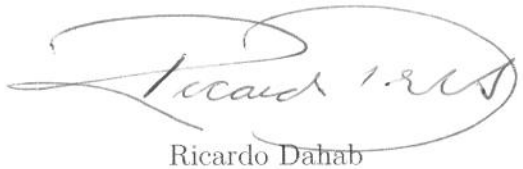
Cifrassinatura sem certificados em curvas supersingulares sobre corpos binários

Este exemplar corresponde à redação da Dissertação apresentada para a Banca Examinadora antes da defesa da Dissertação.

Campinas, 22 de junho de 2009.



Ricardo Dahab
Instituto de Computação
Universidade Estadual de Campinas
(Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Crisllene Queiroz Custódio – CRB8 / 7966

Morais, Eduardo Moraes de
M791c Cifrassinatura sem certificados em curvas elípticas supersingulares sobre corpos binários / Eduardo Moraes de Morais -- Campinas, [S.P. : s.n.], 2009.

Orientador : Ricardo Dahab
Dissertação (Mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Curvas elípticas. 2. Cifrassinatura. 3. Criptografia - Certificados e licenças. 4. Formas bilineares. I. Dahab, Ricardo. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglês: Certificateless signcryption on supersingular elliptic curves over bilinear fields

Palavras-chave em inglês (Keywords): 1. Elliptic curves. 2. Signcryption. 3. Cryptography - certificate and license. 4. Bilinear forms.

Área de concentração: Criptografia

Titulação: Mestre em Ciência da Computação

Banca examinadora: Prof. Dr. Ricardo Dahab (IC-Unicamp)
Prof. Dr. Julio César López Hernández (IC-Unicamp)
Prof. Dr. Anderson C. A. Nascimento (ENE-UnB)

Data da defesa: 27/02/2009

Programa de Pós-Graduação: Mestrado em Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 27 de fevereiro de 2009, pela Banca examinadora composta pelos Professores Doutores:

Anderson C A Nascimento

Prof. Dr. Anderson Clayton Alves Nascimento
Departamento de Engenharia Elétrica / Universidade de Brasília.

Prof. Dr. Julio César López Hernández
IC / UNICAMP.

Prof. Dr. Ricardo Dahab
IC / UNICAMP.

Instituto de Computação
Universidade Estadual de Campinas

Cifrassinatura sem certificados em curvas supersingulares sobre corpos binários

Eduardo Moraes de Morais[1]

Junho de 2009

Banca Examinadora:

- Ricardo Dahab
  Instituto de Computação
  Universidade Estadual de Campinas (Orientador)
- Anderson C. A. Nascimento
  Departamento de Engenharia Elétrica
  Universidade de Brasília
- Julio César López Hernández
  Instituto de Computação
  Universidade Estadual de Campinas
- Marco Aurélio Amaral Henriques (Suplente)
  Faculdade de Engenharia Elétrica e de Computação
  Universidade Estadual de Campinas
- Eduardo Cândido Xavier (Suplente)
  Instituto de Computação
  Universidade Estadual de Campinas

[1]Suporte financeiro de: FAPESP (processo 2005/04248-0) 2006–2008 e Universidade Estadual de Campinas

Resumo

A criptografia baseada em identidades representa uma alternativa ao modelo de certificação digital, exigindo menor esforço para solucionar o problema de autenticidade da chave pública, mas perdendo a custódia da chave privada, que será gerada por uma autoridade de confiança. O modelo de criptografia sem certificados soluciona o problema da custódia da chave privada sem a utilização de certificados digitais. Neste modelo, o usuário tem a posse de uma parte da chave privada e com isso a chave pública passa a ser constituída de uma parte gerada pela autoridade de confiança e uma parte gerada pelo usuário.

A cifrassinatura é uma primitiva criptográfica que reúne as vantagens do ciframento e da assinatura em uma única operação, permitindo maior eficiência e segurança.

A literatura possui diversas propostas de ciframento sem certificados e assinatura sem certificados, mas não tem uma proposta genérica de cifrassinatura sem certificados. Este trabalho propõe um protocolo de cifrassinatura sem certificados eficiente, que pode ser implementado usando dois emparelhamentos bilineares.

Considerando a importância de emparelhamentos bilineares para a construção do protocolo proposto, este trabalho apresenta os conceitos matemáticos necessários para a obtenção de emparelhamentos bilineares eficientes e resistentes a ataques ao problema do logaritmo discreto sobre a curva elíptica e sobre o corpo de extensão resultante do cálculo do emparelhamento bilinear.

São apresentados também algoritmos eficientes para aritmética de precisão arbitrária, aritmética de curvas elípticas e cálculo de emparelhamentos. Além disso, são discutidos modelos formais de segurança, como por exemplo o modelo do oráculo aleatório.

Finalmente, o modelo de criptografia baseada em identidades e o modelo de criptografia sem certificados são discutidos e com isso é possível apresentar a proposta de cifrassinatura sem certificados e argumentar que esta proposta é segura e eficiente.

Abstract

Identity based cryptography is an alternative to digital certification, which requires less computational effort to solve the problem of public key authenticity. On the other hand, identity based cryptography has the problem of key escrow, because the private key is generated by a trust authority. The certificateless cryptography model solves the key escrow problem without digital certificates. In this model, the user computes a parcial private key that is used to compose the entire private key. In the same way, the public key has two parts: one generated by the user and the other generated by the trust authority.

Signcryption is a cryptographic primitive that has the advantages of encryption and signature together in a single operation, allowing the construction of secure and efficient protocols.

The literature has many certificateless encryption and certificateless signature protocols, but there is no generic and efficient certificateless signcryption scheme. This work proposes an efficient certificateless signcryption protocol, that can be implemented with just two bilinear pairings.

Considering the importance of bilinear pairings for the construction of the proposed protocol, this work presents the mathematical concepts for efficient bilinear pairings, that can resist against discrete logarithm atacks on the elliptic curve and on the extension field.

This works also presents efficient algorithms for big number arithmetic, elliptic curve arithmetic and the Miller algorithm for pairings. It also presents formal security models, such as the random oracle model.

Finally, identity based cryptography and certificateless cryptography models are defined and the proposed certificateless signcryption scheme is presented and we argue that it is secure and efficient, although no formal proof is given.

Agradecimentos

Eu gostaria de agradecer ao apoio da FAPESP como financiadora deste projeto, além do Instituto de Computação e da Unicamp, que foram responsáveis por minha formação acadêmica de qualidade. Com isso, gostaria de lembrar a importância dos funcionários que se esforçaram para garantir a continuidade das atividades acadêmicas.

Gostaria também de agradecer minha família que deu o amparo necessário para superar os problemas e as dificuldades encontradas, apoiando minhas escolhas e decisões, permitindo assim que eu chegasse até aqui com segurança e confiança.

Eu gostaria ainda de agradecer aos amigos que moram ou moraram comigo neste período, principalmente por não terem apagado minha tese quando tiveram a chance, e também pelo apoio moral e pelo incentivo que me deram, e a paciência que tiveram para ouvir os relatos dos progressos que eu realizava.

Gostaria de agradecer minha namorada, que apesar de ter aparecido em minha vida no final do meu mestrado, chegou em um momento crítico e foi muito importante para me dar a calma necessária para a reta final do meu trabalho.

Gostaria também de agradecer meus colegas do laboratório de criptografia aplicada pelas discussões que tivemos, pelo convivío em conferências e também em momentos de laser. Em particular, gostaria de agradecer o Diego Aranha e o Rafael Castro pelo trabalho que desenvolvemos no protocolo de cifrassinatura sem certificados e em especial pelo apoio e amizade do Augusto Devegili, com quem aprendi muito durante estes últimos anos.

Finalmente, gostaria de agradecer ao meu orientador, Ricardo Dahab, por ter me ajudado a chegar até aqui de forma tão natural, guiando meu trabalho de forma eficiente e prática. Obrigado pela oportunidade de tão grande aprendizado, espero que minha vida acadêmica esteja apenas começando com este passo inicial, que foi dado com sucesso principalmente pela ajuda que tive destes que citei acima e principalmente do meu orientador.

Sumário

Lista de Algoritmos

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A grande dificuldade de resolver o problema de criptografia simétrica é o acordo de chaves utilizando um meio de comunicação inseguro como a internet. Para resolver este problema é possível usar o protocolo baseado em criptografia assimétrica de Diffie-Hellman. Esta, por sua vez, baseia-se na confiança da posse da chave pública. A certificação digital consolidou-se como um mecanismo que garante a posse da chave pública através de uma hierarquia de garantias, onde a confiança em uma entidade da hierarquia pode ser estendida para as entidades abaixo dela. Uma alternativa a este procedimento é a criptografia baseada em identidades, onde a chave pública é a própria identidade do usuário, não havendo a necessidade de garantir a posse da mesma. Por outro lado, neste cenário é necessária a presença de uma autoridade de confiança para gerar e distribuir a chave privada de cada usuário, levando a problemas com relação à custódia e revogação do par de chaves. O primeiro problema tem conseqüências jurídicas que ainda estão sendo discutidas e analisadas, enquanto o segundo problema, da revogação do par de chaves, implica diretamente na mudança da identidade do usuário, o que é impossível ou indesejado na grande maioria dos casos. A criptografia sem certificados é uma solução intermediária, onde parte da chave privada é gerada pela autoridade de confiança e a outra parte é gerada pelo próprio usuário, resolvendo o problema da custódia da chave privada.

Tanto a criptografia baseada em identidades como a criptografia sem certificados podem ser implementadas de forma eficiente através de emparelhamentos bilineares, que por sua vez são construídos utilizando curvas elípticas. Tendo isso em vista, fica claro que a evolução dos protocolos está atrelada aos algoritmos para cálculo de emparelhamentos bilineares, assim como aos modelos de segurança associados. Desta forma, os conceitos matemáticos subjacentes têm grande importância no processo de construção de novos algoritmos e também no estabelecimento de modelos de segurança mais fieis.

A primeira utilização de emparelhamentos bilineares em criptografia ocorreu em um ataque ao problema do logaritmo discreto em uma curva elíptica supersingular, realizada

por Menezes, Okamoto e Vanstone (emparelhamento de Weil) e também por Frey e Rück (emparelhamento de Tate). Com sua utilização na implementação eficiente do esquema de Boneh e Franklin [BF01] de criptografia baseada em identidades, tornou-se um método conhecido pela comunidade. Outros protocolos podem ser citados como por exemplo (i) o protocolo de uma rodada para Diffie-Hellman tripartido, de Joux [Jou00]; (ii) o criptossistema baseado em emparelhamentos de Sakai, Ohgishi e Kasahara [SOK00], e (iii) o protocolo de acordo de chaves para autenticação baseada em identidade usando emparelhamentos, de Smart [Sma02].

Dentre os emparelhamentos existentes, os mais conhecidos são os de Weil e de Tate [Was03]. Este último pode ser calculado mais eficientemente do que o primeiro, havendo otimizações que o tornaram mais eficaz, como a fórmula fechada para calcular o emparelhamento de Tate em corpos binários, devida a Duursma e Lee [DL03].

Este trabalho visa reunir o conhecimento matemático necessário para o entendimento dos conceitos envolvidos na definição e cálculo de emparelhamentos bilineares, tendo em vista um estudo específico das estruturas algébricas utilizadas, com exemplos simples, para que o leitor possa visualizar na prática os conceitos aqui apresentados.

A motivação deste trabalho é o estudo dos conceitos envolvidos em criptografia baseada em emparelhamentos, desde áreas da matemática como por exempo álgebra comutativa, geometria algébrica e análise complexa, até definições e modelos que permitem demonstrar a segurança de um protocolo. Com isso, deseja-se obter um conjunto de parâmetros bons para implementação do emparelhamento de Tate-Lichtenbaum, isto é, respeitando um compromisso entre eficiência e segurança. Para isso, serão estudados algoritmos para aritmética de corpos finitos e de curvas elípticas, além do algoritmo de Miller para computar emparelhamentos bilineares em tempo polinomial. Com isso, propomos um protocolo de cifrassinatura sem certificados, já que por um lado, a cifrassinatura proporciona confidencialidade, autenticidade e não-repúdio, enquanto a criptografia sem certificados representa uma alternativa que parece resolver problemas tanto da criptografia baseada em certificação digital, quanto da criptografia baseada em identidades.

1.1 Organização deste documento

Este documento está organizado da seguinte forma: o capítulo 2 descreve os conceitos matemáticos importantes para o cálculo de emprelhamentos bilineares; o capítulo 3 apresenta os algoritmos para aritmética de corpos finitos, curvas elípticas e emparelhamentos; o capítulo 4 reúne informações sobre a segurança dos algoritmos e protocolos perante ataques conhecidos na literatura e com isso é possível definir um conjunto de parâmetros bons para o cálculo de emparelhamentos bilineares; o capítulo 5 apresenta aplicações de emparelhamentos e define com mais detalhes a criptografia baseada em identidades, crip-

tografia sem certificados e o conceito de cifrassinatura; o capítulo 6 propõe um protocolo genérico para cifrassinatura sem certificados; o capítulo 7 conclui o trabalho.

Capítulo 2

Conceitos Preliminares

> Quem desejar conhecimento, há de esforçar-se por adquiri-lo
>
> John Ruskin

2.1 Introdução

A relação entre matemática e criptografia sempre foi estreita, mas dentre as áreas da matemática, sem dúvida a álgebra e a teoria dos números têm uma ligação ainda mais íntima. Um código é um mapeamento entre um conjunto de símbolos para outro, sejam esses símbolos letras, números ou bits. Estes conjuntos de símbolos podem ser encarados como elementos de uma estrutura algébrica, permitindo o estudo deste mapeamento através da matemática.

2.2 Grupos

Nesta seção serão introduzidos conceitos de álgebra abstrata, onde operações entre elementos de conjuntos arbitrários são definidas de modo que certas propriedades sejam válidas. Este conjunto, juntamente com estas operações definidas entre seus elementos, possui estrutura interessante, onde certos teoremas podem ser demonstrados, de forma que qualquer outro sistema com a mesma estrutura possa desfrutar dos mesmos teoremas e propriedades. Desta forma, a álgebra abstrata permite generalizar para certos tipos de conjuntos aquelas propriedades que eram válidas apenas para sistemas específicos, de modo a extrair os requisitos essenciais que tornam essas construções possíveis. Exemplos de estruturas algébricas que serão estudadas são: grupos, anéis, corpos, curvas elípticas, etc. Com isso, propriedades que são válidas para grupos genéricos, podem ser aplicadas

tanto em teoria dos números como em geometria algébrica, que são casos particulares de grupos.

2.2.1 Definições Preliminares

Definição 2.2.1. Um grupo é definido como um par $(G, .)$, onde G representa um conjunto e '.' representa uma operação binária definida entre elementos de G, tal que

1. se $a \in G$ e $b \in G$, então $a.b \in G$;
2. existe um elemento $e \in G$, denominado elemento neutro, tal que para qualquer elemento $a \in G$, temos que $a.e = e.a = a$;
3. para qualquer elemento $a \in G$, existe $a^{-1} \in G$, denominado inverso de a, tal que $(a).(a^{-1}) = (a^{-1}).(a) = e$;
4. para quaisquer elementos $a, b, c \in G$, temos que $a.(b.c) = (a.b).c$, esta propriedade é denominada propriedade associativa.

Exemplo 2.2.2. O conjunto dos números inteiros $\mathbb{Z}$, com a operação de soma usal, forma um grupo abeliano, cujo elemento neutro é o 0. Cada elemento $a \in \mathbb{Z}$ possui inverso dado por $-a \in \mathbb{Z}$.

Exemplo 2.2.3. O conjunto dos números reais $\mathbb{R}$ não nulos, com a operação de multiplicação usual, forma um grupo abeliano, cujo elemento neutro é o 1. Cada elemento $a \in \mathbb{R}$ possui inverso dado por $1/a \in \mathbb{R}$.

Exemplo 2.2.4. O conjunto $\mathbb{Z}_n = \{0, 1, ..., n-1\}$, com a operação de soma módulo n, forma um grupo abeliano, cujo elemento neutro é o 0. Cada elemento $a \in \mathbb{Z}_n$ possui inverso dado por $(n - a) \in \mathbb{Z}_n$.

Definição 2.2.5. Um grupo onde $a.b = b.a$, para quaisquer elementos $a, b \in G$ é denominado *grupo abeliano.*

Teorema 2.2.6. O elemento neutro de um grupo é necessariamente único. Da mesma forma, dado um elemento $x \in G$, seu inverso, denotado por x^{-1}, é único.

Definição 2.2.7. A *ordem de um grupo* $(G, .)$ é a cardinalidade do conjunto G.

Definição 2.2.8. Um *grupo finito* é aquele que possui um número finito de elementos, ou seja, possui ordem finita.

Definição 2.2.9. A *ordem de um elemento* $a \in G$ é definida como sendo o menor inteiro k, denotada por $ord_a(G)$, tal que $a^k = e$.

Definição 2.2.10. Um subconjunto $H \subset G$ é um *subgrupo* de G se ele próprio for um grupo com relação a operação definida em G. Para verificar se um dado subconjunto é subgrupo, basta verificar se cada elemento $a \in H$ possui inverso $a^{-1} \in H$ e se para quaisquer elementos $a, b \in H$, então $a.b \in H$.

Exemplo 2.2.11. Seja $\mathbb{Z}$ o grupo definido no exemplo 2.2.2 e $2\mathbb{Z} = \{..., -4, -2, 0, 2, 4, ...\}$, formado pela soma de cada elemento de $\mathbb{Z}$ consigo próprio 2 vezes, obtendo o conjunto dos números pares. Então, $2\mathbb{Z}$ é um subgrupo de $\mathbb{Z}$, pois dados dois elementos $a, b \in 2\mathbb{Z}$, $a + b$ é um número par, portanto pertence a $\mathbb{Z}$. Além disso, dado um elemento $a \in 2\mathbb{Z}$, existe $-a \in \mathbb{Z}$, tal que $-a$ seja o inverso de a.

Exemplo 2.2.12. Analogamente ao exemplo 2.2.11, pode-se construir um subgrupo $n\mathbb{Z}$ de $\mathbb{Z}$, onde $n\mathbb{Z}$ é formado pela soma de cada elemento de $\mathbb{Z}$ consigo próprio n vezes.

Definição 2.2.13. Um elemento $a \in G$ é denominado *gerador* se todo elemento de G puder ser escrito como uma potência de a, ou seja, se $b \in G$, então existe um inteiro x tal que $b = a^x$. O *subgrupo gerado* por um elemento $a \in G$ é composto por elementos da forma a^i, para $0 \leq i \leq ord_a(G)$. Um *grupo cíclico* é aquele que pode ser obtido por apenas um gerador.

2.2.2 Grupo Quociente

Definição 2.2.14. Seja S um conjunto arbitrário. Uma *relação de equivalência* R em S é definida como sendo um subconjunto do produto cartesiano $S \times S$, tal que

(a) $(s, s) \in R$, para qualquer $s \in S$ (reflexividade).

(b) Se $(s, t) \in R$, então $(t, s) \in R$ (simetria).

(c) Se $(s, t) \in R$ e $(t, u) \in R$, então $(s, u) \in R$ (transitividade).

Definição 2.2.15. É importante notar que uma relação de equivalência R em S induz uma partição de S em *classes de equivalência* da forma $[s] = \{t \in S | (s, t) \in R\}$.

Definição 2.2.16. Dado um grupo G, um subgrupo H e $a \in G$, definimos uma *classe lateral a esquerda* como sendo $a.H = \{a.h | h \in H\}$.

Definição 2.2.17. Dado um grupo G, um subgrupo H e $a \in G$, definimos uma *classe lateral a direita* como sendo $H.a = \{h.a | h \in H\}$.

Se G for um grupo abeliano, então $a.H = H.a$ e portanto não faz sentido diferenciar classe lateral a esquerda de classe lateral a direita.

Teorema 2.2.18. Seja H um subgrupo de G. Então a ordem de H divide a ordem de G.

Prova. É fácil ver que se $a \notin H$, então $a.H \cap H = \{\oslash\}$, pois caso contrário, teríamos $a.h_1 = h_2$, para $h_1, h_2 \in H$. De forma que $a = h_2.h_1^{-1}$ e portanto $a \in H$, o que é uma contradição. Além disso, $|a.H| = |H|$, pois senão teríamos dois elementos distintos $h_1, h_2 \in H$, tais que $ah_1 = ah_2$, mas isso implica que $a(h_1 - h_2) = 0$ e portanto, como $a \neq 0$, temos que $h_1 = h_2$, uma contradição. Com isso, G pode ser particionado em classes laterais derivadas de H. Ou seja, $G = H \cup a_1.H \cup ... \cup a_n.H$, onde a_i não pertence a H e também não pertence a nenhuma outra classe lateral $a_j.H$, para $i \neq j$.

