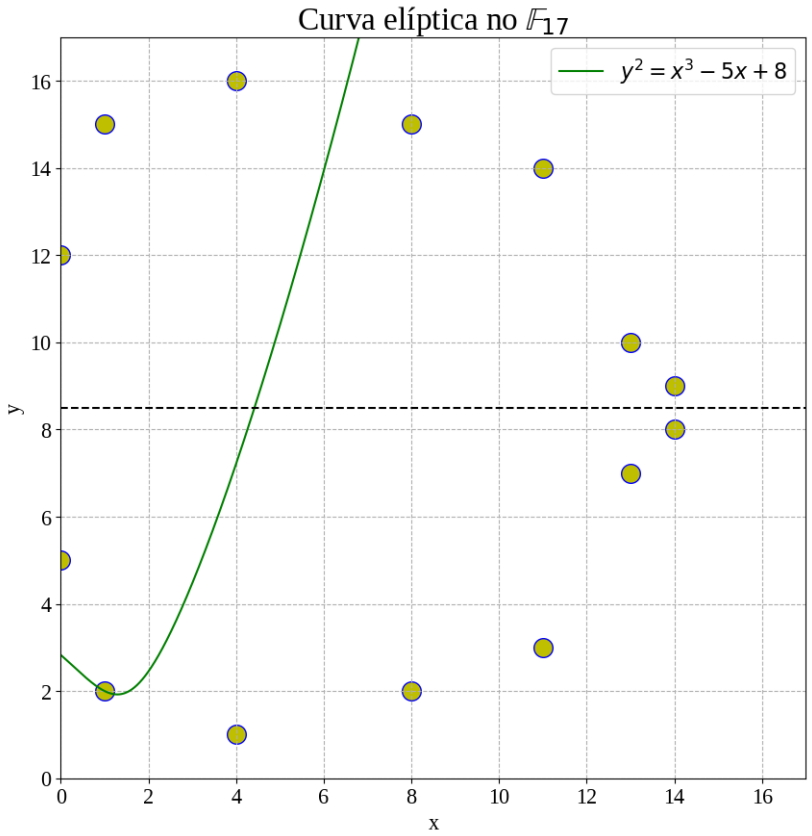

Figura 6.6: Uma mesma curva elíptica em $\mathbb{R}^2$ e $\mathbb{F}_p$ são objetos matemáticos distintos. Por coincidência, aqui o ponto $(1, 2)$ pertence a ambas.

Testando os valores possíveis de x, temos que apenas os valores 0, 1, 4, 7, 11, 13 e 14 fornecem quadrados em $\mathbb{F}_{17}$ (zero aqui é um valor válido - não estamos mais usando $\mathbb{F}_{17}$ como um grupo multiplicativo). Obtemos então os pontos

$$(0, 5) \quad (1, \ 2) \quad (4, \ 1) \quad (8, \ 2) \quad (11, 3) \quad (13, 7) \quad (14, 8)$$

e seus inversos

$$(0,12) \quad (1,15) \quad (4,16) \quad (8,15) \quad (11,14) \quad (13,\ 10) \quad (14,9).$$

A título de exemplo, somemos os pontos (1, 15) e (4, 1), utilizando o algoritmo da adição. Calculamos primeiro

$$\lambda = \frac{1-15}{4-1} = \frac{-14}{3} = \frac{3}{3} = 1,$$

porque $-14 \equiv 3 \pmod{17}$.

Depois disso, temos que

$$x_R = \lambda^2 - x_P - x_Q = 1^2 - 1 - 4 = -4 = 13$$

$$y_R = \lambda(x_P - x_R) - y_P = 1(1-13) - 15 = -12 - 15 = 5 + 2 = 7$$

Logo, $(13, 7) = (1, 15) + (4, 1)$. De fato, (13, 7) é um dos pontos listados acima.

7 Assinaturas Digitais em Curvas Elípticas

Chegamos, enfim, ao objetivo desta monografia. Combinando o mecanismo de ElGamal/DSA com a adição nas curvas elípticas, veremos novas possbilidades de algoritmos de Assinatura Digital. Examinaremos dois algoritmos distintos: o **ECDSA**, que foi padronizado pelo governo dos Estados Unidos em 1999; e, brevemente, o **EdDSA**, que foi concebido por uma equipe internacional de acadêmicos da área e teve seu paper publicado em 2011.

7.1 ECDSA

Quando Miller e Koblitz publicaram suas pesquisas sobre Criptografia em Curvas Elípticas, nenhum dos dois tentou patentar suas descobertas, de modo que ela ficou no domínio público. Em 1992 surge a primeira proposta do ECDSA (*Elliptic Curve Digital Signature Algorithm*) (Rivest_1992), que é então aceito como padrão pela ISO em 1998, pela ANSI (American National Standards Institute) em 1999 e pelo IEEE em 2000. Seguiremos esta definição. Como já comentamos, o algoritmo foi baseado no DSA e sua semelhança será clara.

Parâmetros públicos

1. Uma curva elíptica $y^2 = x^3 + ax + b$
2. Um corpo F_p, sendo p primo.
3. Um ponto G na curva,
4. que tenha ordem prima w.

Por exemplo, **para a curva usada na Bitcoin**:

1. a curva é $y^2 = x^3 + 7$;
2. o primo p é $2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$;

3. a coordenada x do ponto G, em notação hexadecimal, é 04 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8.

4. a ordem de G, novamente em hexadecimal, é FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141 (ver (4)).

Geração de chaves

Samantha escolhe um número x tal que $1 < a < q - 1$. x é sua chave privada. A partir dele, Samantha calcula $P = xG$. P é sua chave pública, que ela publica.

Assinatura

Como em ElGamal e DSA, Samantha tem que escolher um elemento aleatório k, chamado de **nonce**, tal que $1 < k < q$. **É importante notar que k não pode ser repetido em assinaturas diferentes**, e que fazê-lo permite descobrir a chave privada, como aconteceu com a Sony em 2010 (5). Além disso, temos um valor $d \pmod w$ para o documento.

Calculamos então $(x', y') = kG$. Depois, geramos os valores r e s, que juntos, compõem a assinatura digital:

$$r = x' \pmod w$$

$$s \equiv (d + ar)k^{-1} \pmod w$$

Se $r = 0$, escolhemos um novo k e repetimos o processo. Se não, publicamos r e s como nossa assinatura. r não pode ser zero porque teríamos $s = (d + a\,0)k^{-1} = (d)k^{-1}$, de modo que a assinatura não seria afetada pela chave privada e poderia ser forjada por qualquer um.

Verificação

Victor calculará dois valores a partir de r e s:

$$u_1 \equiv ds^{-1} \pmod{w}$$
$$u_2 \equiv rs^{-1} \pmod{w}$$

A coordenada x da soma $u_1G + u_2P \pmod{w}$ precisa ser igual a r. Vejamos por que isso acontece.

$$u_1G + u_2P = ds^{-1}G + rs^{-1}aG = (d + ar)s^{-1}G$$

Como $s = (d + a * r)k^{-1}$, podemos isolar $(d + ar)$ e $(d + ar) = sk$. Voltando ao cálculo:

$$(d + ar)s^{-1}G = sk\, s^{-1}G = kG$$

Como $r = x(kG)$, a verificação está demonstrada.

7.2 Exemplo de cálculo do ECDSA

Voltando à curva $y^2 = x^3 - 5x + 8$ no $\mathbb{F}_{17}$, vamos assinar um documento usando o ECDSA. Quero assinar um documento representado pelo valor **2**. O ponto $(1, 2)$ tem ordem $w = 5$ - um número primo, e portanto que serve para o algoritmo. Minha chave privada será $a = 4$. Já a pública, $4(1, 2) = (0, 5)$. Portanto, eu publico o número 0 como minha chave pública.

Expoente	1	2	3	4	5
Ponto	(1,2)	(13,7)	(13,10)	(0,5)	O

Tabela 7.1: Subgrupo gerado por (1,2) em $y^2 = x^3 - 5x + 8$ no $\mathbb{F}_{17}$.