Logo, $|G| = (n+1).|H|$. □

Definição 2.2.19. O *grupo quociente* G/H é definido como sendo formado pelas classes de equivalência geradas por este particionamento de G em função de H.

Teorema 2.2.20. $|G/H| = |G|/|H|$.

Exemplo 2.2.21. Por exemplo, seja $\mathbb{Z}$ o conjunto dos números inteiros, então $n\mathbb{Z}$ representa o subgrupo formado pelos múltiplos de n, definido no exemplo 2.2.12. Desta forma, o grupo quociente $\mathbb{Z}/n\mathbb{Z}$ é formado pelos conjuntos laterais:

$$\begin{aligned} n\mathbb{Z} &= \{0, n, 2n, ...\} \\ n\mathbb{Z} + 1 &= \{1, n+1, 2n+1, ...\} \\ &\vdots \\ n\mathbb{Z} + (n-1) &= \{n-1, 2n-1, 3n-1, ...\} \end{aligned}$$

Com isso, podemos ver que $\mathbb{Z}/n\mathbb{Z} \sim \mathbb{Z}_n$, pois temos uma bijeção natural entre os grupos, onde um elemento $a \in \mathbb{Z}_n$ é levado ao subgrupo lateral $n\mathbb{Z} + a$ e vice-versa. Além disso é importante notar que $n\mathbb{Z} + a$ é um subgrupo de $\mathbb{Z}$ apenas para $a = 0$, enquanto que para qualquer valor de $a \neq 0$, $n\mathbb{Z} + a$ não representa um subgrupo de $\mathbb{Z}$.

2.2.3 Homomorfismos

Definição 2.2.22. O mapa $f : G \to H$, que leva elementos do grupo G à elementos do grupo H é denominado *homomorfismo* de G em H, se f preserva a operação do grupo G. Ou seja, Se $*$ e . são as operações de G e H respectivamente, então dizemos que f preserva a operação de G se para qualquer $a, b \in G$, então $f(a * b) = f(a).f(b)$. Se f puder ser representada por uma função racional, então f é denominado *endomorfismo*. Se f for um mapa bijetor, então f é denominado *isomorfismo*. Se f for um mapa bijetor de G em G, então f é denominado *automorfismo*.

O símbolo '.' será utilizado para denotar a operação de grupos multiplicativos, de maneira que dados dois grupos multiplicativos G e H, este símbolo será utilizado para representar a operação de ambos os grupos, desde que não haja ambiguidade. Da mesma forma, o símbolo '+' será utilizado para denotar a operação de qualquer grupo aditivo.

Teorema 2.2.23. Seja $f : G \to H$ um homomorfismo entre os grupos G e H. Se $e \in G$ representa o elemento neutro de G, então $f(e)$ representa o elemento neutro de H.

Prova. De acordo com a definição de homomorfismos, temos que $e.e = e$, de modo que $f(e).f(e) = f(e)$. Logo, $f(e)$ é o elemento neutro de H. □

Teorema 2.2.24. Seja $f : G \to H$ um homomorfismo entre os grupos G e H. Então, f leva inversos de G em inversos de H. Isto é, para qualquer $a \in G$, temos que $f(a^{-1}) = (f(a))^{-1}$.

Prova. Para qualquer $a \in G$, temos que $a.a^{-1} = e$. Assim, $f(a)f(a^{-1}) = f(e)$. Logo, $f(a^{-1}) = (f(a))^{-1}$. □

Seja A o conjunto de automorfismos de um grupo G. O conjunto A forma um grupo abeliano com relação a operação de composição usual de mapas.

Exemplo 2.2.25. Um exemplo importante de automorfismo é o *automorfismo interno* $f_a : G \to G$, tal que, para $a \in G$, então $f_a(x) = axa^{-1}$. Os elementos x e axa^{-1} são chamados de *conjugados*. Dado um subgrupo S de G, o conjunto $aSa^{-1} = \{asa^{-1} | s \in S\}$, para $a \in G$, é denominado *conjugado* de S.

Definição 2.2.26. Seja $f : G \to H$ um homomorfismo de G em H. O conjunto $N = \{a | f(a) = e'\}$, onde e' representa o elemento neutro de H, é denominado *núcleo* de f, sendo denotado por $\text{nuc}(f)$.

Teorema 2.2.27. Seja $f : G \to H$ um homomorfismo de G em H. Então $\text{nuc}(f)$ é um subgrupo de G.

Prova. De acordo com a definição 2.2.10, é fácil ver que $\text{nuc}(f)$ é um subgrupo de G, pois para quaisquer elementos $a, b \in \text{nuc}(f)$, temos que $f(a) = e'$ e $f(b) = e'$ e portanto $f(a.b) = f(a).f(b) = e'.e' = e'$. Logo, $a.b$ pertence a $\text{nuc}(f)$. Além disso, para qualquer $a \in \text{nuc}(f)$, temos que $f(a) = e'$. Assim, $f(e) = f(a.a^{-1}) = f(a).f(a^{-1}) = e'$. Portanto, $e'.f(a^{-1}) = e'$. Logo, $f(a^{-1}) = e'$, de modo que $a^{-1} \in \text{nuc}(F)$. □

Exemplo 2.2.28. Seja $f : \mathbb{Z} \to \mathbb{Z}_n$, dado por $f(a) = a \pmod n$. O núcleo $\text{nuc}(f)$ é formado por todos os números inteiros tais que $a \equiv 0 \pmod n$. Desta forma, $\text{nuc}(f)$ é formado por todos os múltiplos de n. De acordo com o exemplo 2.2.12, o conjunto dos múltiplos de n, denotado por $n\mathbb{Z}$, é subgrupo de $\mathbb{Z}$.

Definição 2.2.29. Seja H um subgrupo de G, então H é denominado *subgrupo normal* de G, se para qualquer $h \in H$ e qualquer $a \in G$, então $aha^{-1} \in H$.

Teorema 2.2.30. Seja H um subgrupo de G. H é normal se e somente se H é igual aos seus conjugados. Equivalentemente, H é normal se e somente se H é invariante em relação a todos os automorfismos internos de G.

Teorema 2.2.31. Seja H um subgrupo de G. H é normal se e somente se toda classe lateral a esquerda aH for igual a respectiva classe lateral a direita Ha, para todo $a \in G$.

Teorema 2.2.32. Seja $f : G \to H$ um homomorfismo de G em H. Então o núcleo $\text{nuc}(f)$ é um subgrupo normal de G. Além disso, H é isomorfo ao grupo quociente $G/\text{nuc}(f)$. Por outro lado, se N é um subgrupo normal de G, então o mapa $g : G \to G/N$, definido por $g(a) = aN$, para $a \in G$, é um homomorfismo de G sobre G/N com núcleo $\text{nuc}(g) = N$.

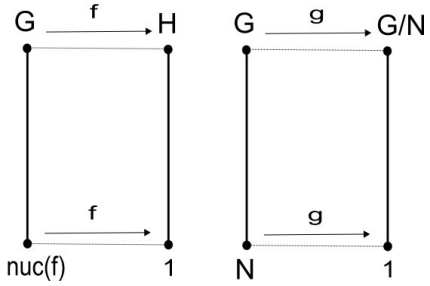


Figura 2.2.1: Homomorfismos em grupos

Teorema 2.2.33. Grupos cíclicos de mesma ordem são isomorfos. Em particular, todo grupo cíclico de ordem n é isomorfo a $(\mathbb{Z}_n, +)$.

2.3 Composição de Grupos

Podemos criar grupos através da operação $\oplus$, que permite compor grupos G_1 e G_2, de modo que $G_1 \oplus G_2$ representa um grupo dado por elementos da forma (a, b), tal que $a \in G_1$ e $b \in G_2$. A soma de elementos desse grupo é realizada utilizando a soma dos grupos G_1 e G_2, ou seja, $(a_1, b_1) + (a_2, b_2) = (a_1 + a_2, b_1 + b_2)$.

A figura a seguir ilustra a estrutura de $\mathbb{Z}_3 \oplus \mathbb{Z}_3$:

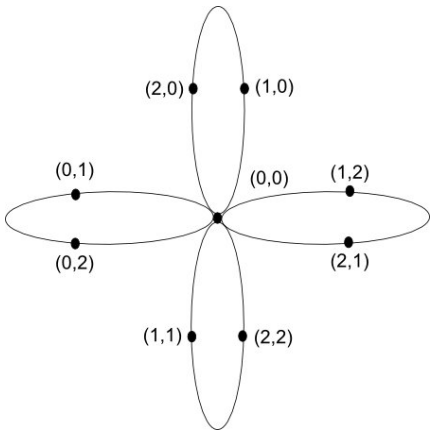


Figura 2.3.2: Estrutura de $\mathbb{Z}_3 \oplus \mathbb{Z}_3$

Como é possível observar na figura 2.4.3, cada uma das 4 pétalas representa um subgrupo cíclico de ordem 3. Todos os subgrupos compartilham o elemento neutro $(0, 0)$. Da mesma forma, $\mathbb{Z}_n \oplus \mathbb{Z}_n$, é constituído de $n + 1$ subgrupos de n elementos e com o elemento neutro $(0, 0)$ comum a todos. Em geral, cada pétala representa um subgrupo cíclico. Dados dois grupos G_1 e G_2, a composição $G_1 \oplus G_2$ pode ser encarada como uma estrutura de duas dimensões, de forma que um elemento genérico de $G_1 \oplus G_2$ pode ser obtido como combinação de elementos de apenas duas pétalas. Isto é, dado um gerador x_1 de uma pétala e um gerador x_2 de uma pétala distinta, todo elemento de $G_1 \oplus G_2$ pode ser obtido como uma combinação da forma $kx_1 + \ell x_2$, onde k e ℓ são números inteiros.

2.4 Anel

Nesta seção vamos ver as principais definições e teoremas a respeito de uma estrutura algébrica denominada anel, que resumidamente é um conjunto que tem definidas duas operações em seus elementos. Normalmente estas duas operações são a soma e a multiplicação, sendo que para ser um anel, uma das operações precisa constituir um grupo abeliano.

2.4.1 Definições Preliminares

Definição 2.4.1. Um anel é definido como sendo um trio $(G, +, .)$, onde G é um conjunto, $(G, +)$ é um grupo abeliano e '.' é uma operação binária definida entre elementos de G, tais que

1. para quaisquer $a, b \in G$, então $a.b \in G$;

2. para quaisquer $a, b, c \in G$, então $a.(b+c) = a.b + a.c$, esta propriedade é responsável por relacionar a operação de soma com a operação de multiplicação, sendo denominada *propriedade distributiva*.

Definição 2.4.2. Um anel onde $a.(b.c) = (a.b).c$, para quaisquer $a, b, c \in G$ é denominado *anel associativo*.

Definição 2.4.3. Um anel onde existe um elemento $e \in G$, tal que $a.e = e.a = a$, para qualquer $a \in G$, é denominado *anel com elemento neutro*.

Definição 2.4.4. Um anel onde $a.b = b.a$, para quaisquer elementos $a, b \in G$, é denominado *anel comutativo*.

Exemplo 2.4.5. O conjunto dos números racionais $\mathbb{Q}$, juntamente com as operações usuais de soma e multiplicação, forma um anel comutativo, tal que, para um elemento arbitrário $a \in \mathbb{Q}$, seu inverso aditivo é $-a \in \mathbb{Q}$ e seu inverso multiplicativo é $1/a \in \mathbb{Q}$.

Exemplo 2.4.6. Dado um anel R, podemos construir um exemplo de anel não-comutativo através de matrizes quadradas de tamanho $n \times n$, composta de elementos $a_{ij} \in R$, com as operações de soma e multiplicação usuais de matrizes.

Exemplo 2.4.7. O conjunto $\mathbb{Z}_p$, para p primo, com as operações usuais de soma e multiplicação modulares, forma um anel, cujo elemento neutro da soma é o 0 e o elemento neutro da multiplicação é o 1.

Definição 2.4.8. Seja R um anel. Um subconjunto S de R é um subanel de R se o próprio S for um anel com relação as operações de soma e multiplicação definidas em R.

Exemplo 2.4.9. Seja $\mathbb{Z}$ o anel dos números inteiros. Então $2\mathbb{Z} = \{..., -4, -2, 0, 2, 4, ...\}$ é um subanel de $\mathbb{Z}$, pois, conforme o exemplo 2.2.11, $2\mathbb{Z}$ é um grupo abeliano com relação a soma. Além disso, dados dois elementos $a, b \in 2\mathbb{Z}$, então $a.b$ é par e portanto pertence a $2\mathbb{Z}$. Por último, a propriedade distributiva vem automaticamente do anel $\mathbb{Z}$.

Definição 2.4.10. Dado um anel R, um *divisor do zero* é um elemento $a \in R$, tal que existe $b \in R$, $b \neq 0$, de modo que $a.b = 0$.

Definição 2.4.11. Seja R um anel comutativo. Este anel é denominado um *domínio de integridade*, se R não possui nenhum divisor do zero.

Definição 2.4.12. A *característica* de um anel é definida como sendo o menor inteiro n, tal que

$$\sum_{1}^{n} e = 0,$$

onde e é o elemento neutro com relação a multiplicação. Se não existir um valor de n com essa propriedade, então dizemos que o anel tem característica 0.

Exemplo 2.4.13. Seja $\mathbb{Z}_p$ o corpo do exemplo 2.4.7. Então, o menor n tal que $n.1 = 0 \pmod{p}$, é o próprio p. Portanto, $\mathbb{Z}_p$ tem característica p.

Exemplo 2.4.14. Seja $\mathbb{Q}$ o corpo definido no exemplo 2.4.5. Sabemos que não existe valor de n tal que $n.1 = 0$. Desta forma, a característica de $\mathbb{Q}$ é 0.

2.4.2 Ideais

Definição 2.4.15. Dado um anel R, um subconjunto I de R é denominado *ideal direito* se I corresponde a um subgrupo de R com relação a soma, e para quaisquer $x \in I$ e $r \in R$, então $xr \in I$.

Definição 2.4.16. Dado um anel R, um subconjunto I de R é denominado *ideal esquerdo* se I corresponde a um subgrupo de R com relação a soma, e para quaisquer $x \in I$ e $r \in R$, então $rx \in I$.

Se R for um anel comutativo, então todo ideal direito é igual ao ideal esquerdo correspondente, sendo denominado apenas *ideal*. Um *ideal próprio* é aquele que é distinto do anel subjacente.

Definição 2.4.17. Um ideal I é denominado *ideal primo* se para $ab \in I$ e $a, b \in R$, então ou $a \in I$ ou $b \in I$.

Definição 2.4.18. Seja R um anel comutativo. Um ideal I de R é denominado *ideal principal* se existir $a \in R$, tal que $I = (a)$, ou seja, é o *ideal gerado* por a, através da multiplicação de todo elemento de R por a.

Definição 2.4.19. Seja R um anel comutativo. Um ideal I de R é denominado *ideal maximal* se não existir um ideal J de R, tal que I seja um subconjunto próprio de J.

Definição 2.4.20. Seja R um anel comutativo. R é denominado *domínio de ideal principal* se existir um ideal I de R, tal que I seja um ideal principal. Em outras palavras, R é um anel comutativo que pode gerado através de um elemento $a \in R$, relação que denotamos por $R = (a)$.

Exemplo 2.4.21. Seja Q um anel como descrito nos exemplos acima, então $3Q$ representa o ideal esquerdo gerado multiplicando cada elemento de Q por 3. Como Q é um anel comutativo, $3Q = Q3$. Além disso, $3Q$ é um ideal primo. É fácil ver que pQ é um ideal primo, se e somente se p é primo, pois para $n = ab$, nQ possui elementos da forma kab para $k \in Q$, e nem $ka \in nQ$, nem $b \in nQ$.

2.4.3 Polinômios

Nesta seção serão estudadas as principais definições e teoremas sobre polinômios definidos sobre anéis. O estudo de polinômios tem papel fundamental para a implementação de algoritmos criptográficos, tendo em vista que a representação polinomial para as estruturas algébricas envolvidas no cálculo de emparelhamentos bilineares permite uma série de otimizações.

Definição 2.4.22. Um polinômio é uma expressão da forma $f(x) = a_0 + a_1x + ... + a_nx^n$, onde $a_n \neq 0$. Deste modo, um polinômio é uma função na variável x, onde os valores de a_i, para $0 \leq i \leq n$, são denominados *coeficientes* do polinômio. Com isso, um polinômio é determinado univocamente a partir de seus coeficientes. O *grau* de um polinômio f é um número inteiro n tal que $a_i = 0$, para $i > n$. Denotamos o grau de um polinômio f por grau(f).

Definição 2.4.23. Dado um anel R, podemos construir o *anel polinomial*, denotado por $R[x]$, com coeficientes pertencentes a R, ou seja,

$$R[x] = \{a_nx^n + ... + a_1x^1 + a_0 | a_i \in R\}.$$

Dados $f(x) = \sum_{i}^{n} a_ix^i$ e $g(x) = \sum_{i}^{m} b_ix^i$, a operação de soma neste anel polinomial é definida da seguinte forma:

$$f(x) + g(x) = \sum_{i=0}^{max\{n,m\}} (a_i + b_i)x^i,$$

onde $max\{a, b\}$ é definido como sendo o valor máximo entre a e b.

O produto entre $f(x)$ e $g(x)$ é definido da seguinte forma:

$$f(x).g(x) = \sum_{k=0}^{n+m} c_kx^k,$$

onde $c_k = \sum_{i+j=k}^{i<n,j<m} a_ib_j$.

Dado um elemento $\alpha \in R$, pode-se substituir a variável x por α, de maneira que o polinômio $f(x)$ pode ser avaliado no elemento α; ou seja, $f(\alpha) = a_0 + a_1(\alpha) + ... + a_n(\alpha)^n$. Como $\alpha \in R$, temos que $f(\alpha) \in R$.

Definição 2.4.24. Seja R um anel e $R[x]$ o anel polinomial correspondente. Seja $f(x) \in R[x]$. Então, um elemento $r \in R$ é denominado *raíz* de $f(x)$, se $f(r) = 0$, onde 0 representa o elemento neutro de R com relação a soma.

Definição 2.4.25. Seja R um anel e $R[x]$ o anel polinomial correspondente. Dado um polinômio $f(x) \in R[x]$, o *conjunto dos zeros* de $f(x)$ é definido como sendo $\mathbb{Z} = \{r \in R | f(r) = 0\}$.

Definição 2.4.26. Dado um polinômio $f(x) = a_0 + ... + a_n x^n$, o elemento a_n é denominado *coeficiente líder*. Este elemento é tal que n é o maior valor de i para o qual $a_i \neq 0$. O elemento a_0 é denominado *termo constante* de $f(x)$. Além disso, $f(x)$ é denominado *polinômio mônico* se o coeficiente líder for 1.

Teorema 2.4.27. Sejam $f, g \in R[x]$. Então, $\text{grau}(f + g) \leq max\{\text{grau}(f), \text{grau}(g)\}$ e $\text{grau}(f.g) \leq \text{grau}(f) + \text{grau}(g)$.

Se R for um domínio de integridade, então temos a igualdade $\text{grau}(f.g) = \text{grau}(f) + \text{grau}(g)$.

Se interpretarmos os termos constantes de $R[x]$ como elementos de R, então R é um subanel de $R[x]$.

Teorema 2.4.28. Seja R um anel, então

(i) $R[x]$ é comutativo se e somente se R for comutativo.

(ii) $R[x]$ é um anel com identidade se e somente se R for um anel com identidade, caso em que a identidade de $R[x]$ é a própria identidade de R.

(iii) $R[x]$ é um domínio de integridade se e somente se R for um domínio de integridade.

Definição 2.4.29. Um *Anel Euclidiano* é um anel R com pelo menos dois elementos, sem nenhum divisor do zero, tal que existe um mapa γ que leva elementos não nulos de R em inteiros não negativos, satisfazendo as seguintes propriedades:

(i) Se $a, b \in R$ tal que $a.b \neq 0$, então $\gamma(a.b) \geq \gamma(a)$.

(ii) Para quaisquer $a, b \in R$, $b \neq 0$, então existe elementos $q, r \in R$, tal que $a = b.q + r$ e $\gamma(r) \leq \gamma(b)$.

Aneís Euclidianos introduzem o conceito de divisibilidade em anéis, pois desta forma é possível generalizar a idéia de divisão nos números reais para divisão de elementos de uma estrutura algébrica abstrata. Se considerarmos o conjunto dos números reais $\mathbb{R}$, então o mapa $\gamma(x)$, para $x \in \mathbb{R}$, corresponde ao valor absoluto x, isto é, $\gamma(x) = |x|$.