O elemento aleatório k é 3; $3(1, 2) = (13, 10)$. Então $r \equiv 13 \equiv 3 \pmod{5}$ e

$$s \equiv (d + ar)k^{-1} \equiv (2 + 4 * 3) * 3^{-1} \equiv (14) * 2 \equiv 28 \equiv 3 \pmod{5},$$

porque $2 * 3 \equiv 6 \equiv 1 \pmod{5}$. Assim, a assinatura é o par $r = 3, s = 3$.

Para verificá-la, calculamos

$$u_1 \equiv ds^{-1} \equiv 2 * 2 \equiv 4 \pmod 5$$

$$u_2 \equiv 3 * 2 \equiv 1 \pmod 5$$

E finalmente

$$u_1 G + u_2 P \equiv 4 * (1,2) + 1 * (0,5) = 4 * (1,2) + 4 * (1,2) = 8 * (1,2) = 3 * (1,2) = (13,10)\,,$$

pois a ordem do grupo é 5.

Lembrando que $r = 3$ e $x(13,10) \equiv 13 \equiv 3 \pmod 5$, nossa assinatura funcionou corretamente.

7.3 EdDSA

Algumas suspeitas políticas pairam sobre como o padrão ECDSA foi especificado e particularmente sobre os parâmetros Secp256k1 especificados pelo governo dos Estados Unidos (veja (6), (Bernstein_2013)) - os mesmos que foram adotados pela Bitcoin e várias outras criptomoedas. Motivados por isso, pesquisadores independentes publicaram, em 2011, um artigo (Bernstein_2011) propondo um novo algoritmo de assinatura digital. Este novo algoritmo foi aceito pela Internet Research Task Force (IRTF), que é uma organização internacional para padrões da Internet, sendo a responsável por vários protocolos da rede, na RFC 8032 (7), em 2017. Nessa seção, apresentamos brevemente o funcionamento desse tipo de AD.

7.3.1 Curva de Edwards

A principal diferença do EdDSA para o ECDSA é não usar a curva elíptica tal qual definida pela forma de Weierstrass. Na verdade, **curvas de Edwards** são as adotadas pelo EdDSA. Curvas de Edwards são assim chamadas por terem sido estudadas

pelo matemático Harold Edwards em 2007 e tem a forma

$$ax^2 + y^2 = 1 + dx^2y^2\,,$$

onde $a, d \in \mathbb{K} - \{0\}$. No caso específico do EdDSA, utilizamos $a = -1$.

Observamos que esse tipo de curva tem grau 4 (devido ao termo x^2y^2), enquanto curvas elípticas na forma de Weierstrass têm grau 3. No entanto, na Geometria Algébrica, ambas fazem parte da classe das curvas elípticas, que podem ser definidas de maneira geral como curvas projetivas birracionalmente equivalentes a curvas não-singulares de gênero 1 com um ponto destacado, definição que foge do escopo desse trabalho. É possível provar que toda curva elíptica não-singular tem uma representação polinomial de grau 3, o que não é o caso da curva de Edwards, que é singular.

Também podemos observar que os pontos $(0, 1)$ e $(0, -1)$ pertencem a todas as curvas de Edwards. Além disso, observamos que existe uma simetria dada pelos termos quadrados da equação, o que dá a essa classe de curvas uma aparência bastante distinta da forma de Weierstrass.