Teorema 2.4.30. Todo anel polinomial é um anel euclidiano.

O teorema 2.4.30 mostra que o conceito de divisão pode ser utilizado também para polinômios, onde a função γ corresponde ao grau do polinômio em questão, ou seja, dado um anel polinomial $R[x]$, então $\gamma(f) = \text{grau}(f)$, para $f \in R[x]$.

Teorema 2.4.31. Seja R um domínio de integridade e $R[x]$ o anel polinomial correspondente. Então, $R[x]$ é um domínio de ideal principal. De fato, para todo ideal $J \neq (0)$ de $R[x]$, existe um único polinômio mônico $h \in R[x]$, tal que $J = (h)$.

Prova. De acordo com o teorema 2.4.28 (iii), $R[x]$ é um domínio de integridade. Seja $J \neq (0)$ um ideal de $R[x]$. Seja $h(x)$ um polinômio não nulo de grau mínimo em $R[x]$. Para facilitar a demonstração, suponha que $h(x)$ seja um polinômio mônico (caso contrário basta considerar um polinômio $g(x) = h(x)/a_n$, onde a_n é o coeficiente líder de $h(x)$). Seja f um polinômio arbitrário de J. Dividindo f por h, obtemos os polinômios q e r, tais que $f = hq + r$, onde $\text{grau}(r) < \text{grau}(h)$. Portanto, $r = f - hq$ pertence a J, mas como h é minimal, temos que $r = 0$ e com isso, f é divisível por h. Logo, $J = (h)$. Para mostrar que h é único, basta considerar a hipótese de haver outro polinômio $h' \in J$, tal que $J = (h')$, neste caso, é fácil notar que h' é um múltiplipo de h por um termo constante $c \neq 1$, ou seja, h' não é um polinômio mônico. □

Teorema 2.4.32. Sejam $f_1, f_2, ..., f_n$ polinômios em $R[x]$, com pelo menos um deles não nulo. Então, existe um polinômio $d \in R[x]$, tal que d divide cada f_i, para $1 \leq i \leq n$ e d é de grau máximo dentre os polinômios com essa propriedade, ou seja, se c divide cada f_i, para $1 \leq i \leq n$ e $c \in R[x]$, então c divide d. Além disso, d pode ser expresso na forma $d = b_1 f_1 + ... + b_n f_n$, onde $b_i \in R[x]$, para $1 \leq i \leq n$.

Definição 2.4.33. Um polinômio $p \in R[x]$ é denominado *irredutível sobre* R se possui grau positivo e $p = bc$, para $b, c \in R[x]$, implica que b ou c é um polinômio constante.

Teorema 2.4.34. Se o polinômio irredutível $p \in R[x]$ divide um produto na forma $f_1...f_n$, de polinômios em $R[x]$, então p divide pelo menos um dos fatores f_i, para $1 \leq i \leq n$.

Teorema 2.4.35. Um polinômio mônico $f \in R[x]$ de grau positivo pode ser escrito na forma

$$f = p_1^{e_1} p_2^{e_2} ... p_k^{e_k},$$

onde p_i, para $1 \leq i \leq k$, são polinômios irredutíveis em $R[x]$ e e_i, para $1 \leq i \leq k$, são inteiros positivos. Além disso, esta fatoração é única.

O teorema 2.4.32 mostra que, assim como nos números inteiros, os polinômios possuem uma estrutura algébrica que permite encontrar um elemento que é o máximo divisor comum de um conjunto de outros polinômios. Os teoremas 2.4.32, 2.4.34 e 2.4.35 são válidos

também para qualquer outro tipo de anel euclidiano. Ou seja, é possível fazer uma analogia entre polinômios e os números inteiros, onde polinômios irredutíveis correspondem a número primos. De fato, o termo *polinômio primo* pode ser utilizado no lugar de polinômio irredutível. Da mesma forma, a decomposição usual de um número inteiro em fatores primos é análoga à decomposição de um polinômio em fatores irredutíveis.

Teorema 2.4.36. Um elemento $b \in R$ é uma raíz de um polinômio $f \in R[x]$ se e somente se o polinômio $(x - b)$ divide f.

Prova. Dividindo f por $(x-b)$, temos que $f(x) = q(x)(x-b)+r(x)$, onde $q(x) \in R[x]$ é o polinômio quociente e $r(x)$ é o resto da divisão, e grau$(r) <$ grau$(x - b)$, de forma que $r \in R$. Substituindo x por b, obtemos $f(b) = q(b)(b - b) + r$. Como b é raíz de $f(x)$, então $f(b) = 0$ e temos que $r = 0$. □

Definição 2.4.37. Seja $b \in R$ uma raíz de um polinômio $f \in R[x]$ com grau positivo. Se k é maior inteiro positivo tal que $(x - b)^k$ divide f, então k é denominado *multiplicidade* de b para o polinômio f.

Definição 2.4.38. Seja $f \in R[x]$, $f(x) = a_0+...+a_nx^n$. Então, a *derivada* de f, denotada f' é definida como sendo $f'(x) = a_1 + 2a_2x + ... + na_nx^{n-1}$.

Teorema 2.4.39. Um elemento $b \in R$ é uma raíz de multiplicidade maior que 1 se e somente se for raíz tanto de f como de f', a derivada de f.

Teorema 2.4.40. (Teorema de interpolação de Lagrange) Para $n \geq 0$, sejam $a_0, ..., a_n$ elementos distintos de R, e $b_0, ..., b_n$ elementos arbitrários de R. Então, existe exatamente um polinômio $f \in R[x]$ de grau menor ou igual a n, tal que $f(a_i) = b_i$, para $0 \leq i \leq n$. Este polinômio é dado por

$$f(x) = \sum_{i=0}^{n} b_i \prod_{k=0,k\neq i}^{n} (a_i - a_k)^{-1}(x - a_k).$$

2.4.4 Homomorfismos

É possível agora estender a definição de homomorfismos de grupos para homomorfismos em anéis:

Definição 2.4.41. Seja o mapa $\psi : R \to S$, entre os anéis R e S, então ψ é um homomorfismo se para quaisquer $a, b \in R$, temos que $\psi(a + b) = \psi(a) + \psi(b)$ e $\psi(a.b) = \psi(a).\psi(b)$.

O núcleo de homomorfismo em anéis também é definido como sendo o conjunto nuc$(\psi) = \{a \in R|\psi(a) = 0\}$. Ou seja, são os elementos $a \in R$ que são mapeados no elemento neutro com relação a soma em S.

Teorema 2.4.42. Seja $\psi : R \to S$ um homomorfismo de R sobre S. Então $\text{nuc}(\psi)$ é um ideal de R e S é isomorfo ao anel quociente $R/\text{nuc}(\psi)$. Por outro lado, se J é um ideal de R, então o mapa $\psi : R \to R/J$, definido por $\psi(a) = a + J$, para $a \in R$, é um homomorfismo cujo núcleo é J.

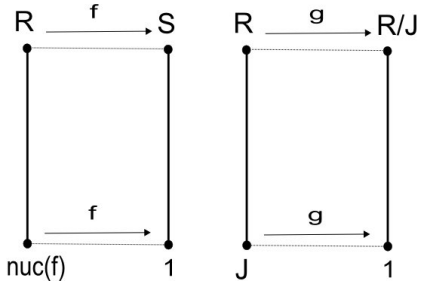


Figura 2.4.3: Homomorfismos em anéis

Teorema 2.4.43. Seja R um anel comutativo de característica prima p. Então

$$(a+b)^{p^n} = a^{p^n} + b^{p^n},$$

para $a, b \in R$ e n inteiro positivo.

Prova. Utilizando a expansão binomial de $(a+b)^p$, temos que

$$(a+b)^p = a^p + \binom{p}{1} a^{p-1}b + ... + \binom{p}{p-1} ab^{p-1} + b^p.$$

Mas sabemos que

$$\binom{p}{i} \equiv 0 \pmod{p},$$

para $0 < i < p$, de modo que $(a+b)^p \equiv a^p + b^p \pmod{p}$. Com isso, utilizando o princípio de indução finita em n, obtemos o resultado desejado. □

Teorema 2.4.44. Seja R um anel comutativo de característica prima p. Então

$$(a-b)^{p^n} = a^{p^n} - b^{p^n},$$

para $a, b \in R$ e n inteiro positivo.

Prova. A demonstração deste teorema é análoga à prova do teorema 2.4.43.

Teorema 2.4.45. Seja R um anel comutativo com identidade. Então

(i) Um ideal I de R é maximal se e somente se R/I for um corpo.

(ii) Um ideal I de R é um ideal primo se e somente se R/I for um domínio de integridade.

(iii) Todo ideal maximal é um ideal primo.

(iv) Se R é um domínio de ideal principal, então $R/(c)$ é um corpo se e somente se c for um elemento primo de R.

2.5 Corpos

Nesta seção, serão estudadas estruturas que correspondem a uma restrição da definição de anel. Basicamente, um corpo é um anel em que a segunda operação também forma um grupo abeliano com relação aos elementos não nulos. Esta restrição permite que uma série de propriedades sejam válidas, especialmente para corpos com um número finito de elementos, onde a estrutura de subcorpos e homomorfismos são bem conhecidas.

2.5.1 Definições Preliminares

Definição 2.5.1. Um corpo é definido como sendo um trio $(\mathbb{F}, +, .)$, onde $\mathbb{F}$ é um conjunto, e além disso, temos que $(\mathbb{F}, +)$ e $(\mathbb{F}^*, .)$ são grupos abelianos.

Exemplo 2.5.2. Seja $\mathbb{Z}_p = \{0, 1, ..., p-1\}$. Este conjunto, com as operações usuais de soma e multiplicação modulares, forma um corpo, cujo elemento neutro da soma é o 0 e o elemento neutro da multiplicação é o 1. Este corpo normalmente é denotado por $\mathbb{F}_p$ e é conhecido como *corpo de Galois de ordem p*.

Exemplo 2.5.3. O conjunto $\mathbb{Q}$ dos números racionais, com as operações usuais de soma e multiplicação, forma um corpo, cujo elemento neutro da soma é o 0 e o elemento neutro da multiplicação é o 1.

Com isso, podemos ver que um corpo é um anel associativo, com elemento neutro, comutativo, onde para cada elemento $a \in \mathbb{F}$, existe o inverso de a com relação a multiplicação, denotado por a^{-1}, tal que $(a).(a^{-1}) = (a^{-1}).(a) = e$, onde e representa o elemento neutro com relação a multiplicação.

Definição 2.5.4. Seja $\mathbb{K}$ um corpo. Um subconjunto L de $\mathbb{K}$ é denominado *subcorpo* se ele próprio for um corpo com relação às operações definidas em $\mathbb{K}$. Por sua vez, $\mathbb{K}$ é denominado *corpo de extensão* em relação a L.

Definição 2.5.5. Seja L uma extensão do corpo finito $\mathbb{K}$. Se L, considerado como um espaço vetorial sobre $\mathbb{K}$, tiver dimensão finita, então L é denominada *extensão finita* de $\mathbb{K}$. A dimensão do espaço vetorial de L sobre $\mathbb{K}$ é denominada *grau de extensão de L sobre $\mathbb{K}$*, denotada por $[L : \mathbb{K}]$.

Definição 2.5.6. Um corpo que não contenha nenhum subcorpo próprio é denominado *corpo primo*.

Definição 2.5.7. Um corpo onde o conjunto $\mathbb{F}$ possui um número finito de elementos é denominado *corpo finito*.

Teorema 2.5.8. Todo domínio de integridade finito é um corpo.

Prova. Seja R um domínio de integridade finito, cujos elementos são $a_1, a_2, ..., a_n$. Dado $a \in R$, $a \neq 0$, então os elementos $aa_1, aa_2, ..., aa_n$ são distintos dois a dois, pois caso contrário teríamos $1 \leq i, j \leq n$, $i \neq j$, tal que $aa_i = aa_j$. Então, $a(a_i - a_j) = 0$. Mas como não existem divisores do zero em R e $a \neq 0$, então $a_i = a_j$. Logo, todo elemento de R é da forma aa_i, para $1 \leq i \leq n$. Em especial, temos que $e = aa_i$, para algum i, onde e é o elemento neutro multiplicativo. Como R é um anel comutativo, sabemos que $a_i a = e$, e assim a_i é o inverso de a. Logo, os elementos não nulos de R formam um grupo comutativo, de forma que R é um corpo. □

Teorema 2.5.9. Seja $f \in R[x]$. O anel quociente $R[x]/(f)$, onde (f) é o ideal gerado por f, é um corpo se e somente se f for um polinômio irredutível em $R[x]$.

Teorema 2.5.10. Seja $\mathbb{K}$ um corpo finito. Então, $\mathbb{K}$ possui característica p, onde p é um inteiro primo.

Prova. Seja $\mathbb{K}$ um corpo finito com elemento neutro da soma dado por 0 e elemento neutro da multiplicação dado por 1. Primeiramente, vamos mostrar que a característica de $\mathbb{K}$ é maior que 0. Para isso, basta considerar os elementos $e, 2e, 3e, ...$ Como $\mathbb{K}$ possui um números finito de elementos, existe $1 \leq k < m$, tal que $ke = me$, e portanto temos que $(m - k)e = 0$. Logo, $\mathbb{K}$ possui característica positiva. Agora suponha que a característica de $\mathbb{K}$ seja $n = k.\ell$, para $1 < k, \ell, < n$. Então, $n.1 = 0$. Com isso, temos que $(k.\ell).1 = 0$, e portanto $(k.1).(\ell.1) = 0$. Mas $\mathbb{K}$ não possui divisores do zero, de modo que ou $k.1 = 0$, ou $\ell.1 = 0$, contrariando a minimalidade de n. □

Teorema 2.5.11. Seja $\mathbb{F}$ um corpo finito e $\mathbb{K}$ um subcorpo contendo q elementos. Então $\mathbb{F}$ tem q^m elementos, onde $m = [\mathbb{F} : \mathbb{K}]$.

Prova. De acordo com a definição 2.5.5, $\mathbb{F}$ é um espaço vetorial sobre $\mathbb{K}$. A dimensão deste espaço vetorial é $[\mathbb{F} : \mathbb{K}] = m$. Então $\mathbb{F}$ possui uma base sobre $\mathbb{F}$, constituída de

m elementos $b_1, b_2, ..., b_m$. Assim, todo elemento de $\mathbb{F}$ pode ser representado por uma expressão na forma $a_1b_1 + ... + a_mb_m$, onde $a_i \in \mathbb{K}$. Como existem q valores para cada a_i, então $\mathbb{F}$ possui exatamente q^m elementos. $\square$

Teorema 2.5.12. Seja $\mathbb{F}$ um corpo finito. Então $\mathbb{F}$ contém p^m elementos, onde p é a característica prima de $\mathbb{F}$ e m é o grau de $\mathbb{F}$ sobre seu subcorpo primo.

Prova. De acordo com o teorema 2.5.10, $\mathbb{F}$ possui característica prima, portanto possui um subcorpo primo $\mathbb{K}$. Logo, pelo teorema 2.5.11, $\mathbb{F}$ possui p^m elementos, onde $m = [\mathbb{F} : \mathbb{K}]$.

Com isso, todo corpo finito $(\mathbb{F}, +, .)$, denotado apenas por $\mathbb{F}_q$, possui uma quantidade de elementos $q = p^m$, onde p é primo. Podemos distinguir três tipos de corpos, de acordo com os valores de p e m. Se $p > 2$ e $m = 1$, $\mathbb{F}_p$ é denominado *corpo primo*. Se $p = 2$, $\mathbb{F}_{2^m}$ é denominado *corpo binário*. Se $p > 2$ e $m > 1$, $\mathbb{F}_{p^m}$ é denominado *corpo de extensão*.

Dado um corpo $\mathbb{F}_{p^m}$, uma *extensão* deste corpo é qualquer corpo da forma $\mathbb{F}_{p^{km}}$, para k inteiro positivo.

Uma questão importante a ser resolvida é sobre a representação de elementos de um corpo e a definição das operações envolvidas. No caso de corpos primos, podemos utilizar o conjunto $\mathbb{F}_p = \{0, 1, ..., p-1\}$, com as operações usuais de soma e multiplicação modulares.

Uma representação padrão para elementos de um corpo $\mathbb{F}_{p^m}$ é através de polinômios da forma:

$$p(x) = a_{m-1}x^{m-1} + ... + a_0,$$

onde $a_i \in \mathbb{F}_p$. Além disso, é preciso estabelecer um *polinômio irredutível* de grau m, isto é, um polinômio que não pode ser fatorado como a multiplicação de polinômios de grau menor que m, com coeficientes em $\mathbb{F}_p$.

A soma de elementos desse corpo é definida através da soma de polinômios, com redução modular dos coeficientes, ou seja, seja $p_1(x) = a_{m-1}x^{m-1} + ... + a_0$ e $p_2(x) = b_{m-1}x^{m-1} + ... + b_0$, então

$$p_1(x) + p_2(x) = c_{m-1}x^{m-1} + ... + c_0,$$

onde $c_i \equiv a_i + b_i \pmod{p}$.

Já a multiplicação de elementos desse corpo, pode ser realizada através da multiplicação usual de polinômios, seguida de uma redução módulo o polinômio irredútivel previamente escolhido.

2.5.2 Fecho algébrico

Um elemento $a \in \mathbb{F}_{p^{km}}$ é denominado *algébrico* sobre $\mathbb{F}_{p^m}$, se existir um polinômio $p[x]$ com coeficientes em $\mathbb{F}_{p^m}$ tal que $p[a] = 0$, ou seja, elementos algébricos são aqueles que são raízes de polinômios.

Em geral, dado um corpo $\mathbb{K}$ e um anel polinomial $\mathbb{K}[x]$, nem todas as raízes de $\mathbb{K}[x]$ pertencem a $\mathbb{K}$, mas existe uma extensão de $\mathbb{K}$ que contém todas essas possíveis raízes, esta extensão é denominada *fecho algébrico* de $\mathbb{K}$ e é denotada por $\overline{\mathbb{K}}$.

Por exemplo, considerando os números reais $\mathbb{R}$, o polinômios $p(x) = x^2 + 1$ não possui raízes reais, mas se considerarmos os números complexos $\mathbb{C}$, $p(x) = (x-i)(x+i)$, portanto i é algébrico sobre $\mathbb{R}$. De fato, $\mathbb{C}$ é o fecho algébrico de $\mathbb{R}$, pois toda raíz de um polinômio com coeficientes reais está contida em $\mathbb{C}$.

Definição 2.5.13. Se $\theta \in \mathbb{F}$ é algébrico sobre $\mathbb{K}$, então o polinômio mônico $g \in \mathbb{K}[x]$, que gera o ideal $J = \{f \in \mathbb{K}[x] | f(\theta) = 0\}$ é denominado *polinômio minimal* de θ sobre $\mathbb{K}$. O grau de θ sobre $\mathbb{K}$ é o grau de g.

Teorema 2.5.14. O polinômio g, da definição 2.5.13 é único. Isto é, dado $\theta \in \mathbb{F}$, algébrico sobre $\mathbb{K}$, então existe um único polinômio minimal mônico g, tal que $g(\theta) = 0$.

Teorema 2.5.15. Seja $\theta \in \mathbb{F}$, algébrico sobre $\mathbb{K}$, então seu polinômio minimal g sobre $\mathbb{K}$ tem as seguintes propriedades:

(i) g é irredutível sobre $\mathbb{K}[x]$.

(ii) Para $f \in \mathbb{K}[x]$, $f(\theta) = 0$ se e somente g divide f.

(iii) g é um polinômio mônico em $\mathbb{K}[x]$ de grau mínimo tal que θ é raiz.

Teorema 2.5.16. Toda extensão finita L de $\mathbb{K}$ é algébrica sobre $\mathbb{K}$. Isto é, Dada uma extensão finita L, todo elemento de L é raíz de um polinômio com coeficientes em K.

Prova. Seja $\theta \in L$, uma extensão finita de $\mathbb{K}$. Então, pelo teorema 2.5.11, $[L : \mathbb{K}] = m$, para algum inteiro positivo m. Portanto, os elementos $1, \theta, \theta^2, ..., \theta^m$ são linearmente dependentes sobre $\mathbb{K}$, de modo que existem valores de $a_0, a_1, ..., a_m \in \mathbb{K}$, não todos nulos, tais que $a_0 + a_1\theta + ... + a_m\theta^m = 0$. Logo, existe um polinômio em $\mathbb{K}[x]$ que θ seja raíz, ou seja, θ é algébrico sobre $\mathbb{K}$. □

Teorema 2.5.17. Seja $\theta \in \mathbb{F}$, algébrico de grau n sobre $\mathbb{K}$ e seja g o polinômio minimal de θ sobre $\mathbb{K}$. Então:

(i) $\mathbb{K}(\theta)$ é isomorfo a $\mathbb{K}[x]/(g)$.