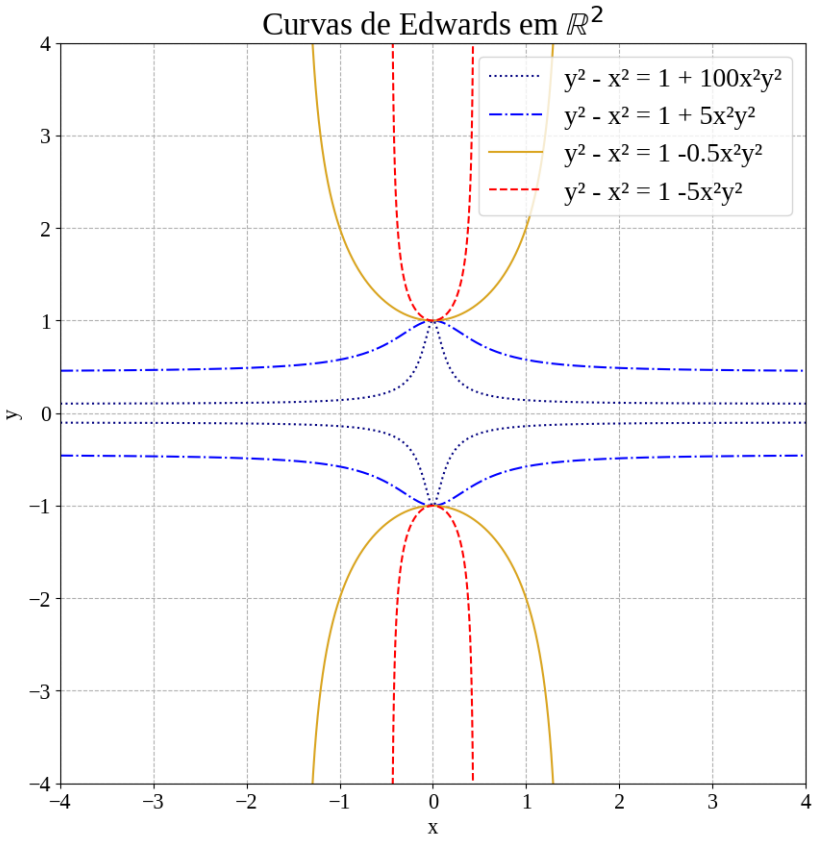

Figura 7.1: Exemplos de curvas de Edwards em $\mathbb{R}^2$.

A soma, por sua vez, também é definida de modo distinto:

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1 y_2 + x_2 y_1}{1 + d x_1 x_2 y_1 y_2}, \frac{y_1 y_2 - a\, x_1 x_2}{1 - d\, x_1 x_2 y_1 y_2} \right)$$

Nesse caso, o elemento neutro da soma é $(0, 1)$ e simétrico de qualquer ponto (x, y) é o ponto $(-x, y)$.

7.3.2 Parâmetros

Os principais parâmetros públicos para uma implementação da EdDSA são:

1. Um primo p que define um corpo $\mathbb{F}_p$;
2. Uma curva de Edwards, na forma $-x^2 + y^2 = 1 + dx^2y^2$, onde $d \in \mathbb{F}_p$;
3. Um ponto base B na curva, que tenha ordem do tipo $2^c\ell$, isto é, B somado com si mesmo $2^c\ell$ vezes é igual ao elemento neutro $(0, 1)$;
4. Uma função de hashing resistente a colisão (usada para gerar *nonces*, os elementos aleatórios).

7.3.3 Funcionamento

As chaves pública e privada são geradas de forma análoga ao ECDSA. Geramos uma chave privada $a \pmod{\ell}$ e a chave pública será o ponto $A = aB$.

A assinatura de uma mensagem M é novamente um par (R, s), onde R é um ponto da curva e $0 < S < \ell$ é o inteiro necessário para que a equação de verificação

$$2^c SB \equiv 2^c R + 2^c \,\text{Hashing}(R, A, M)\, A \pmod{\ell}$$

esteja correta. Aqui, o interessante é que o elemento aleatório k chamado **nonce**, já visto em ElGamal/DSA, é gerado automaticamente pela definição da assinatura, através de uma função de hashing segura. Sem dúvida, os autores da cifra escolheram implementá-la assim para já incluir esse nível de segurança na própria definição e com isso diminuir riscos desnecessários devido a possíveis desatenções.

Se considerarmos que $R = rB$, $A = aB$ e $\text{Hashing}(R, A, M) = h$, podemos reescrever a equação, a ser resolvida para S, como

$$2^c S \equiv 2^c r + 2^c ha \pmod{\ell} \Rightarrow S \equiv r + ha \pmod{\ell}.$$

O conhecimento da chave privada a permite ao assinante calcular s trivialmente, enquanto que um atacante teria que resolver o Problema do Logaritmo Discreto no grupo gerado pelo ponto G para poder falsificar uma assinatura.

7.3.4 ed25519

Bem como a curva "oficial" do ECDSA é a Secp256k1, os criadores da EdDSA também especificaram parâmetros padrão, que juntos são chamados de **ed25519**.