(ii) $[\mathbb{K}(\theta) : \mathbb{K}] = n$ e $\{1, \theta, ..., \theta^{n-1}\}$ é uma base de $\mathbb{K}(\theta)$ sobre $\mathbb{K}$.

(iii) Para qualquer $\alpha \in \mathbb{K}(\theta)$, α é algébrico sobre $\mathbb{K}$ e o grau de α sobre $\mathbb{K}$ é um divisor de n.

Definição 2.5.18. Seja $\mathbb{K}$ um corpo finito. Seja $\alpha \notin \mathbb{K}$, algébrico sobre $\mathbb{K}$. Então $\mathbb{K}(\alpha)$ é uma extensão de $\mathbb{K}$. Este processo de construção de extensões é denominado *adjunção de α sobre $\mathbb{K}$*.

Teorema 2.5.19. Seja $f \in \mathbb{K}[x]$ um polinômio irredutível sobre $\mathbb{K}$. Então existe uma extensão algébrica de $\mathbb{K}$ gerada a partir da adjunção de uma raíz de f em $\mathbb{K}$. Seja α esta raíz de f, então $\mathbb{K}[x]/(f)$ é isomorfo a $\mathbb{K}(\alpha)$.

Teorema 2.5.20. Sejam α e β duas raízes de um polinômio $f \in \mathbb{K}[x]$, irredutível sobre $\mathbb{K}$. Então $\mathbb{K}(\alpha)$ e $\mathbb{K}(\beta)$ são isomorfos e o mapa deste isomorfismo é tal que α é levado em β e os elementos de $\mathbb{K}$ são fixados pelo mapa.

2.5.3 Caracterização de Corpos

Teorema 2.5.21. (Existência e Unicidade de Corpos de Decomposição). Se $\mathbb{K}$ é um corpo e f é um polinômio de grau positivo em $\mathbb{K}[x]$, então existe um corpo de decomposição de f sobre $\mathbb{K}$. Dois corpos de decomposição de f sobre $\mathbb{K}$ são isomorfos e o isomorfismo é dado por um mapa que mantém fixados os elementos de $\mathbb{K}$ e permuta as raízes de f.

Teorema 2.5.22. Seja $\mathbb{K}$ um corpo finito com q elementos, então para todo $a \in \mathbb{K}$, temos que $a^q = a$

Prova. Primeiramente, a identidade $a^q = a$ é trivialmente válida para $a = 0$. Portanto, seja $a \neq 0$, $a \in \mathbb{K}$. Como $\mathbb{K}$ forma um grupo multiplicativo de ordem $q-1$, então a ordem de a divide $q-1$ para qualquer $a \in \mathbb{K}$. Com isso, temos que $a^{q-1} = 1$. Logo, $a^q = a$. □

Teorema 2.5.23. Seja $\mathbb{F}$ um corpo finito com q elementos e $\mathbb{K}$ um subcorpo de $\mathbb{F}$. Então, o polinômio $x^q - x$ em $\mathbb{K}[x]$ fatora em $\mathbb{F}[x]$ como

$$x^q - x = \prod_{a \in \mathbb{F}[x]} (x - a)$$

e $\mathbb{F}$ é um *corpo de decomposição* de $x^q - x$ sobre $\mathbb{K}$.

Prova. O polinômio $x^q - x$ de grau q tem no máximo q raízes em $\mathbb{F}$. De acordo com o teorema 2.5.22 cada elemento de $\mathbb{F}$ é uma raíz do polinômio, de modo que sabemos exatamente quais são as q raízes. Assim, o polinômio $x^q - x$ pode ser decomposto desta forma em $\mathbb{F}$, mas não em nenhum outro corpo de tamanho menor.

Teorema 2.5.24. (Existência e Unicidadede corpos finitos) Para todo primo p e todo inteiro positivo m, existe um corpo finito com p^m elementos. Além disso, todo corpo finito com p^m elementos é isomorfo ao corpo de decomposição de $x^q - x$ sobre $\mathbb{F}_p$.

Prova. (Existência) Considere o polinômio $x^q - x$ sobre $\mathbb{F}_p[x]$, para $q = p^m$. Seja $\mathbb{K}$ corpo de decomposição deste polinômio sobre $\mathbb{F}_p$. Este polinômio tem q raízes distintas em $\mathbb{K}$, pois sua derivada, $qx^{q-1} - 1$, possui valor -1 em $\mathbb{F}_p$ e então não possui nenhuma raíz de multiplicidade maior que 1. Seja $S = \{a \in \mathbb{K} | a^q - a = 0\}$. Então, S é um subcorpo de $\mathbb{K}$, pois: (i) contém o 0 e o 1; (ii) $a, b \in S$ implica, pelo teorema 2.4.44, que $a - b$ pertence a S; (iii) $a, b \in S$ implica que $ab^{-1} \in S$. Portanto, como $\mathbb{K}$ é o menor corpo tal que o polinômio $x^q - x$ pode ser decomposto, e S é um subcorpo de $\mathbb{K}$ que permite essa decomposição, então temos que $\mathbb{K} = S$.

(Unicidade) Seja $\mathbb{K}$ um corpo finito com $q = p^m$ elementos. Então $\mathbb{K}$ tem característica p pelo teorema 2.5.10. Logo, pelo teorema 2.5.23, temos que $\mathbb{K}$ é um corpo de decomposição de $x^q - x$ sobre $\mathbb{F}_p$ e a unicidade segue pelo teorema 2.5.21.

2.5.4 Traço, Norma e Polinômio Característico

Seja $\mathbb{K}$ um corpo finito e $\mathbb{F}$ uma extensão de $\mathbb{K}$ finita de dimensão m. Então, dado um elemento $\alpha \in \mathbb{F}$, α pode ser representado como

$$\alpha = c_1\alpha_1 + ... + c_m\alpha_m,$$

onde $c_i \in \mathbb{K}$, para $1 \leq i \leq m$.

Definição 2.5.25. O *traço* de um elemento $\alpha \in \mathbb{F}$ é definido como sendo

$$\mathrm{Tr}_{\mathbb{F}/\mathbb{K}}(\alpha) = \alpha + \alpha^q + ... + \alpha^{q^{m-1}}.$$

Se $\mathbb{K}$ for um subcorpo primo de F, então $\mathrm{Tr}_{\mathbb{F}/\mathbb{K}}(\alpha)$ é denominado *traço absoluto* e é denotado apenas por $\mathrm{Tr}_{\mathbb{F}}(\alpha)$.

Em outras palavras, α é uma combinação linear de seus conjugados com relação a $\mathbb{K}$. Seja $f \in \mathbb{K}[x]$ um polinômio minimal tal que $f(\alpha) = 0$. Seja d o grau de f. Então d divide m e podemos definir o polinômio mônico g da seguinte forma:

$$g = f^{m/d}.$$

A função $g \in \mathbb{K}[x]$, definida desta forma, é denominada *polinômio característico* de α sobre $\mathbb{K}$.

Com isso, $g(x) = x^m + a_{m-1}x^{m-1} + ... + a_0$ e portanto é simples notar que

$$\mathrm{Tr}\,(\mathbb{F}/\mathbb{K}) = -a_{m-1}.$$

Definição 2.5.26. A *norma* de um elemento $\alpha \in \mathbb{F}$ sobre $\mathbb{K}$ é definida como sendo

$$\mathrm{Nor}_{(\mathbb{F}/\mathbb{K})}(\alpha) = \alpha.\alpha^q...\alpha^{q^{m-1}} = \alpha^{(q^m-1)/(q-1)}.$$

É simples ver que a norma de α também pode ser obtida diretamente do polinômio característico, pois $\mathrm{Nor}_{(\mathbb{F}/\mathbb{K})}(\alpha) = (-1)^m a_0$.

2.5.5 Raízes n-ésimas da Unidade

Dado um grupo $\mathbb{G}$ e $n \in \mathbb{Z}$, denominamos *raíz n-ésima da unidade* um elemento $z \in \mathbb{G}$, tal que $z^n = 1$, onde 1 representa o elemento neutro do grupo em questão.

Qualquer grupo possui pelo menos uma raíz n-ésima da unidade, já que o próprio elemento neutro satisfaz a condição acima. Em um grupo $\mathbb{G}$ qualquer, podemos afirmar que o fecho algébrico de $\mathbb{G}$ possui n raízes n-ésimas da unidade distintas. Se n divide a ordem de G, então todas as raízes n-ésimas da unidade estão contidas em G.

Para ilustrar, podemos utilizar o caso do grupo dos reais, com a operação de multiplicação usual, para verificar que a equação $z^n = 1$ possui as soluções -1 e 1 se n for par, ou apenas a solução 1 se n for ímpar. Mas no caso dos números complexos, se expressarmos z na forma polar, utilizando θ para representar o ângulo do segmento de reta que liga z a origem e λ para representar a distância de z a origem, temos que $z = \lambda(\cos\theta + i\sin\theta)$ e $z^n = \lambda^n(\cos(n\theta) + i\sin(n\theta))$. Portanto, $z^n = 1$ possui n soluções, dados por

$$z_k = \cos\tfrac{2k\pi}{n} + i\sin\tfrac{2k\pi}{n},\ k = 1, 2, \ldots, n;$$

como mostra a figura 2.5.5.

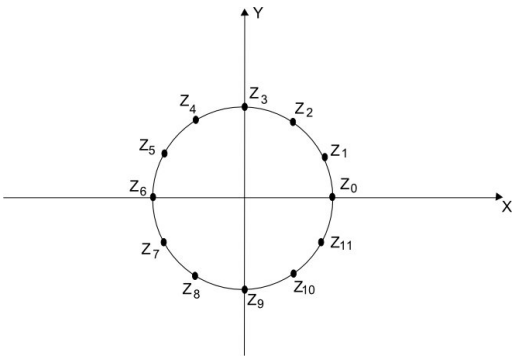


Figura 2.5.4: Raízes 12-ésimas da unidade no corpo dos números complexos

2.5.6 Grupo quociente e raízes n-ésimas da unidade

Seja G um grupo multiplicativo e $n \in \mathbb{N}$ tal que $|G| = hn$, para h inteiro positivo (h é denominado *índice* de G com relação a n). Então, G^n é um subgrupo de G, formado por elementos que são da forma a^n, para $a \in G$. Este subgrupo é composto por h elementos de ordem h, já que para qualquer $x \in G^n$, temos que $x^h = a^{hn} = 1$.

O grupo quociente G/G^n é composto por conjuntos laterais da forma $a.G^n$. Cada um desses subconjuntos possui apenas um elemento de ordem n. Uma forma de mapear elementos $a \in G$ para elementos de ordem n é calculando a^h. Com isso, temos que cada conjunto lateral de G/G^n é formado por elementos que quando elevados a h, resultam em uma mesma raíz n-ésima da unidade, de modo que G/G^n é isomorfo a μ_n, as raízes n-ésimas com relação a G.

Exemplo

Seja $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$, $n = 2$ e índice $h = 6/2 = 3$. Com isso, $(\mathbb{Z}_7^*)^2 = \{1^2, 2^2, 3^2, 4^2, 5^2, 6^2\} = \{1, 2, 4\}$. Deste modo, $\mathbb{Z}/(\mathbb{Z}_7^*)^2$ é composto pelos seguintes conjuntos:

$$(\mathbb{Z}_7^*)^2 = \{1, 2, 4\},$$

$$3(\mathbb{Z}_7^*)^2 = \{3, 5, 6\}.$$

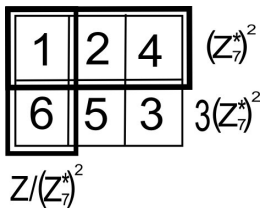


Figura 2.5.5: Grupos quocientes

Note que $3(\mathbb{Z}_7^*)^2 = 5(\mathbb{Z}_7^*)^2 = 6(\mathbb{Z}_7^*)^2$, ou seja, para gerar um subconjunto lateral distinto de $\{1, 2, 4\}$ basta multiplicá-lo por um elemento que não pertence a este próprio subconjunto. Além disso podemos ver que $\{1, 2, 4\}$ é composto de elementos de ordem 3. Cada um dos subconjuntos laterais possui apenas um elemento de ordem 2, sendo eles 1 e 6. Finalmente, se elevarmos ao cubo cada elemento de $(\mathbb{Z}_7^*)^2$, obtemos sempre o valor 1, enquanto que se fizermos o mesmo com cada elemento de $3(\mathbb{Z}_7^*)^2$, obtemos sempre o valor 6, de maneira que conseguimos mapear $\mathbb{Z}/(\mathbb{Z}_7^*)^2$ nas raízes 2-ésimas da unidade.

2.6 Curvas Elípticas

Nesta seção serão estudados os principais conceitos envolvidos em criptografia de curvas elípticas. Para isso, um grupo elíptico de pontos é definido, suas principais propriedades são discutidas e além disso, são definidas as condições que permitem a utilização de curvas elípticas para a construção de criptossistemas baseados em emparelhamentos bilineares.

2.6.1 Definições Preliminares

Definição 2.6.1. Uma curva elíptica $\mathbb{E}$, definida sobre um corpo $\mathbb{K}$, é um conjunto de pontos $P = (x, y)$, com $x, y \in \overline{\mathbb{K}}$, tais que $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ (forma padrão de Weierstrass), para $a_i \in \mathbb{K}$, além do *ponto no infinito*, denotado por ∞. Além disso, as derivadas parciais $2y + a_1x + a_3$ e $3x^2 + 2a_2x + a_4 + a_1y$ não assumem valor nulo simultaneamente, para um dado ponto $P = (x, y)$. Esta última condição, quando satisfeita, indica que a curva elíptica em questão é uma curva *não singular*.

Se definirmos $b_2 = a_1^2 + 4a_2$, $b_4 = a_1a_3 + 2a_4$, $b_6 = a_3^2 + 4a_6$ e $b_8 = a_1^2a_6 - a_1a_3a_4 + 4a_2a_6 + a_2a_3^2 - a_4^2$; então o mapa $y \to y - (a_1x + a_3)/2$ leva ao isomorfismo:

$$y^2 = x^3 + b_2x^2/4 + b_4x/2 + b_6/4.$$

Este polinômio tem apenas raízes simples se e somente se o seu *discriminante*, definido como sendo $\Delta = -b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6$, for não nulo. Uma curva E não singular, é uma curva elíptica se e somente se $\Delta \neq 0$.

Definição 2.6.2. Seja E uma curva elíptica sobre o corpo $\mathbb{K}$, o *invariante* j de E é definido como sendo $j(E) = (b_2^2 - 24b_4)^3/\Delta$.

Exemplo 2.6.3. Seja E uma curva elíptica sobre o corpo finito $\mathbb{F}_p$, com $p = 2003$ e cuja equação é dada por $y^2 + 2xy + 8y = x^3 + 5x^2 + 1136x + 531$. Para esta curva, temos que $b_2 = 24$, $b_4 = 285$, $b_6 = 185$, $\Delta = 1707$ e $j(E) = 171$.

A tabela 2.1 apresenta os valores de Δ, do invariante j e as respectivas equações de curva elíptica para valores da característica p do corpo finito subjacente.

Sejam $E/\mathbb{K}$ e $E'/\mathbb{K}$ duas curvas elípticas. Se E e E' são isomorfas sobre $\mathbb{K}$, então as curvas possuem o mesmo invariante j. Por outro lado, se $j(E) = j(E')$, então E e E' são isomorfos sobre $(\overline{\mathbb{K}})$, o fecho algébrico de $\mathbb{K}$.

Definição 2.6.4. Seja d um não resíduo quadrático em $\mathbb{F}_q$. A *curva entrelaçada* (twisted curve) de uma curva elíptica $E/\mathbb{F}_q : y^2 = x^3+ax+b$ é dada por $E^t/\mathbb{F}_q : y^2 = x^3+d^2ax+d^3b$. A curva $E/\mathbb{F}_{q^k}$ é isomorfa a $E^t/\mathbb{F}_{q^{k/2}}$.

Tabela 2.1: Equações de Weierstrass simplificadas

p	Equação	Δ	invariante j
$\neq 2, 3$	$y^2 = x^3 + a_4x + a_6$	$-16(4a_4^3 + 27a_6^2)$	$1728a_4^3/4\Delta$
3	$y^2 = x^3 + a_4x + a_6$	$-a_4^3$	0
3	$y^2 = x^3 + a_2x^2 + a_6$	$-a_2^3a_6$	$-a_2^3/a_6$
2	$y^2 + a_3y = x^3 + a_4x + a_6$	a_3^4	0
2	$y^2 + xy = x^3 + a_2x^2 + a_6$	a_6	$1/a_6$

Com isso, para $k = 2$, podemos realizar a aritmética em $E(\mathbb{F}_{q^2})$ de forma mais eficiente através de operações do grupo $E^t(\mathbb{F}_q)$.

Definindo-se uma operação de soma adequada, a curva elíptica forma um grupo aditivo abeliano, com o elemento neutro dado pelo ponto no infinito.

A figura 2.6.6 mostra a soma de dois pontos P_1 e P_2 em uma curva definida sobre os números reais. Conforme podemos notar pela figura, o resultado $-P_3$ da figura é obtido através do inverso do ponto P_3 encontrado pela reta que une P_1 e P_2

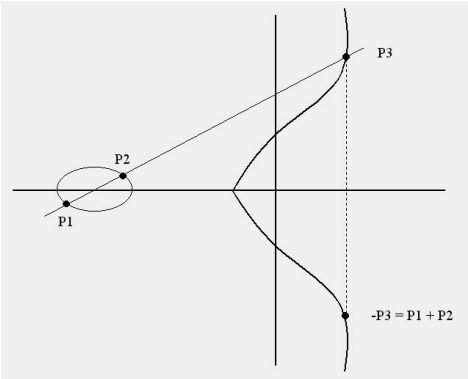


Figura 2.6.6: Soma de pontos da curva elíptica

É preciso notar que a curva elíptica no corpo dos números reais é uma função de 2 variáveis e portanto pode ser representada por uma superfície tridimensional, como na figura 2.6.8. O conjunto de pontos do grupo elíptico é dado pela intersecção desta superfície com o plano xy.

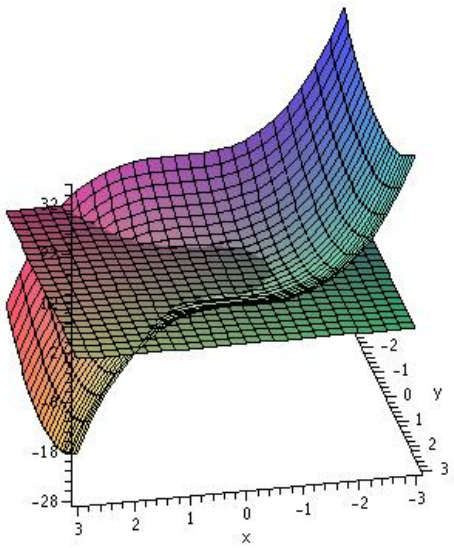


Figura 2.6.7: Curva elíptica no espaço tridimensional

Dados dois pontos distintos $P = (x_1, y_1)$ e $Q(x_2, y_2)$ de uma curva elíptica cujo corpo subjacente tenha característica diferente de 2, podemos calcular $P + Q = (x_3, y_3)$. A inclinação da reta que liga P a Q é

$$\lambda = \begin{cases} \dfrac{y_1 - y_2}{x_1 - x_2} & \text{se } P \neq Q, \\ \dfrac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} & \text{se } P = Q \end{cases}$$

Com isso, temos que

$$P + Q = (\lambda^2 + a_1\lambda - a_2 - x_1 - x_2, \lambda(x_1 - x_3) - y_1 - a_1x_3 - a_3),$$

Dado o inverso de um ponto P, o seu inverso $-P$, tal que $P = (-P) = \infty$, é dado por

$$-P = (x_1, y_1 - a_1x_1 - a_3).$$

Exemplo 2.6.5. Seja E a curva definida no exemplo 2.6.3, então os pontos $P = (1118, 269)$ e $Q = (892, 529)$ pertencem a curva e de acordo com as equações previamente estabelecidas, podemos calcular:

$$\begin{aligned} -P &= (1118, 1493), \\ P + Q &= (1681, 1706), \\ 2P &= (1465, 677) \end{aligned}$$

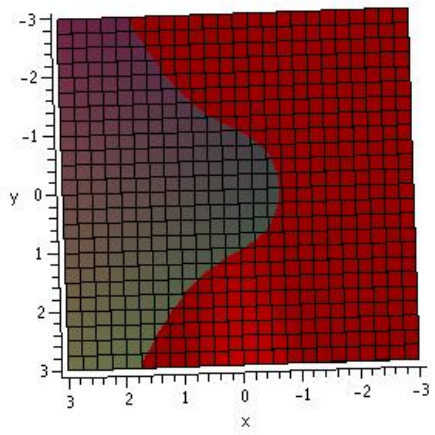


Figura 2.6.8: Intersecção da curva com o plano $z = 0$

Uma operação muito utilizada em criptografia de curvas elípticas é a multiplicação por escalar, em que um ponto P é somado com ele próprio k vezes, para $k \in \mathbb{Z}$. Um ponto de ordem n é um ponto tal que $nP = \infty$ e n é o menor inteiro positivo com esta propriedade.

Este sistema de coordenadas é denominado *sistemas de coordenadas afins*, denotado por $A^2_{\mathbb{K}} = (x, y) \in \mathbb{K} \times \mathbb{K}$, onde $\mathbb{K}$ é o corpo subjacente. O *sistema de coordenadas projetivas* é freqüentemente utilizado e será tratado na próxima seção.

2.6.2 Coordenadas projetivas

Em situações onde a inversão no corpo finito é muito mais cara que a multiplicação, é possível obter vantagem representando um ponto através de coordenadas projetivas. Neste modelo, um ponto elíptico padrão é representado por (X, Y, Z), com X, Y, Z pertencentes a um corpo finito $\mathbb{K}$ e pelo menos um dos valores diferente de zero. Este sistema de coordenadas é denotado por $P^2_{\mathbb{K}}$. Dois pontos (X_1, Y_1, Z_1) e (X_2, Y_2, Z_2) são ditos *equivalentes* se existir um elemento $\lambda \in \mathbb{K}$, diferente do elemento neutro aditivo, tal que $(X_1, Y_1, Z_1) = (\lambda X_2, \lambda Y_2, \lambda Z_2)$. Com isso, podemos definir uma classe de equivalência dada pela proporção entre X, Y e Z. Logo, para representar uma classe de equivalência, vai ser utilizada a notação $(X : Y : Z)$

Um polinômio $F(X, Y, Z)$ é *homogêneo* de grau n, se a soma dos graus de X, Y e Z de cada termo deste polinômio é n. Por exemplo, $F(X, Y, Z) = X^2YZ^3 + 5X^3YZ -$

$7XY^2Z^2$ é um polinômio homogêneo de grau 5. Em um polinômio homogêneo de grau n, $F(\lambda X, \lambda Y, \lambda Z) = \lambda^n F(X, Y, Z)$.

Em coordenadas projetivas, a equação da curva elíptica dada pela forma padrão de Weierstrass precisa ser generalizada de forma que se um ponto $P = (X : Y : Z)$ é solução, então qualquer ponto equivalente a P também deve ser solução da curva. Para isso, inserimos potências da coordenada Z para definir uma curva elíptica homogênea de grau 3, dada por $E(X, Y, Z) = Y^2Z + a_1XYZ + a_3YZ^2 + X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$.

Se $Z \neq 0$, então $(X : Y : Z) \equiv (X/Z : Y/Z : 1)$. Pontos desta forma são pontos comuns da curva elíptica, correspondendo a um ponto elíptico afim dado por $(X/Z, Y/Z)$. Se $Z = 0$ e pelo menos $X \neq 0$ ou $Y \neq 0$, então $(X : Y : 0)$ representa o ponto do infinito. Desta maneira, o ponto do infinito surge em coordenadas projetivas como um outro ponto qualquer, não sendo necessário acrescentar um ponto artificial ao sistema para ter um elemento neutro no grupo elíptico.

Em corpos binários a equação da curva elíptica projetiva é um caso específico da forma padrão de Weierstrass, dada por $Y^2Z + XYZ = X^3 + aX^2Z + bZ^3$. Pode-se ainda utilizar coordenadas projetivas Jacobianas, onde um ponto projetivo $(X : Y : Z)$ corresponde a um ponto em coordenadas afins dado por $(X/Z^2, Y/Z^3)$, onde a curva elíptica projetiva é dada pela equação $Y^2 + XYZ = X^3 + aX^2Z^2 + bZ^6$. Em [LD99], é proposto um novo sistema de coordenadas projetivas, onde um ponto projetivo $(X : Y : Z)$ corresponde a um ponto em coordenadas afins dado por $(X/Z, Y/Z^2)$, cuja curva elíptica projetiva equivalente é a equação

$$Y^2 + XYZ = X^3Z + aX^2Z^2 + bZ^4.$$

Fórmulas para somar e duplicar pontos podem ser derivadas de um sistema para o outro, sendo que em coordenadas projetivas não é preciso calcular o inverso de um elemento do corpo subjacente. Neste trabalho será estudada a utilização apropriada de coordenadas projetivas, tendo em vista a maior eficiência nos cálculos.

Até agora foi descrita a forma algébrica de lidar com coordenadas projetivas. Para adquirir intuição e familiaridade com este tipo de geometria, é importante entender como é possível transformar um sistema geométrico de coordenadas afins do conjunto dos números reais em duas dimensões para um sistema geométrico de coordenadas projetivas em três dimensões. A figura 2.6.9 representa a transformação de pontos em coordenadas afins para projetivas, como em coordenadas projetivas utilizamos polinômios homogênios, a esfera de raio 1 e centrada na origem representa cada ponto do sistema projetivo. De fato, apenas meia esfera é suficiente para a representação. A transformação é realizada através da projeção pelo pólo $P = (0, 0, 0)$, o centro da esfera. O ponto $(0 : 1 : 0)$ é o ponto que representa o ponto no infinito, mas qualquer ponto do eixo Y pode ser considerado como equivalente ao ponto do infinito. A *linha do equador* na figura representa a *linha do*

infinito e os pontos nesta linha também são equivalentes ao ponto do infinito. O ponto $(0:0:1)$ é projetado nele próprio. Se for considerado um ponto cada vez mais próximo de P através da esfera, a sua projeção no plano é cada vez mais próxima do "infinito" deste plano.

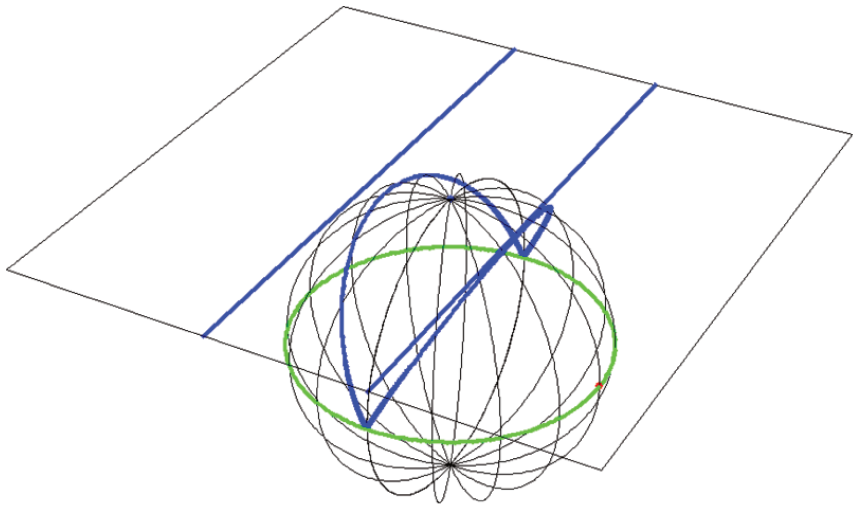


Figura 2.6.9: Coordenadas Projetivas

2.6.3 Multiplicação escalar

Seja $P \in E/\mathbb{K}$ um ponto arbitrário de uma curva E definida sobre um corpo $\mathbb{K}$. Dado $n \in \mathbb{N}$, definimos *multiplicação escalar*, denotada por $[n]P$, correspondendo a soma de P com ele próprio n vezes. Ou seja,

$$[n]P = \underbrace{P + ... + P}_{\text{n vezes}}.$$

2.6.4 Grupo de n-Torção

Definição 2.6.6. O grupo de n-torção $E[n]$ é composto por pontos $P \in \mathbb{E}(\overline{\mathbb{F}}_{p^m})$ tais que $nP = \infty$, ou seja, são os pontos de ordem que divide n. O grau de mergulho de $E[n]$ é o menor inteiro k tal que $E[n] \subseteq \mathbb{E}(\mathbb{F}_{p^{km}})$.

Teorema 2.6.7. Sejam n e p inteiros positivos primos entre si, isto é, $p \nmid n$, então $E[n] \sim \mathbb{Z}_n \oplus \mathbb{Z}_n$.

Exemplo 2.6.8. Por exemplo, seja a curva elíptica E sobre o corpo finito $\mathbb{F}_7$ definida pela equação

$$y^2 = x^3 + 2.$$

O grupo elíptico é formado pelos pontos $\{(0,3),\ (0,4),\ (3,1),\ (3,6),\ (5,1),\ (5,6),\ (6,1),\ (6,6),\ \infty\}$. Portanto, $E(\mathbb{F}_7)$ possui 9 elementos. Se escolhermos $n = 3$, temos que $E(\mathbb{F}_7)[3] \sim \mathbb{Z}_3 \oplus \mathbb{Z}_3$, pois 3 e 7 são primos entre si. De fato, $E(\mathbb{F}_7)[3]$ é composto por todos os elementos de E.

Podemos observar a estrutura de $E(\mathbb{F}_7)[3]$ na figura 2.6.10. Todo elemento, exceto o ponto do infinito, é gerador de um subgrupo de ordem 3, composto pelo ponto do infinito e outros dois elementos cuja coordenada x possui valor comum. Assim, o subconjunto $\{(0,3),(0,4),\infty\}$ forma um subgrupo onde $(0,3)$ e $(0,4)$ são geradores. Ou seja, $(0,3) + (0,3) = (0,4)$, $(0,4) + (0,4) = (0,3)$ e $(0,3) + (0,4) = \infty$.

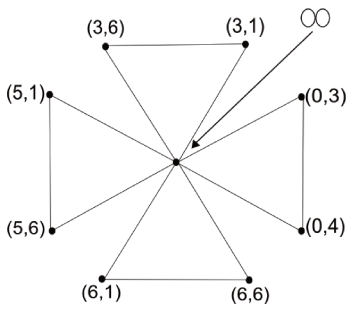


Figura 2.6.10: Estrutura dos grupos de 3-torção para $E(\mathbb{F}_7)$.

Teorema 2.6.9. *(Teorema de Hasse.)* Seja E uma curva eliptíca definida sobre um corpo finito $\mathbb{F}_q$. Então

$$|E(\mathbb{F}_q)| = q + 1 - t \text{ e } |t| \leq 2\sqrt{q}.$$

O inteiro t é chamado *traço do endomorfismo de Frobenius*. Para cada valor de t no intervalo $[-2\sqrt{p}, 2\sqrt{p}]$, para p primo, existe uma curva elíptica E definida sobre $\mathbb{F}_p$ cuja cardinalidade é $p + 1 - t$.

Teorema 2.6.10. Seja $q = p^d$, então existe uma curva elíptica E definida sobre $\mathbb{F}_q$, tal que $|E(\mathbb{F}_q)| = q + 1 - t$, se e somente se

(i) $t \not\equiv 0 \pmod{p}$ e $t^2 \leq 4q$.

(ii) d é ímpar e uma das situações seguintes é válida:

(a) $t = 0$.

(b) $p = 2$ e $t^2 = 2q$.

(c) $p = 3$ e $t^2 = 3q$.

(iii) d é par e uma das situações seguintes é válida:

(a) $t^2 = 4q$.

(b) $p \not\equiv 1 \pmod 3$ e $t^2 = q$.

(c) $p \not\equiv 1 \pmod 4$ e $t = 0$.

Definição 2.6.11. Uma curva elíptica $E/\mathbb{F}_q$, onde $q = p^m$, é denominada *supersingular* se a característica p do corpo finito subjacente for tal que $p|t$, onde t é o traço de Frobenius. Caso contrário, a curva é denominada *não supersingular*. Em outras palavras, a curva E é supersingular se e somente se $t \equiv 0 \pmod p$.

Teorema 2.6.12. Seja p a característica do corpo finito sobre o qual a curva elíptica E é definida. Se $E[p^r] = \{P_\infty\}$ para r um inteiro positivo qualquer, então a curva E é *supersingular*. Caso contrário, a curva é *ordinária*.

Um corolário importante deste teorema é que curvas cupersingulares sobre corpos binários não possuem pontos de ordem 2, ou seja, não há pontos tais que $P = -P$.

2.6.5 Endomorfismos

Dado $n \in \mathbb{Z}$, então a multiplicação de cada ponto de uma curva elíptica E por n é um endomorfismo, denotado por $[n]$. O conjunto de todos os endomorfismos de E é denotado por $\mathrm{End}\,(E)$ forma um grupo com relação a composição usual de mapas.

Definição 2.6.13. Se $\mathrm{End}\,(E)$ for estritamente maior que $\mathbb{Z}$ então dizemos que E possui *multiplicação complexa*.

Uma curva elíptica E definida sobre um corpo finito $\mathbb{F}_q$ sempre possui multiplicação complexa. De fato, o *endomorfismo de Frobenius*, definido como sendo $\phi_q(P) = (x^q, y^q)$, para $P = (x, y)$. É fácil ver que $\phi_q(P)$ é um ponto da curva E. Além disso, não existe valor de $n \in \mathbb{Z}$, tal que ϕ_q seja isomorfo a $[n]$.

Se aplicarmos o endomorfismo de Frobenius ϕ_q em elementos de $\mathbb{F}_q$, então temos que $\phi_q(x) = x$, para $x \in \mathbb{F}_q$. Por isso, se $q = p^m$, para $m > 1$, então ϕ_p é um automorfismo de $E(\mathbb{F}_{p^m})$ que fixa elementos de $E(\mathbb{F}_p)$.

O endomorfimo de Frobenius ϕ_q está associado a um polinômio característico

$$\mathcal{X}_E(T) = T^2 - tT + q,$$

tal que ϕ_q é raíz e portanto temos que

$$\phi_q^2(P) + [-t]\phi_q(P) + [q](P) = \infty.$$

A cardinalidade da curva elíptica $E(\mathbb{F}_q)$ é

$$|E(\mathbb{F}_q)| = \mathcal{X}_E(1).$$

Sejam τ e $\overline{\tau}$ as raízes do polinômio característico $\mathcal{X}_E$. Então, podemos calcular a ordem de uma curva elíptica definida sobre uma extensão finita de $\mathbb{F}_q$. Mais especificamente, temos que

$$|E(\mathbb{F}_{q^k})| = q^k + 1 - \tau^k - \overline{\tau}^k.$$

Com alguns cálculos é possível observar que o traço da curva sobre a extensão do corpo subjacente é $t_k = \tau^k + \overline{\tau}^k$ e t_k é uma sequência tal que $t_0 = 2$, $t_1 = t$ e $t_{k+1} = t.t_k - q.t_{k-1}$, para $k \geq 1$.

2.6.6 Divisores

Para efeito de emparelhamentos, divisores são somas formais finitas de símbolos $[P]$ associados a pontos P da curva, definida sobre o fecho algébrico do corpo subjacente. Um divisor D é dado por

$$D = \sum_{P \in \mathbb{E}(\overline{\mathbb{F}})} a_P[P],$$

onde $a_P \in \mathbb{Z}$. Define-se a operação de soma de dois divisores através da soma dos coeficientes de cada ponto, ou seja, dados $D_1 = \sum a_P[P]$ e $D_2 = \sum b_P[P]$, tem-se que

$$D_1 + D_2 = \sum c_P[P], c_P = a_P + b_P.$$

Desta forma, o conjunto dos divisores de uma curva elíptica forma um grupo aditivo, $Div(E)$. Podemos definir a seguir o grau de um divisor, $\delta(D)$, e a soma de um divisor, $\sigma(D)$, dados por

$$\delta(D) = \sum_{P \in \mathbb{E}(\overline{\mathbb{F}})} a_P$$

e

$$\sigma(D) = \sum_{P \in \mathbb{E}(\overline{\mathbb{F}})} a_P P.$$

Note que $\delta(D)$ resulta um número enquanto $\sigma(D)$ resulta um ponto da curva.

O *suporte* de um divisor é definido como sendo os pontos P cujo coeficiente a_P é diferente de zero.

Um subgrupo importante do grupo de divisores de uma curva é aquele formado pelos divisores de grau zero, denotado por $Div^0(E)$.

Uma função racional f em uma curva elíptica é aquela que pode ser expressa como a divisão de polinômios de duas variáveis, definida em pelo menos um ponto da curva, com coeficientes no corpo sobre o qual a curva é definida, módulo a equação da curva. Com isso,

$$f(x, y) = \frac{p(x, y)}{q(x, y)},$$

onde p e q são polinômios de duas variáveis definidos sobre o mesmo corpo finito de E. É preciso ter em mente que $f(x, y)$ representa uma superfície no espaço tridimensional quando estamos no corpo dos números reais, mas como estamos interessados apenas em aplicações desta função para pontos da curva elíptica, onde estamos limitados a pontos discretos do plano xy, a função f pode ser encarada como se fosse um conjunto de pontos de uma linha no plano xy.

Para uma dada função racional f, é possível transformá-la de tal maneira que não exista um ponto (x, y) de E onde $f(x, y) = 0/0$. Se $p(x, y) = 0$, dizemos que o ponto (x, y) é um *zero* de f. Se $q(x, y) = 0$, dizemos que o ponto (x, y) é um *pólo* de f, neste caso $f(x, y) = \infty$.

Seja P um ponto da curva, é possível mostrar que existe uma função u_P, denominada uniformizador em P, tal que $u_P(P) = 0$ e para qualquer função racional f podemos escrever

$$f(x, y) = u_P^r g(x, y).$$

Em outras palavras, existe uma função que permite inferir a ordem de um ponto P para uma dada função racional. O mesmo conceito de uniformizador pode ser utilizado em funções polinomiais de uma variável, permitindo entender mais facilmente o que está acontecendo. Em polinômios de uma variável, um ponto P é representado apenas uma coordenada x_P, de modo que $u_P = (x - x_P)$ é um uniformizador válido. Se x_P é raíz de um polinômio $p(x)$, então podemos escrever $p(x) = (x - x_P)^r g(x)$, onde r é a multiplicadade de x_P e $g(x_P) \neq 0$. Por exemplo, se temos $x_P = 1$ e $p(x) = x^3 - x^2 - x + 1$, então é fácil ver que $p(x) = (x - 1)^2(x + 1)$, de forma que $x_P = 1$ é raíz de $p(x)$ com multiplicidade 2 e

$u_P = (x-1)$ é o uniformizador em questão. É possível mostrar que para funções racionais sobre uma dada curva elíptica também é possível encontrar tais uniformizadores.

Assim como em polinômios de uma variável, é possível determinar facilmente uniformizadores de funções racionais sobre curvas elípticas, sendo necessário apenas distinguir entre dois casos:

- se o ponto $P = (x_P, y_P)$ possui coordenada $y_P \neq 0$, então o uniformizador é a linha perpendicular $u_P = x - x_P$.
- Caso contrário, se $y_P = 0$, então $u_P = y$.

O divisor de uma função racional f é definido como sendo a soma formal dos zeros e pólos da função na curva, com suas respectivas multiplicidades, e sinal negativo no caso dos pólos. Divisores de funções racionais são denominados divisores principais. É possível demonstrar que um divisor D é principal se e somente se $\delta(D) = 0$ e $\sigma(D) = \infty$. Divisores de grau zero podem ser mapeados para pontos da curva através do mapa σ.

Exemplos

Existem três problemas interessantes de serem compreendidos em divisores de curvas elípticas. Primeiramente, dado um ponto P, calcular a ordem de P. Em segundo lugar, dada uma função racional f, definida em pelo menos um ponto da curva, calcular o divisor associado a esta função. Em terceiro lugar, temos o problema inverso, ou seja, dado um divisor principal D, calcular a função racional correspondente a D. Nesta seção vamos dar exemplos de como resolver estes três problemas.

Cálculo da ordem de um ponto

Seja a curva $y^2 = x^3 + 72$ e o ponto $P = (-2, 8)$. A linha $x + 2 = 0$ passa por P, então podemos considerar um uniformizador $u_P(x, y) = x + 2$. A função $f(x, y) = x + y - 6$ tem valor 0 em P. Com alguns cálculos é possível reescrever a função f de modo a obter uma expressão da forma $u_P^r g(x, y)$. Neste caso, é possíver reescever $f(x, y)$ da seguinte forma:

$$f(x, y) = (x + 2)\left(1 + \frac{(x + 2)^2 - 6(x + 2) + 12}{y + 8}\right).$$

Como a função entre parênteses é diferente de 0 e é finita (não possui denominador 0), então podemos considerar

$$g(x, y) = \left(1 + \frac{(x + 2)^2 - 6(x + 2) + 12}{y + 8}\right).$$

Logo, a ordem de f em P é 1.