1. O primo p é $2^{255} - 19$ (daí o nome da especificação);
2. A curva é $-x^2 + y^2 = 1 - (121665/121666)x^2y^2$;
3. A base é $x = 9$, que tem ordem prima $w = 2^{252}+27742317777372353535851937790883648 49$;
4. A função de hashing é SHA-512.

7.3.5 Vantagens

Como fica evidente no paper de 2011, a principal preocupação dos criadores da EdDSA foi com aspectos práticos de Assinatura Digital em Curvas Elípticas. A implementação oficial, por exemplo, gera k's pseudo-aleatoriamente para assinaturas distintas (alguns usuários de Bitcoin já tiveram seus fundos roubados por não tomar esse cuidado e repetir k em assinaturas diferentes). Várias seções são dedicadas a mostrar que o algoritmo para a ed25519 tem boa eficiência para assinar e verificar assinaturas com processadores da época, e outras argumentam que ele é mais resistente que os algoritmos existentes aos chamados **ataques de canais laterais** (em inglês, *side-channel attacks*). Esses ataques não atacam a Matemática do algoritmo em si - na verdade, eles tentam obter informação secreta de modo indireto. Por exemplo, se conseguimos monitorar o uso de energia elétrica de uma máquina, ou a temperatura da CPU, conseguimos saber

quando ela está realizando cálculos mais intensos e quando não - isso pode servir para tentar determinar a posição de zeros e uns numa chave privada. Outro ataque lateral é tentar ver o uso de memória, quais páginas de memória estão sendo acessadas em que ordem, etc. De qualquer modo, o paper não defende que o ECDSA não é seguro do ponto de vista matemático.

7.4 ECDLP - Problema do Logaritmo Discreto em Curvas Elípticas

> *The strongest techniques known for the [Discrete Logarithm Problem in $\mathbb{F}_p^*$] do not seem to be applicable to the elliptic curve analog.*
>
> Neal Koblitz, (Koblitz_1987)

No passo de geração da chave, Samantha calculou

$$P = aG,$$

e o número a se tornou sua chave privada. O **Problema do Logaritmo Discreto na Curva Elíptica** (ECDLP, na sigla em inglês) consiste em, dados P, G e demais elementos públicos da cifra que vimos no item anterior, recuperar o número a. A segurança da ECDSA se baseia na dificuldade de realizar isso.

O Algoritmo de Colisão que vimos, como caso ilustrativo de algoritmos de colisão para resolver o Problema do Logaritmo Discreto, funciona em qualquer grupo. Por outro lado, não existe um análogo para o Cálculo de Índice para o ECDLP em qualquer curva. Portanto, não se conhece um algoritmo subexponencial para o ECDLP para curvas escolhidas corretamente (ainda que certas classes de curvas tenham fraquezas). Citando (Hoffstein_2008):

> Em outras palavras, apesar da natureza altamente estruturada do grupo $E(\mathbb{F}_p)$, os algoritmos mais rápidos para resolver o ECDLP não são melhores que o algoritmo genérico que funciona igualmente bem para resolver o Problema do

> Logaritmo Discreto em qualquer grupo. [...] Portanto, o ECDLP parece ser muito mais difícil que o DLP. (p. 296)

Aliando isso à maior eficiência de espaço da assinatura e chaves, fica claro o motivo da Assinatura Digital em Curvas Elípticas estar ganhando terreno.

8 Considerações finais

Neste texto, vimos os fundamentos da Matemática que permitem a construção dos sistemas de Assinatura Digital mais comuns; mais precisamente, elementos de Teoria dos Números e Álgebra Abstrata. Exploramos também a resolução do Problema do Logaritmo Discreto, da dificuldade do qual depende a segurança dessas cifras. Vimos também como as idéias num corpo finito $\mathbb{F}_p$ definido por um primo p podem ser transladadas ao grupo comutativo da soma em curvas elípticas, definido cuidadosamente, e por que essa opção está ganhando força entre matemáticos e desenvolvedores de sistemas.