De maneira semelhante é possível mostrar que se considerarmos uma reta tangente em P, $t(x, y) = 3/4(x + 2) - y + 8$, a ordem desta reta é 2.

De modo geral, uma reta secante ao ponto P possui ordem 1, enquanto que retas tangentes ao ponto P, possuem ordem 2 se $3P \neq \infty$, ou ordem 3 se $3P = \infty$.

Não é tão simples mostrar que o uniformizador no ponto do infinito é $u_\infty = x/y$.

Cálculo do divisor correspodente a uma função racional

Para calcular o divisor correspondente a uma dada função racional é preciso calcular a ordem de cada ponto da curva em relação a esta função. Mas se um ponto da curva não é zero ou pólo da função racional, então isto significa que este ponto possui ordem 0. Com isso, o cálculo do divisor correspondente é realizado levando em consideração apenas pontos da curva tais que a função racional assume valor 0 ou ∞. Ou seja,

$$\mathrm{div}(f) = \sum_{P \in E} \mathrm{ord}_P(f)[P].$$

Por exemplo, seja E a curva dada pela equação $y^2 = x^3 + 2$, sobre o corpo finito $\mathbb{F}_7$. Esta curva possui apenas elementos de ordem 3, isto é, pontos P tais que $3P = \infty$. Desta forma, como acabamos de ver, uma reta r_P tangente ano ponto P possui ordem 3 neste ponto. Com isso,

$$\mathrm{div}(r_P) = 3[P] - 3[\infty].$$

Se $P = (0, 3)$, então a reta $y - 3$ é tangente a P, de forma que $\mathrm{div}(y - 3) = 3[(0, 3)] - 3[\infty]$.

Se quisermos calcular a função racional cujo divisor é $D = 3[P] - 3[Q]$, para P e Q pontos de ordem 3 da curva E, então $D = (3[P] - 3[\infty]) - (3[Q] - 3[\infty])$. Com isso,

$$D = \mathrm{div}(r_P) - \mathrm{div}(r_Q).$$

Ou seja, a função $f(x, y)$ que corresponde ao divisor $3[(3, 6)] - 3[(6, 1)]$ é dada pela divisão da reta tangente em $(3, 6)$ pela reta tangente em $(6, 1)$, isto é,

$$f(x, y) = \frac{4x - y + 1}{5x - y - 1}.$$

Estas relações também podem ser utilizadas para o cálculo da função racional correspondente a um divisor principal, como veremos na próxima seção.

Cálculo da função racional correspondente a um divisor principal

Seja $ax + by + c$ a reta que passa pelos pontos P_1, P_2 e P_3 da curva E. Então, $\text{div}(ax + by + c) = [P_1] + [P_2] + [P_3] - 3[\infty]$.

A linha que passa pelos pontos P_3 e $-P_3$ é dada pela equação $x - x_3$, cujo divisor é $\text{div}(x - x_3) = [P_3] + [-P_3] - 2[\infty]$.

Logo, como $P_1 + P_2 = -P_3$, então

$$[P_1] + [P_2] = [P_1 + P_2] + [\infty] + \text{div}\left(\frac{ax + by + x}{x - x_3}\right). \tag{2.1}$$

Isto significa que existe uma forma simples de relacionar divisores equivalentes. Além disso, dado um divisor $D = \sum_{P \in \mathbb{E}} a_P[P]$, é possível reduzir este divisor de forma a obter uma expressão da forma:

$$D = \text{div}(f(x, y)).$$

A função $f(x, y)$ é obtida através de equações de retas como as mostradas na equação 2.1.

2.7 Emparelhamentos Bilineares

Emparelhamentos bilineares são mapas sobrejetivos, levando pares de elementos de um grupo aditivo G_1, cujo elemento neutro é ∞, em elementos de um grupo multiplicativo cíclico G_2, cujo elemento neutro é 1, com as seguintes propriedades:

1. **Bilinearidade.** Para quaisquer elementos S, S_1, S_2, T, T_1, T_2 de G_1, temos que

$$<S_1 + S_2, T> = <S_1, T> . <S_2, T>,$$

$$<S, T_1 + T_2> = <S, T_1> . <S, T_2>,$$

2. **Não degeneração.** Se $<S, T> = 1$, para todo elemento $T \in G_1$, então $S = \infty$. Se $<S, T> = 1$, para todo elemento $S \in G_1$, então $T = \infty$.

3. **Computabilidade.** Para quaisquer $S, T \in G_1$, $< S, T >$ pode ser calculado em tempo polinomial por uma máquina de Turing.

2.7.1 Emparelhamento de Tate

Seja uma curva elíptica E, definida sobre o corpo finito $\mathbb{F}_{q^k}$. Seja n um inteiro tal que n divida a ordem do corpo finito, ou seja, $n|q^k - 1$. Se $E[n] \subset E(\mathbb{F}_{q^k})$, isto é, n^2 divide a ordem da curva, então podemos definir o emparelhamento de Tate da seguinte forma:

$$<,>_n: E(\mathbb{F}_{q^k})[n] \times E(\mathbb{F}_{q^k})/nE(\mathbb{F}_{q^k}) \rightarrow \mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^n.$$

O emparelhamento de Tate $<S,T>_n$ pode ser calculado através de uma função racional f_S, cujo divisor correspondente é nD_S, onde D_S é um divisor de grau zero, equivalente a S e com suporte distinto de T. Mais especificamente,

$$<S,T>_n = f_S(D_T), \tag{2.2}$$

onde $D_T = \sum_i a_i[T_i]$.

Para calcular $f_S(D_T)$, podemos utilizar a seguinte relação:

$$f_S(D_T) = \prod_i f_S(T_i)^{a_i}.$$

2.7.2 Emparelhamento de Weil

Seja E uma curva elíptica definida sobre o corpo $\mathbb{F}_{q^k}$ e $E[n]$ o subgrupo de pontos de n-torção, tais que $E[n] \subset E(\mathbb{F}_{q^k})$. Seja $S, T \in E[n]$, então sejam D_S e D_T divisores de grau zero tais que $\sigma(D_S) = S$ e $\sigma(D_T) = T$. Sejam as funções f_S e f_T tais que $\text{div}(f_S) = nD_S$ e $\text{div}(f_T) = nD_T$.

O emparelhamento de Weil é definido como sendo o mapa $e_n : E[n] \times E[n] \rightarrow \mu_n$, dado por

$$e_n(S,T) = \frac{f_T(D_S)}{f_S(D_T)}.$$

Assim, podemos ver que o emparelhamento de Weil pode ser calculado usando duas vezes o emparelhamento de Tate, descrito na seção anterior, ou seja,

$$e_n(S,T) = \frac{<T,S>_n}{<S,T>_n}. \tag{2.3}$$

2.7.3 Emparelhamento de Tate-Lichtenbaum

Nesta seção será apresentada a descrição do processo que permite calcular o emparelhamento de Tate-Lichtenbaum, objetivo principal deste trabalho.

Seja k um inteiro tal que $\mathbb{F}_{q^k}$ contém as raízes n-ésimas da unidade $\mu_n \in \mathbb{F}_{q^k}$, isto é, $n|q^k - 1$. O emparelhamento de Tate é definido através do seguinte mapeamento:

$$e : E(\mathbb{F}_{q^k})[n] \times E(\mathbb{F}_{q^k})/nE(\mathbb{F}_{q^k}) \rightarrow \mu_n,$$

onde $E(\mathbb{F}_{q^k})[n]$ são os pontos P da curva tais que $nP = \infty$, de modo que esta curva precisa ser tal que n^2 divide a ordem de $E(\mathbb{F}_{q^k})$, isto é $|E(\mathbb{F}_{q^k})| = hn^2$, onde h é denominado *cofator* da curva com relação a n^2. Desta forma, $E(\mathbb{F}_{q^k})/nE(\mathbb{F}_{q^k})$ é composto por classes de equivalênvia análogas ao grupo quociente mostrado na seção 2.2.19, ou seja, cada classe possui exatamente um elemento de ordem n. Se temos n^2 elementos de ordem n, então temos também n^2 classes de equivalência em $E(\mathbb{F}_{q^k})/nE(\mathbb{F}_{q^k})$, cada uma com h elementos.

O emparelhamento de Tate-Lichtenbaum pode ser calculado como sendo $e(P, Q) = g(D)^{\frac{q^k-1}{n}}$, onde D é um divisor associado ao ponto Q e g é uma função racional cujo divisor é $n[P] - n[\infty]$. O algoritmo de Miller [Mil04a] pode ser usado para calcular a função g.

Este mapeamento possui algumas propriedades importantes, como a bilinearidade ($e(aQ, bP) = e(Q, P)^{ab}$), a não degeneração, que garante que nem todos os pares de pontos serão levados ao elemento neutro do grupo gerado pelo emparelhamento; e a computabilidade, que garante que o emparelhamento pode ser calculado eficientemente.

Como podemos notar, o emparelhamento de Tate-Lichtenbaum também pode ser expresso em função do emparelhamento de Tate, descrito na seção 2.7.1, através da equação

$$e(S, T) = <T, S>_n^{\frac{q^k-1}{n}} . \tag{2.4}$$

2.7.4 Exemplo

Como vimos na seção 2.6.8, a curva $y^2 = x^3 + 2$ possui 9 soluções no corpo $\mathbb{F}_7$ e todas estas soluções correspondem a elementos de ordem 3, ou seja, neste caso o cofator h vale 1 e $E(\mathbb{F}_7) \sim \mathbb{Z}_3 \oplus \mathbb{Z}_3$. Com isso, esta curva não precisa de uma extensão de $\mathbb{F}_7$ para o cálculo do emparelhamento de Tate, isto é, $k = 1$.

Para calcular $e_3((0, 3), (5, 1))$, é preciso encontrar funções racionais $f_{(0,3)}$ e $f_{(5,1)}$, tais que $\text{div}(f_{(0,3)}) = 3[(0, 3)] - 3[(\infty)]$ e $\text{div}(f_{(5,1)}) = 3[(3, 6)] - 3[(6, 1)]$. Em particular, sabemos que $(5, 1) = (3, 6) - (6, 1)$ e como precisamos usar divisores com suportes distintos, podemos utilizar esta relação para o cálculo do emparelhamento de Tate.

Podemos utilizar as relações descritas na seção 2.6.6 para calcular $f_{(0,3)} = y - 3$ e $f_{(5,1)} = \frac{4x-y+1}{5x-y-1}$.

Com isso, temos que

$$f_{(0,3)}(D_{(5,1)}) = \frac{f_{(0,3)}(3,6)}{f_{(0,3)}(6,1)} = 2. \tag{2.5}$$

Analogamente, temos que

$$f_{(5,1)}(D_{(0,3)}) = 4. \tag{2.6}$$

Desta forma, podemos ver que o emparelhamento de Tate pode ser calculado através da equação 2.5 ou da equação 2.6, o que nos permite ver que o emparelhamento de Tate não é simétrico com relação aos seus argumentos. Além disso, podemos ver que $<(0,3),(5,1)>_3 = 2$ e $<(5,1),(0,3)>_3 = 4$, portanto são raízes 3-ésimas da unidade, já que $2^3 \equiv 1 \pmod 7$ e $4^3 \equiv 1 \pmod 7$. O emparelhamento de Tate não necessariamente resulta em uma raíz 3-ésima da unidade. Por exemplo, $<(5,1),(6,6)>_3 = f_{(5,1)}(D_{(6,6)}) = 6$ e 6 não é raíz 3-ésima da unidade.

Já o cálculo do emparelhamento de Weil pode ser realizado através da equação 2.3, de modo que $e_n((0,3),(5,1)) = 4/2 = 2$, isto é, um raíz 3-ésima da unidade. O emparelhamento de Tate-Lichtenbaum por sua vez, pode ser calculado utilizando a equação 2.4, obtendo $e((0,3),(5,1)) = 4^2 = 2$. Com podemos notar, tanto o emparelhamento de Weil como o de Tate-Lichtenbaum resultam em uma raíz 3-ésima da unidade. No caso do emparelhamento de Weil, é pelo fato de termos duas aplicações do emparelhamento de Tate que temos a garantia de obter uma raíz 3-ésima da unidade, enquanto que para o emparelhamento de Tate-Lichtenbaum é a exponenciação final que nos dá essa garantia.

2.7.5 Algoritmo de Miller

Como vimos na seção anterior, o emparelhamento de Weil e o de Tate-Lichtenbaum utilizam como princípio o cálculo de uma função racional f tal que $\mathrm{div}(f) = n[P+R] - n[R]$ e o cálculo do resultado desta função em um divisor da forma $D_T = [T_1] - [T_2]$. Ou seja, é preciso obter $f(D_T) = f(T_1)/f(T_2)$. A idéia do algoritmo de Miller é poder encontrar f_{j+k} através de f_j e f_k, onde f_i representa a função racional cujo divisor correspondente é

$$D_i = i[P+R] - i[R] - [iR] + [\infty]. \tag{2.7}$$

Este divisor foi criado porque $D_n = n[P+R] - n[R] - [nP] + [\infty]$, e como $P \in E[n]$, $nP = \infty$ e portanto $D_n = n[P+R] - n[R]$, como desejamos. Além disso, cada D_i é um divisor principal, de maneira que existe uma função f_i correspondente. Agora podemos ver como é possível construir essas funções de modo a obter $f_n(T_1)/f_n(T_2)$.

Supondo que já calculamos $f_j(T_1)/f_j(T_2)$ e $f_k(T_1)/f_k(T_2)$. Seja $ax + by + c = 0$ a equação da reta que passa por jP e kP e seja $x + d = 0$ a reta vertical que passa por $(j+k)P$, então

$$\text{div}\left(\frac{ax+by+c}{x+d}\right) = [jP] + [kP] - [(j+k)P] - [\infty].$$

Logo,

$$D_{j+k} = D_j + D_k + div\left(\frac{ax+by+c}{x+d}\right).$$

Portanto,

$$f_{j+k} = \gamma f_j f_k \frac{ax+by+c}{x+d}.$$

Assim,

$$\frac{f_{j+k}(T_1)}{f_{j+k}(T_2)} = \frac{f_j(T_1)}{f_j(T_2)}.\frac{f_k(T_1)}{f_k(T_2)}.\frac{\frac{ax+by+c}{x+d}|_{(x,y)} = T_1}{\frac{ax+by+c}{x+d}|_{(x,y)} = T_2}. \tag{2.8}$$

Capítulo 3

Algoritmos

> Nós podemos agora construir uma máquina para realizar o trabalho deste computador.
>
> Alan Turing

3.1 Introdução

Neste capítulo serão descritos os principais algoritmos de aritmética em $\mathbb{Z}$. Com esses algoritmos é possível construir estruturas algébricas como os números racionais, complexos, e p-ádicos, assim como polinômios com coeficientes nestes conjuntos. Justamente por ser utilizada como base de outras estruturas, a aritmética de números inteiros precisa ser o mais eficiente possível. Os algoritmos que serão apresentados nesta seção foram diretamente tirados de [CF06].

Algoritmos para aritmética em corpos binários serão descritos a seguir, permitindo analisar o uso de bases polinomiais contra o uso de bases normais. Estes algoritmos são a bases para a implementação de deste projeto, pois permitem o cálculo eficiente de duplicações em certas classes de curvas elípticas supersingulares.

A aritmética de curvas elípticas é estudada a seguir, com objetivo principal de obter um algoritmo eficiente para cálculo da multiplicação escalar.

Serão apresentados ataques ao problema do logaritmo discreto que restringirão a escolha de parâmetros para criptossistemas baseados em curvas elípticas. Algoritmos para ataque de índices serão descritos a seguir, juntamente com as razões pelas quais esses ataques não funcionam em curvas elípticas. Para finalizar, o conceito de transferência de logaritmo discreto é brevemente apresentado.

3.2 Aritmética nos Números Inteiros

Seja $b \geq 2$ um inteiro denominado *base*. Todo inteiro $u > 0$, pode ser escrito como a soma

$$u = u_{n-1}b^{n-1} + ... + u_1 b + u_0,$$

onde $0 \leq u_i < b$ e $u_{n-1} \neq 0$.

Esta soma é denominada *representação de u na base b* e é denotada por $(u_{n-1}...u_0)_b$. Cada u_i denominado *dígito* de u na base b.

Normalmente, computadores podem efetuar cálculos utilizando uma quantidade de memória previamente fixada, ou seja, os números envolvidos na operação precisam ser representados utilizando uma quantidade fixa de bytes. Esta unidade lógica de processamento é denominada *palavra*. Atualmente as palavras são compostas de 4 bytes, isto é, 32 bits. Mas é comum que certas arquiteturas utilizem palavras de 8 bits, 16 bits e 64 bits.

Assim, para representar um número inteiro de tamanho arbitrário, é preciso utilizar uma sequência de palavras, cada uma contendo um dígito do número. Portanto, em uma arquitetura de palavra de 32 bits, cada dígito pode assumir valor entre zero e 2^{32}. Com isso, o número pode ser expresso na base $b = 2^{32}$ como um vetor de palavras. A figura a seguir mostra um exemplo desta representação.

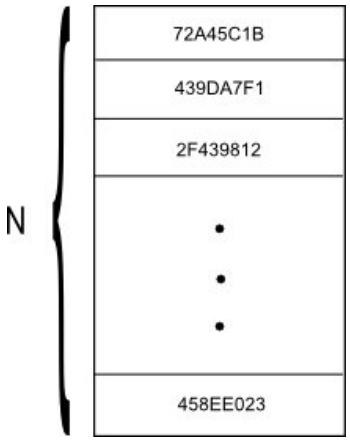


Figura 3.2.1: Representação de número de precisão arbitrária na base $b = 2^{32}$.

Assim, o número representado na figura 3.2.1 corresponde a N dígitos na base $b = 2^{32}$, dados por:

$$(72A45C1B\ 439DA7F1\ 2F439812\ ...\ 458EE023)_{2^{32}}.$$

Para representar números com sinal existem duas abordagens possíveis:

- utilizar um bit de sinal, de modo que um número seja representado por um vetor de digítos e um bit extra. Se este bit for 1, então o número é positivo, caso contrário, o número é negativo.

- utilizar a representação em *complemento de 2*. Nesta representação, se o bit mais significativo de $u = (u_{n-1}...u_0)_b$ for 1, então u é o número positivo $u_{n-1}b^{n-1} + ... + u_0$. Caso contrário, então u é negativo, igual a $-b^n + u_{n-1}b^{n-1} + ... + u_0$. Note que nesta representação, $-u$ é definido como sendo $(\overline{u_{n-1}}...\overline{u_0})_b + 1$, onde $\overline{u_i} = b - 1 - u_i$. Com esta representação, $(u) + (-u)$ naturalmente resulta em zero.

É importante, juntamente com cada número de precisão arbitrária, armazenar a quantidade de palavras utilizada em sua representação. Além disso, também é comum armazenar uma última informação, correspondente à posição da primeira palavra não nula do número.

3.2.1 Soma e Subtração

Nesta seção vamos descrever os algoritmos para soma e substração de números de precisão arbitrária. Estes algoritmos são os métodos clássicos e dados dois números de n palavras, podemos efetuar tanto a soma como a subtração em tempo linear, ou seja, utilizando $O(n)$ operações elementares.

Algoritmo 3.1 Adição de números de precisão arbitrária não negativos

ENTRADA Dois números de precisão arbitrária de n palavras, $u = (u_{n-1}...u_0)_b$ e $v = (v_{n-1}...v_0)_b$.
SAÍDA Um número de precisão arbitrária de $n + 1$ palavras, $w = (w_n...w_0)_b$, tal que $w = u + v$, w_n sendo 0 ou 1.

```
k ← 0
para i = 0 até n − 1 faça
    w_i ← (u_i + v_i + k) (mod b)
    k ← ⌊(u_i + v_i + k)/b⌋
w_n ← k
retorne (w_n...w_0)_b
```

A subtração é basicamente o mesmo algoritmo, alterando alguns sinais das operações envolvidas.

Algoritmo 3.2 Subtração de números de precisão arbitrária não negativos

ENTRADA Dois números de precisão arbitrária de n palavras, $u = (u_{n-1}...u_0)_b$ e $v = (v_{n-1}...v_0)_b$, tais que $u \geq v$.

SAÍDA Um número de precisão arbitrária de n palavras, $w = (w_{n-1}...w_0)_b$, tal que $w = u - v$.

$k \leftarrow 0$

para $i = 0$ até $n - 1$ **faça**

$w_i \leftarrow (u_i - v_i + k) \pmod b$

$k \leftarrow \lfloor (u_i - v_i + k)/b \rfloor$

retorne $(w_{n-1}...w_0)_b$

3.2.2 Multiplicação

A multiplicação é uma operação muito importante, pois além de representar grande parte do consumo de tempo de muitas aplicações, é uma operação que é utilizada como base para outras operações como por exemplo a exponenciação e a divisão. É importante portanto ter um método para medir a complexidade de realizar a multiplicação. Dados dois números de precisão arbitrária de n palavras, denotamos por $M(n)$ o número de operações básicas utilizadas para multiplicar estes dois números.

Algoritmo Clássico

A seguir é descrito o algoritmo clássico para multiplicação de números. Este algoritmo é exatamente aquele ensinado nas escolas, onde os dígitos de cada número são multiplicados individualmente. É possível utilizar uma tabela para auxiliar nesta multiplicação básica. De fato, esta tabela é conhecida como *tabuada*, e são decoradas para que tornem possível a multiplicação de número de vários dígitos na base 10.

Como podemos notar, este algoritmo multiplica exatamente uma vez cada dígito de um dos números por cada dígitos do outro número. Logo, dados dois números de precisão arbitrária de n palavras, temos que $M(n) = O(n^2)$.

Algoritmo Karatsuba

Na década de 50, Kolmogorov fez a conjectura de que a multiplicação de números de n dígitos não poderia ser efetuada com menos que $O(n^2)$ operações. Contudo, em 1960, Karatsuba descobriu um método que realiza esta tarefa utilizando $O(n^{\lg 3})$ operações, onde $\lg 3$ representa o logaritmo de 3 na base 2. O método foi publicado em [KO63]. Uma explicação interessante do método pode ser encontrada em [Kar95].

Seja $R = b^n$ e $d = 2n$. Dados dois números de precisão arbitrária de $2n$ palavras na base b, $u = (u_{d-1}...u_0)_b$ e $v = (v_{d-1}...v_0)_b$, podemos verificar que $u = U_0R + U_1$ e

Algoritmo 3.3 Multiplicação de números de precisão arbitrária positivos

ENTRADA Um número de precisão arbitrária de m palavras $u = (u_{m-1}...u_0)_b$ e um número de n palavras $v = (v_{n-1}...v_0)_b$.

SAÍDA Um número de precisão arbitrária de $m+n$ palavras, $w = (w_{m+n-1}...w_0)_b$, tal que $w = uv$.

para $i = 0$ até $n-1$ **faça**
  $w_i \leftarrow 0$
para $i = 0$ até $n-1$ **faça**
  $k \leftarrow 0$
  se $v_i = 0$ **então**
    $w_{m+i} \leftarrow 0$
  senão
    para $j = 0$ até $m-1$ **faça**
      $t \leftarrow v_i u_j + w_{i+j} + k$
      $w_{i+j} \leftarrow t \pmod b$
      $k \leftarrow \lfloor t/b \rfloor$
    $w_{m+i} \leftarrow k$
retorne $(w_{m+n-1}...w_0)_b$

$v = V_0 R + V_1$, onde U_0 e U_1 representam a primeira e a segunda metade dos dígitos de u respectivamente. Analogamente, definimos V_0 e V_1 como sendo esta separação ao meio dos dígitos de v.

Com isso, temos que

$$uv = U_1 V_1 R^2 + ((U_0 + U_1)(V_0 + V_1) - U_1 V_1 - U_0 V_0)R + U_0 V_0.$$

Assim, são necessárias apenas as multiplicações: $U_0 V_0$, $U_1 V_1$ e $(U_0 + U_1)(V_0 + V_1)$. Além disso são efetuadas algumas somas e deslocamentos. É possível utilizar este algoritmo recursivamente, até um limiar d_0, onde a multiplicação de números de d_0 palavras possa ser efetuada mais eficientemente através do método clássico.

3.2.3 Quadrado

Parece natural que a operação de elevar um número ao quadrado seja mais simples que a operação de multiplicação, esta hipótese pode ser confirmada pelo equação a seguir:

$$\left(\sum_{i=0}^{n-1} u_i b_i \right)^2 = \sum_{i=0}^{n-1} u_i^2 b^{2i} + 2 \sum_{i<j} u_i u_j b^{i+j}. \tag{3.1}$$

Algoritmo 3.4 Multiplicação Karatsuba de números de precisão arbitrária positivos

ENTRADA Um número de precisão arbitrária de m palavras $u = (u_{m-1}...u_0)_b$, um número de n palavras $v = (v_{n-1}...v_0)_b$, $d = \max\{m, n\}$ e um limiar d_0.

SAÍDA Um número de precisão arbitrária de $m + n$ palavras, $w = (w_{m+n-1}...w_0)_b$, tal que $w = uv$.

se $d \leq d_0$ **então**

retorne uv [Use o algortimo 3.3]

$p \leftarrow \lfloor d/2 \rfloor$

$q \leftarrow \lfloor d/2 \rfloor$

$U_0 \leftarrow (u_{q-1}...u_0)_b$

$V_0 \leftarrow (v_{q-1}...v_0)_b$

$U_1 \leftarrow (u_{p+q-1}...u_0)_b$

$V_1 \leftarrow (v_{p+q-1}...v_0)_b$

$U_s \leftarrow U_0 + U_1$

$V_s \leftarrow V_0 + V_1$

Calcule recursivamente U_0V_0, U_1V_1 e U_sV_s

retorne $U_1V_1b^{2q} + (U_sV_s - U_1V_1 - U_0V_0)b^q + U_0V_0$

Através desta equação é possível obter um algoritmo que calcule o quadrado usando $(n^2 + n)/2$ multiplicações elementares (contra n^2 para um algorítmo genérico).

O algoritmo a seguir representa a equação 3.1.

3.2.4 Redução Modular

Nesta seção serão descritos algoritmos para o cálculo do resto da divisão de um dado número inteiro de precisão arbitrária por um inteiro n. Na prática, n é sempre primo, de forma que algoritmos mais eficientes podem ser descritos para realizar esta operação. Em resumo, a operação consiste em realizar uma divisão simples, com a obtenção do resto. Mas existem possibilidades mais eficientes para certos casos, de modo que serão tratados nesta seção algoritmos como o método de Barrett, o método de Montgomery, além de casos de primos específicos e o caso de redução módulo múltiplos primos.

Método de Barrett

Quando é preciso realizar várias reduções modulares utilizando um mesmo módulo é possível utilizar o método de Barrett. Para isso, seja N um inteiro de precisão arbitrária de n palavras. Definimos $R(N) = \lfloor b^{2n}/N \rfloor$. Seja u um inteiro de precisão arbitrária de no máximo $2n$ palavras, então temos que $q = \lfloor u/N \rfloor$, ou seja,

Algoritmo 3.5 Quadrado de um número de precisão arbitrária positivo

ENTRADA Um número de precisão arbitrária de n palavras $u = (u_{n-1}...u_0)_b$.
SAÍDA Um número de precisão arbitrária de $2n$ palavras, $w = (w_{2n-1}...w_0)_b$, tal que $w = u^2$.

```
para i = 0 até 2n − 1 faça
    w_i ← 0
para i = 0 até n − 1 faça
    t ← u_i^2 + w_{2i}
    w_{2i} ← t (mod b)
    k ← ⌊t/b⌋
    para j = i + 1 até n − 1 faça
        t ← 2u_i u_j + w_{i+j} + k
        w_{i+j} ← t (mod b)
        k ← ⌊t/b⌋
    w_{m+i} ← k
retorne (w_{2n−1}...w_0)_b
```

$$q = \left\lfloor \frac{\frac{u}{b^{n-1}} \frac{b^{2n}}{N}}{b^{n+1}} \right\rfloor .$$

Logo, q pode ser aproximado por

$$\hat{q} = \left\lfloor \frac{\lfloor \frac{u}{b^{n-1}} \rfloor R(N)}{b^{n+1}} \right\rfloor .$$

É fácil mostrar que $q - 2 \leq \hat{q} \leq q$, além disso, sabe-se que $\hat{q} = q$ em 90% dos casos, enquanto $\hat{q} = q - 2$ em apenas 1% dos casos.

Portanto, para realizar a redução modular basta calcular $\hat{u} = u - \hat{q}N$. Se $\hat{u} > N$, então bastam no máximo 2 subtrações para obter um valor menor que N. Com isso, quando a operação de redução modular for realizada repetidas vezes, utilizando o mesmo N, é possível pré-calcular $R(N)$, tornando fácil a obtenção do resto da divisão por N.

Para realizar o pré-cálculo de $R(N)$, podemos utilizar o algoritmo 3.7:

Método de Montgomery

Montgomery introduziu uma forma diferente de representar números de $\mathbb{Z}/N\mathbb{Z}$ tal que a mutiplicação pode ser efetuada mais eficientemente [Mon85]. De forma resumida, é utilizado um automorfismo de $\mathbb{Z}/N\mathbb{Z}$ que permite que a multiplicação seja concluída sem a utilização de divisão bruta.

Algoritmo 3.6 Método de Barrett para redução modular de números de precisão arbitrária positivos

ENTRADA Um número de precisão arbitrária de $2n$ palavras $u = (u_{2n-1}...u_0)_b$ e um número de precisão arbitrária de n palavras $N = (N_{n-1}...N_0)_b$, tal que $N_{n-1} \neq 0$ e o valor pré-calculado $R(N) = \lfloor b^{2n}/N \rfloor$.

SAÍDA Um número de precisão arbitrária de n palavras, $r = (r_{n-1}...r_0)_b$, tal que $r = u \pmod N$.

$\hat{q} \leftarrow \lfloor \lfloor (u/b^{n-1}) \rfloor R(N)/b^{n+1} \rfloor$
$r_1 \leftarrow u \pmod{b^{n+1}}$
$r_2 \leftarrow (\hat{q}N) \pmod{b^{n+1}}$
$r \leftarrow r_1 - r_2$
se $r < 0$ **então**
  $r \leftarrow r + b^{n+1}$
enquanto $r \geq N$ **faça**
  $r \leftarrow r - N$
retorne $(r_{n-1}...r_0)_b$

Algoritmo 3.7 Pré-cálculo de $R(N)$

ENTRADA Um número de precisão arbitrária de n palavras $N = (N_{n-1}...N_0)_b$.

SAÍDA Um número de precisão arbitrária de $n + 2$ palavras, $R = (R_{n+1}...R_0)_b$, tal que $R = \lfloor b^{2n}/N \rfloor$.

$R \leftarrow b^n$
enquanto $R > s$ **faça**
  $s \leftarrow R$
  $R \leftarrow 2R - \lfloor N \lfloor R^2/b^n \rfloor / b^n \rfloor$
$t \leftarrow b^{2n} - NR$
enquanto $t < 0$ **faça**
  $R \leftarrow R - 1$
  $t \leftarrow t + N$
retorne $(R_{n+1}...R_0)_b$

Definição 3.2.1. Seja N um inteiro positivo. Um *sistema completo de restos* é definido como sendo o conjuto composto por elementos $a \in \mathbb{Z}/N\mathbb{Z}$ tal que $(a, N) = 1$.

Primeiramente, é fácil notar que dados N e R primos entre si, o sistema completo de restos módulo N, $S \in \mathbb{Z}/N\mathbb{Z}$, pode ser permutado pela multiplicação por R. Ou seja, ao multiplicarmos cada elemento $s \in S$ por R, obtemos o próprio conjunto S. É fácil mostrar que esta permutação é um automorfismo, de modo que efetuar a multiplicação $a.b \in S$ é o mesmo que calcular $(R.a)(R.b) \in RS$. Assim, se a multiplicação em RS é mais eficiente que em S, a exponenciação de números de precisão arbitrária pode ter um grande ganho se utilizarmos este automorfismo para mapear elementos de S em RS, efetuarmos todas as multiplciações necessárias em RS e finalmente voltarmos para o conjunto S original.

Definição 3.2.2. Sejam R e N primos entre si, tais que $R > N$. A *representação de Montgomery* de $x \in [0, N-1]$ é $[x] = (xR) \pmod N$. A *redução de Montgomery* de $u \in [0, RN-1]$ é $\text{REDC}\,(u) = (uR^{-1}) \pmod N$.

Algoritmo 3.8 Redução de Montgomery para multiplicação de inteiros

ENTRADA Um número de precisão arbitrária de n palavras $N = (N_{n-1}...N_0)_b$, tal que $(N, b) = 1$, $R = b^n$, $N' = (-N^{-1}) \pmod b$ e um número de precisão arbitrária de $2n$ palavras $u = (u_{2n-1}...u_0)_b$, tal que $u < RN$.
SAÍDA Um número de precisão arbitrária de n palavras, $t = (t_{n-1}...t_0)_b$, tal que $t = \text{REDC}\,(u) = (uR^{-1}) \pmod N$.
$(t_{2n-1}...t_0)_b \leftarrow (u_{2n-1}...u_0)_b$
para $i = 0$ até $n-1$ **faça**
  $k_i \leftarrow (t_i N') \pmod b$
  $t \leftarrow t + k_i N b^i$
$t \leftarrow t/R$
se $t \geq N$ **então**
  $t \leftarrow t - N$
retorne $(t_{n-1}...t_0)_b$

Primos Específicos

Quando estamos efetuando redução modular utilizando um número primo p específico, é possível aproveitar alguma característica que primo para acelerar os cálculos. Nesta seção serão apresentados algoritmos que aproveitam de alguma forma uma característica específica de um número primo, mas é preciso ter em vista que tais características representam uma potencial vantagem também a um atacante.

Um *primo de Mersenne* é definido como sendo um número primo da forma $p_k = 2^k - 1$, para algum inteiro positivo k. Um primo desta forma, é representado através de uma sequência de 1's na base 2. Por exemplo, para $k = 3$, temos $p_3 = 7 = (111)_2$.

Seja x um número inteiro tal que $0 \leq x < p_k^2$, então podemos escrever $x = x_1 2^k + x_0$, com $0 \leq x_i < p_k$. Como $2^k \equiv 1 \pmod{p)_k}$, então $x \equiv x_1 + x_0 \pmod{p)_k}$. Isto é, a redução modular para primos de Mersenne se resume a uma simples operação de soma modular.

Os valores de k, menores que 1000, tal que p_k é primos são: 2, 3, 5, 7, 13, 17, 19, 31, 89, 107, 127, 521 e 607. Com isso, podemos notar que existe um intervalo de 2^{128} a 2^{520}, em que não temos nenhum primo de Mersenne. Como este intervalo é importante para criptossistemas baseados em curvas elípticas, algumas generalizações deste conceito foram criadas. Crandall, por exemplo, utilizou primos da forma $2^k - c$, para $c > 0$ suficientemente pequeno [Cra92]. De fato, podemos considerar primos da forma $p = b^k + c$, onde b é uma potência de 2 e $|c|$ é pequeno. Assim, temos a seguinte relação:

$$x \equiv ((x \pmod{b^k}) - c\lfloor x/b^k \rfloor) \pmod{p}.$$

Com esta equação, podemos derivar o algoritmo 3.9:

Algoritmo 3.9 Redução rápida para primos específicos

ENTRADA Um inteiro x e um primo $p = b^k + c$, tal que $|c| < b^{k-1} - 1$.
SAÍDA O resíduo $r \equiv x \pmod{p}$.

```
q_0 ← ⌊x/b^k⌋
r_0 ← x − q_0 b^k
r ← r_0
i = 0
c' ← |c|
enquanto q_i > 0 faça
    q_{i+1} ← ⌊q_i c'/b^k⌋
    r_{i+1} ← q_i c' − q_{i+1} b^k
    r ← r + (−1)^{i+1} r_i
    i ← i + 1
enquanto r ≥ p faça
    r ← r − p
enquanto r < 0 faça
    r ← r + p
retorne r
```

Se x for um número de precisão arbitrária de no máximo $2k$ dígitos na base b e se $|c| \leq b^{k/2} - 1$, então no máximo 3 multiplicações por c' são necessárias para obter o resíduo módulo p. Também é possível considerar um número primo p que divide $N = b^k + c$. Se

o fator N/p for pequeno o suficiente podemos realizar os cálculos para obter o resíduo módulo N e posteriormente realizar uma redução módulo p.

Redução Módulo Múltiplos Primos

A idéia de *árvore de restos* ([MB72] e [Ber04]) pode ser usada para tornar mais eficiente reduções modulares de x módulo um número determinado de primos. Sejam p_1, p_2, p_3 e p_4 números primos. A idéia é realizar primeiramente uma redução módulo $p_1p_2p_3p_4$. Em seguida, são realizadas duas reduções, módulo p_1p_2 e módulo p_3p_4 respectivamente. Finalmente são realizadas as reduções módulo cada primo individualmente. A complexidade deste algoritmo é $n(\lg n)^{2+o(1)}$, onde n é o total de bits de x, p_1, ..., p_k.

3.2.5 Divisão

O próximo algoritmo é uma versão refinada do método clássico de divisão [Knu97], ensinado nas escolas:

Algoritmo 3.10 Algoritmo clássico para divisão de números de precisão arbitrária positivos

---

ENTRADA Um número de precisão arbitrária de $(m+n)$ palavras, $u = (u_{m+n-1}...u_0)_b$ e um número de precisão arbitrária de n palavras, $v = (v_{n-1}...v_0)_b$.

SAÍDA Um número de precisão arbitrária de $(m+1)$ palavras, $q = (q_m...q_0)_b$ e o número de precisão arbitrária de n palavras, $r = (r_{n-1}...r_0)_b$, tal que $u = vq + r$.

$u_{m+n} \leftarrow 0$
$d \leftarrow 1$
enquanto $v_{n-1} < b/2$ **faça**
  $v \leftarrow 2v$
  $u \leftarrow 2u$
  $d \leftarrow 2d$
  para $i = m$ até 0 **faça**
    $\hat{q} \leftarrow \min\left(\lfloor (u_{i+n}b + u_{i+n-1})/v_{n-1} \rfloor, b-1\right)$
    enquanto $\hat{q}(v_{n-1}b + v_{n-2}) > (u_{i+n}b^2 + u_{i+n-1}b + u_{i+n-2})$ **faça**
      $\hat{q} \leftarrow \hat{q} - 1$
    $(u_{i+n}...u_i)_b \leftarrow (u_{i+n}...u_i)_b - \hat{q}(v_{n-1}...v_0)_b$
    se $(u_{i+n}...u_i)_b < 0$ **então**
      $\hat{q} \leftarrow \hat{q} - 1$
      $(u_{i+n}...u_i)_b \leftarrow (u_{i+n}...u_i)_b + (0v_{n-1}...v_0)_b$
    $q_i \leftarrow \hat{q}$
  $r \leftarrow u/d$
retorne (q, r)

O algoritmo 3.10 possui complexidade $O((n+m)^2)$. Como já vimos anteriormente existem algoritmos mais eficientes para obter o resto da divisão de números de precisão arbitrária. Por isso, este algoritmo deve ser usado apenas em alguns casos. Nos criptossistemas de nosso interesse, estaremos interessados normalmente em realizar reduções modulares de forma eficiente, podendo aproveitar certas informações como por exemplo uma característica especial do módulo ou então valores que possam ser pré-calculados. Como vimos anteriormente, o algoritmo 3.6 precisa receber como parâmetro o valor $R(N)$. Este cálculo é uma divisão clássica que pode ser efetuada apenas uma vez.

3.2.6 Máximo Divisor Comum

Os algoritmos apresentados nesta seção podem ser encontrados em [Coh00, Ler97]. Os algortimos de Euclides, Lehmer e o binário estendido, são opções eficientes para o cálculo do MDC entre dois números de precisão arbitrária, x e N. Mas além de retornar o MDC, estes algoritmos também retornam dois valores, u e v, tais que $xu + Nv = d$, onde d é o MDC entre x e N.

Algoritmo 3.11 Algoritmo Estendido de Euclides para cálculo de MDC de números positivos

---

ENTRADA Dois números de precisão arbitrária x e N, tal que $x < N$.
SAÍDA Inteiros de precisão arbitrárias (u, v, d), tais que $xu+Nv = d$ e $d = MDC(x, N)$.