Escolher este tema para minha monografia foi uma decisão feliz, porque aprendi blocos essenciais de conhecimento para entender não só Assinaturas Digitais, mas também a prática criptográfica moderna em sistemas digitais. Ainda assim, pela falta de tempo ou pela dificuldade do tópico, não examinei outras questões que cativaram meu interesse:

1. Em algumas aplicações, a própria curva elíptica pode permanecer secreta. Quais ataques existem sobre o ECDSA, se a curva for mal escolhida? Como escolher uma boa combinação de curva elíptica e corpo $\mathbb{F}_p$?

2. Qual a Álgebra necessária para entender mais sobre a Curva de Edwards, e portanto o EdDSA?

3. Como um computador quântico pode ser explorado para atacar o DLP/ECDLP, visto que ambos não são considerados "à prova de computadores quânticos"?

Bibliografia

[1] **SSH keys**. https://wiki.archlinux.org/index.php/SSH_keys. Acessado: 30/03/2019.

[2] **How to design an elliptic-curve signature system**. http://blog.cr.yp.to/20140323-ecdsa.html. Acessado: 30/03/2019.

[3] **FIPS 186 - (DSS), Digital Signature Standard**. http://www.itl.nist.gov/fipspubs/fip186.htm. Acessado: 07/06/2019.

[4] **Secp256k1 - Bitcoin Wiki**. https://en.bitcoin.it/wiki/Secp256k1. Acessado: 22/06/2019.

[5] **Hackers Describe PS3 Security As Epic Fail, Gain Unrestricted Access**. https://www.exophase.com/20540/hackers-describe-ps3-security-as-epic-fail-gain-unrestricted-access/. Acessado: 22/06/2019.

[6] **SafeCurves: Rigidity**. http://safecurves.cr.yp.to/rigid.html.

[7] **RFC 8032 - Edwards-Curve Digital Signature Algorithm (EdDSA)**. https://tools.ietf.org/html/rfc8032. Acessado: 22/06/2019.

[Bernstein_2011] Bernstein, D. J.; Duif, N.; Lange, T.; Schwabe, P. ; Yang, B.-Y. **High-speed high-security signatures**. https://ed25519.cr.yp.to/ed25519-20110926.pdf, 2011.

[Bernstein_2013] Bernstein, D. J.; Lange, T. **Security dangers of the NIST curves**. https://www.hyperelliptic.org/tanja/vortraege/20130531.pdf, 2013.

[ElGamal_1985] ElGamal, T. **A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms**. In: IEEE Transactions on Information Theory, volume IT-31 no. 4, p. 469–472. IEEE, 1985.

[Gallian_2003] Gallian, J. A. **Contemporary Abstract Algebra, 5th ed.** Cengage Learning, 2003.

[Ganley_1994] Ganley, M. J. **Digital signatures and their uses**. In: Computers & Security, volume 13 de **Issue 5**, p. 385–391. Springer, 1994.

[Hoffstein_2008] Hoffstein, J.; Pipher, J. ; Silverman, J. **An Introduction to Mathematical Cryptography, 1a ed.** Springer, 2008.

[Koblitz_1987] Koblitz, N. **A Method for Obtaining Digital Signatures and Public-Key Cryptosystems**. In: Mathematics of Computation, volume 48, no. 177, p. 203–209. American Mathematical Society, 1987.

[Miller_1985] Miller, V. S. **Use of Elliptic Curves in Cryptography**. In: Advances in Cryptology, volume 218, p. 417–426, Berlin, 1985. Springer Verlag.

[Rivest_1978] Rivest, R. L.; Shamir, A. ; Adleman, L. **A Method for Obtaining Digital Signatures and Public-Key Cryptosystems**. In: Communications of the ACM, volume 21 Issue 2, p. 120–126, New York, USA, 1978. ACM.

[Rivest_1992] Rivest, R. L.; Hellman, M. E.; Anderson, J. C. ; Lyons, J. W. **Responses to NIST's proposal**. In: Communications of the ACM, volume 35, p. 41–54, New York, NY, USA, 1992. ACM.

[Silverman_2015] Silverman, J. H.; Tate, J. T. **Rational Points on Elliptic Curves, 2a ed.** Springer, 2015.