$A \leftarrow N$
$B \leftarrow x$
$U_A \leftarrow 0$
$U_B \leftarrow 1$
enquanto $B \neq 0$ **faça**
  $q \leftarrow \lfloor A/B \rfloor$
  $A \leftarrow B$
  $B \leftarrow A - qB$
  $U_A \leftarrow U_B$
  $U_B \leftarrow U_A - qU_B$
$d \leftarrow A$
$u \leftarrow U_A$
$v \leftarrow (d - xu)/N$
retorne (u, v, d)

No algoritmo 3.11, podemos ver facilmente que $xU_A + NV_A = A$ e $xU_B + NV_B = B$ durante toda a execução do algoritmo. Além disso, $|U_A|$ e $|U_B|$ são sempre menores que N/A, enquanto $|V_A|$ e $|V_B|$ são sempre menores que x/A. A complexidade deste algoritmo é $O(\lg^2 N)$. Mas a maior parte do consumo de tempo deste algoritmo está no cálculo de

$q = \lfloor A/B \rfloor$. Como na maioria das vezes temos que $q = 1$, podemos utilizar este valor sempre que necessário, ao custo de um número maior de rodadas do algoritmo.

A idéia do algoritmo de Lehmer é portanto aproximar o valor de $\lfloor A/B \rfloor$ com os bits mais significativos de A e B, atualizando-os quando necessário através da equação:

$$\begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} \alpha & \beta \\ \alpha' & \beta' \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix}.$$

Os inteiros α, α', β e β', podem ser calculados pelo algoritmo 3.13. Este algoritmo foi melhorado por Collins [Col80], Jebelean [Jeb93] e Lercier [Ler97].

Algoritmo 3.12 Algoritmo Estendido de Euclides de Lehmer para cálculo de MDC de números positivos

---

ENTRADA Dois números de precisão arbitrária x e N na base $b = 2^\ell$, tal que $x < N$.
SAÍDA Inteiros de precisão arbitrárias (u, v, d), tais que $xu + Nv = d$ e $d = MDC(x, N)$.

$A \leftarrow N$
$B \leftarrow x$
$U_A \leftarrow 0$
$U_B \leftarrow 1$
enquanto $B \neq 0$ **faça**
  Calcule α, α', β, β' usando o algoritmo 3.13
  $A \leftarrow \alpha A + \beta B$
  $B \leftarrow \alpha' A - \beta' B$
  $U_A \leftarrow \alpha U_A + \beta U_B$
  $U_B \leftarrow \alpha U_A + \beta U_B$
Calcule (u, v, d) usando o algortimo 3.11
$u \leftarrow uU_A + vU_B$
$v \leftarrow (d - xu)/N$
retorne (u, v, d)

Para calcular o MDC entre dois números A e B, podemos repetir os seguintes passos:

1. Se A e B são pares, então
$$(A, B) = 2(A/2, B/2)$$
2. Se A é par e B é ímpar, então
$$(A, B) = (A/2, B)$$
3. Analogamente, se B é par e A é ímpar, então
$$(A, B) = (A, B/2)$$

Algoritmo 3.13 MDC parcial para números de precisão arbitrária positivos na base $b = 2^\ell$

ENTRADA Dois números de precisão arbitrária positivos A e B.
SAÍDA Inteiros de precisão arbitrárias $(\alpha, \alpha', \beta, \beta')$.

$\hat{A} \leftarrow \left\lfloor \frac{A}{2^{\max(|A|_2 - \ell, 0)}} \right\rfloor$
$\hat{B} \leftarrow \left\lfloor \frac{B}{2^{\max(|B|_2 - \ell, 0)}} \right\rfloor$
$\alpha \leftarrow 1$, $\beta \leftarrow 0$, $\alpha' \leftarrow 0$, $\beta' \leftarrow 1$ e $T \leftarrow 0$
se $\hat{B} \neq 0$ **então**
  $q \leftarrow \lfloor \hat{A}/\hat{B} \rfloor$ e $T \leftarrow \hat{A} \pmod{\hat{B}}$
se $T \geq 2^{\ell/2}$ **então**
  enquanto 1 **faça**
    $q' \leftarrow \lfloor \hat{B}/T \rfloor$ e $T' \leftarrow \hat{B} \pmod{T}$
    se $T' < 2^{\ell/2}$ **então**
      pare
    $\hat{A} \leftarrow \hat{B}$ e $\hat{B} \leftarrow T$
    $T \leftarrow \alpha - q\alpha'$, $\alpha \leftarrow \alpha'$ e $\alpha' \leftarrow T$
    $T \leftarrow \beta - q\beta'$, $\beta \leftarrow \beta'$ e $\beta' \leftarrow T$
    $T \leftarrow T'$ e $q \leftarrow q'$
se $\beta = 0$ **então**
  $\alpha \leftarrow 0$, $\beta \leftarrow 1$, $\alpha' \leftarrow 1$ e $\beta' \leftarrow -\lfloor A/B \rfloor$
  retorne $(\alpha, \beta, \alpha', \beta')$
$\hat{A} \leftarrow \left\lfloor \frac{A}{2^{\max(|A|_2 - 2\ell, 0)}} \right\rfloor$
$\hat{B} \leftarrow \left\lfloor \frac{B}{2^{\max(|B|_2 - 2\ell, 0)}} \right\rfloor$
$A \leftarrow \alpha\hat{A} + \beta\hat{B}$
$B \leftarrow \alpha'\hat{A} - \beta'\hat{B}$
$\hat{A} \leftarrow \left\lfloor \frac{\hat{A}}{2^{\max(|\hat{A}|_2 - \ell, 0)}} \right\rfloor$
$\hat{B} \leftarrow \left\lfloor \frac{\hat{B}}{2^{\max(|\hat{B}|_2 - \ell, 0)}} \right\rfloor$
$T \leftarrow 0$
se $\hat{B} \neq 0$ **então**
  $q \leftarrow \lfloor \hat{A}/\hat{B} \rfloor$ e $T \leftarrow \hat{A} \pmod{\hat{B}}$
se $T \geq 2^{\ell/2}$ **então**
  enquanto 1 **faça**
    $q' \leftarrow \lfloor \hat{B}/T \rfloor$ e $T' \leftarrow \hat{B} \pmod{T}$
    se $T' < 2^{\ell/2}$ **então**
      retorne $(\alpha, \beta, \alpha', \beta')$
    $T \leftarrow \alpha - q\alpha'$, $\alpha \leftarrow \alpha'$ e $\alpha' \leftarrow T$
    $T \leftarrow \beta - q\beta'$, $\beta \leftarrow \beta'$ e $\beta' \leftarrow T$
    $T \leftarrow T'$ e $q \leftarrow q'$
retorne $(\alpha, \beta, \alpha', \beta')$

4. Se A e B são ímpares, então $|A - B|$ é par e então

$$(A, B) = (\max(A, B), |A - B|/2)$$

Como a divisão por potência de 2 é eficientemente calculada através de um deslocamento, podemos usar esta idéia para derivar um algoritmo mais eficiente para o cálculo do MDC entre dois números. Em cada passo anterior, foi considerado apenas o caso em que o 2 é fator comum entre A e B, mas é possível adaptar esta idéia para em cada passo, eliminar qualquer potência de 2 que seja comum a ambos os fatores. Por exemplo, para calcular o MDC entre 12 e 32, podemos inicialmente dividir ambos os elementos por 4, de maneira que $(32, 12) = 4(8, 3)$. Com isso, pelo passo 2, podemos concluir que $(32, 12) = 4(3, 1) = 4$. O algortimo 3.14 implementa esta idéia.

3.2.7 Teorema Chinês dos Restos

Suponha que desejamos calcular a solução para o seguinte sistema:

$$\begin{cases} x \equiv x_1 \pmod{n_1} \\ x \equiv x_2 \pmod{n_2} \\ \quad \vdots \\ x \equiv x_k \pmod{n_k} \end{cases},$$

onde os valores de n_i são primos entre si e os valores de x_i são previamente estabelecidos. Existe uma única solução módulo $N = n_1...n_k$. De fato, seja $N_i = N/n_i$ e $a_i \equiv N_i^{-1} \pmod{n_i}$. A solução é dada por

$$x \equiv a_1 N_1 x_1 + ... + a_k N_k x_k \pmod{N}.$$

O algoritmo 3.15 calcula esta solução eficientemente.

Este algoritmo pode ser facilmente estendido para operar com elementos de um anel polinomial.

3.2.8 Raíz Quadrada

O *método de Newton* pode ser utilizado para encontrar soluções para uma função, ou seja, é capaz de encontrar valores de x tais que $f(x) = 0$. Para calcular raíz quadrada de um número inteiro de precisão arbitrária, podemos utilizar esta idéia, onde a função em questão corresponde a $f(x) = x^2 - u$. O método de Newton calcula a sequência $x_{i+1} = x_i - f(x_i)/f'(x_i)$, de modo que os elementos desta sequência estejam cada vez mais próximos da solução desejada. Assim, o algoritmo 3.16 calcula estes elementos até um momento em que para algum valor de k, ele retorna $x_k = \sqrt{u}$.

Algoritmo 3.14 Algoritmo Estendido de Euclides Binário para cálculo de MDC de números positivos

ENTRADA Dois números de precisão arbitrária x e N, tal que $x < N$.
SAÍDA Inteiros de precisão arbitrárias (u, v, d), tais que $xu+Nv = d$ e $d = MDC(x, N)$.

$q \leftarrow \lfloor N/x \rfloor$
$T \leftarrow N \pmod{x}$
$N \leftarrow x$
$x \leftarrow T$
se $x = 0$ **então**
  $u \leftarrow 0$, $v \leftarrow 1$, $d \leftarrow N$
  retorne (u, v, d)
$k \leftarrow 0$ e $f \leftarrow 0$
enquanto $N \equiv 0 \pmod{2}$ e $x \equiv 0 \pmod{2}$ **faça**
  $k \leftarrow k + 1$, $N \leftarrow N/2$ e $x \leftarrow x/2$
se $x \equiv 0 \pmod{2}$ **então**
  $T \leftarrow x$, $x \leftarrow N$, $N \leftarrow T$ e $f \leftarrow 1$
$U_B \leftarrow 1$, $A \leftarrow N$, $B \leftarrow x$ e $v' \leftarrow x$
se $N \equiv 1 \pmod{2}$ **então**
  $U_A \leftarrow 0$ e $t' \leftarrow -x$
senão
  $U_A \leftarrow (1 + x)/2$ e $t' \leftarrow n/2$
enquanto $t' \neq 0$ **faça**
  se $t' > 0$ **então**
    $U_B \leftarrow U_A$ e $A \leftarrow t'$
  senão
    $B \leftarrow x - U_A$ e $v' \leftarrow -t'$
  $U_A \leftarrow U_B - B$ e $t' \leftarrow A - v'$
  se $U_A > 0$ **então**
    $U_A \leftarrow U_A + x$
  enquanto $t' \equiv 0 \pmod{2}$ e $t' \neq 0$ **faça**
    $t' \leftarrow t'/2$
    se $U_A \equiv 0 \pmod{2}$ **então**
      $U_A \leftarrow U_A/2$
    senão
      $U_A \leftarrow (U_A + x)/2$
$u \leftarrow U_B$, $v \leftarrow (A - xu)/N$ e $d \leftarrow 2^k A$
se $f = 1$ **então**
  $T \leftarrow u$, $u \leftarrow v$ e $v \leftarrow T$
$u \leftarrow u - vq$
retorne (u, v, d)

Algoritmo 3.15 Teorema Chinês dos Restos

ENTRADA Números inteiros positivos primos entre si, $n_1, ..., n_k$ e inteiros $x_1, ..., x_k$.
SAÍDA Um inteiro x tal que $x \equiv x_i \pmod{n_i}$.

$N \leftarrow n_1$
$x \leftarrow x_1$
para $i = 2$ até k **faça**
  Calcule u e v tais que $un_i + vN = 1$ usando algum dos algoritmos anteriores
  $x \leftarrow un_i x + vNx_i$
  $N \leftarrow Nn_i$
  $x \leftarrow n \pmod{N}$
retorne x

Algoritmo 3.16 Raíz quadrado de inteiros de precisão arbitrária

ENTRADA Um número de precisão arbitrária positivo de n palavras, $u = (u_{n-1}...u_0)_b$.
SAÍDA Um inteiro positivo v tal que $v = \lfloor \sqrt{u} \rfloor$.

$t \leftarrow 2^{\lceil \lg u/2 \rceil}$
enquanto $t \neq v$ **faça**
  $v \leftarrow t$
  $t \leftarrow \lfloor (v + \lfloor u/v \rfloor)/2 \rfloor$
retorne v

3.2.9 Exponenciação

Nesta seção serão descritos algoritmos para o cálculo da exponenciação de elementos de um grupo multiplicativo. Esta operação tem grande importância para os criptossistemas em que estamos interessados, pois dados $x, y \in (G, \times)$, não existe um algoritmo eficiente para calcular o valor de $n \in \mathbb{Z}$ tal que $x^n = y$. De fato, determinar n é conhecido como o *problema do logaritmo discreto*. Normalmente, a exponenciação representa a maior parte do consumo de tempo de criptossistemas baseados neste problema, mas além disso, existem várias primitivas que têm como base a exponenciação, entre elas a raíz quadrada e alguns testes de primalidade. Com isso, é importante que a exponenciação possa ser efetuada eficientemente. O algoritmo básico para o cálculo da exponenciação utiliza principalmente operações de multiplicação e quadrado, mas quando estamos em um grupo aditivo, como é o caso de curvas elípticas, podemos adaptar as mesmas idéias para utilizar respectivamente operações de soma e duplicação.

Podemos ter três situações distintas, sendo que em cada uma delas existe uma solução mais adequada:

- Se o valor de x e n variam constantemente, o ideal é utilizar uma abordagem genérica para calcular x^n.

- Se x é utilizado muitas vezes para diversos valores de n, então podemos pré-calcular alguns valores que permitem um ganho no cálculo de x^n cada vez que for necessário.
- Se o expoente n for mantido constante, então é possível procurar por cadeias de adição pequenas, que permitam um menor número de rodadas do algortimo para o cálculo de x^n.

O método mais trivial para o cálculo de x^n requer $n-1$ multiplicações, mas existem formas mais eficientes para calcular a exponenciação. De fato, para calcular x^{2^k}, podemos utilizar k quadrados, obtendo a sequência $x^2, x^4, x^8, ..., x^{2^k}$. Com esta idéia, podemos calcular a exponenciação usando $O(\lg n)$ multiplicações. Por exemplo, se $n = 10$, calcular x^{10} se resume a calcular o produto x^8x^2. Portanto, para calcular x^n, basta considerar a representação binária de n e onde encontrarmos um bit igual a um, acumulamos a potência correspondente em uma variável. O algoritmo 3.17 representa esta idéia.

Algoritmo 3.17 Método do quadrado e multiplicação

ENTRADA Um elemento $x \in \mathbb{F}_{2^m}$ e um inteiro não negativo $n = (n_{\ell-1}...n_0)_2$.
SAÍDA $x^n \in \mathbb{F}_{2^m}$.

```
y ← 1
i ← ℓ − 1
enquanto i ≥ 0 faça
    y ← y²
    se n_i = 1 então
        y ← x × y
    i ← i − 1
retorne y
```

Este método também é conhecido como *exponenciação binária da esquerda para a direita*. Outra abordagem possível é analisar os bits de n na direção inversa, da direita para a esquerda, como mostra o algoritmo 3.18.

É interessante notar que nesta segunda abordagem, é preciso uma variável z para acumular as potências x^2, x^4, x^8, etc. Com isso, mais memória é utilizada neste algoritmo, mas por outro lado, estes cálculos podem ser feitos em paralelo, o que torna possível uma implementação mais eficiente em hardware.

O número de quadrados que precisam ser calculados é em ambos os casos igual a $O(\lg n)$, mas o número de multiplicações depende diretamente da quantidade de bits iguais a um na representação binária de n. Esta quantidade é denotada por $\nu(n)$ e é chamada de *peso de Hamming de n*.

Exemplo 3.2.3. Vamos utilizar o algoritmo 3.17 e 3.18 para calcular x^{314}. A representação binária de 314 é $(100111010)_2$. Com isso, os algoritmos executam em 9 rodadas,

Algoritmo 3.18 Exponenciação binário da direita para a esquerda

ENTRADA Um elemento $x \in \mathbb{F}_{2^m}$ e um inteiro não negativo $n = (n_{\ell-1}...n_0)_2$.
SAÍDA $x^n \in \mathbb{F}_{2^m}$.

```
y ← 1
z ← x
i = 0
enquanto i ≤ ℓ − 1 faça
  se n_i = 1 então
    y ← y × z
  z ← z^2
  i ← i + 1
retorne y
```

como podemos ver nas tabelas que seguem:

Tabela 3.1: **Algoritmo 3.17**

i	8	7	6	5	4	3	2	1	0
n_i	1	0	0	1	1	1	0	1	0
y	x	x^2	x^4	x^9	x^{19}	x^{39}	x^{78}	x^{157}	x^{314}

Tabela 3.2: **Algoritmo 3.18**

i	0	1	2	3	4	5	6	7	8
n_i	0	1	0	1	1	1	0	0	1
z	x	x^2	x^4	x^8	x^{16}	x^{32}	x^{64}	x^{128}	x^{256}
y	x	x^2	x^2	x^{10}	x^{26}	x^{58}	x^{58}	x^{58}	x^{314}

Estes dois algoritmos executam de forma parecida, pois em toda rodada é calculado um quadrado, e, dependendo se o bit do expoente for zero ou um, é efetuada também uma multiplicação. Desta forma, um atacante pode deduzir informações sobre este expoente, dado o tempo de execução de cada rodada. Existem vários ataques parecidos, que aproveitam esta característica destes algoritmos para obter informações que não deveriam ter acesso. Uma alternativa a estes algoritmos é o *método da escada de Montgomery*. Neste método, um quadrado e uma multiplicação são cálculados em todas as rodadas do algoritmo.

Este algoritmo é claramente menos eficiente que os dois anteriores, mas por outro lado resolve a vulnerabilidade que existia.

Exemplo 3.2.4. Vamos calcular novamente x^{314}, agora usando o Algoritmo 3.19:

Algoritmo 3.19 Exponenciação pelo método da escada de Montgomery

ENTRADA Um elemento $x \in \mathbb{F}_{2^m}$ e um inteiro não negativo $n = (n_{\ell-1}...n_0)_2$.
SAÍDA $x^n \in \mathbb{F}_{2^m}$.

$x_1 \leftarrow x$
$x_2 \leftarrow x^2$
para $i = \ell - 2$ até 0 **faça**
  se $n_i = 0$ **então**
    $x_1 \leftarrow x_1^2$
    $x_2 \leftarrow x_1 \times x_2$
  senão
    $x_1 \leftarrow x_1 \times x_2$
    $x_2 \leftarrow x_2^2$
retorne y

Tabela 3.3: **Algoritmo 3.19**

i	7	6	5	4	3	2	1	0
n_i	0	0	1	1	1	0	1	0
(x_1, x_2)	(x^2, x^3)	(x^4, x^5)	(x^9, x^{10})	(x^{19}, x^{20})	(x^{39}, x^{40})	$^{78}, x^{79})$	(x^{157}, x^{158})	(x^{314}, x^{315})

Outra alternativa que existe é processar mais bits do expoente por rodada do algoritmo, baseando-se em uma tabela pré-computada de potências ímpares de x. A idéia é representar o expoente na base $b = 2^k$ e utilizar uma função σ, tal que $\sigma(0) = (k, 0)$ e $\sigma(m) = (s, u)$, onde $m = 2^s u$.

O Algoritmo 3.20 executa usando $O(\lg n + \lg n(1 + o(1)) \lg n / \lg \lg n)$ operações elementares. A eficiência ótima é obtida quando k for o menor valor tal que

$$\lg n \leq \frac{k(k+1)2^{2k}}{2^{k+1} - k - 2}.$$

Assim, podemos determinar qual o melhor valor de k para um certo valor de n. A tabela 3.4 mostra que, para um dado intervalo de tamanho de n, existe um valor ideal para k:

Tabela 3.4: **Valor ótimo de** k

k	1	2	3	4	5	6	7
$\lg n$	$[1, 9]$	$[10, 25]$	$[26, 70]$	$[70, 197]$	$[197, 539]$	$[539, 1434]$	$[1434, 3715]$

Exemplo 3.2.5. Seja $n = 11957708941720303968251$, a representação binária de n é dada por:

Algoritmo 3.20 Exponenciação 2^k-ária da esquerda para a direita

ENTRADA Um elemento $x \in \mathbb{F}_{2^m}$, um parâmetro $k \geq 1$ e um inteiro não negativo $n = (n_{\ell-1}...n_0)_{2^k}$ e os valores pré-calculados $x^3, x^5, ..., x^{2^k-1}$.
SAÍDA $x^n \in \mathbb{F}_{2^m}$.

```
y ← 1
i ← ℓ − 1
(s, u) ← σ(n_i)
enquanto i ≥ 0 faça
  para j = 1 até k − s faça
    y ← y^2
  y ← y × x^u
  para j = 1 até s faça
    y ← y^2
  i ← i − 1
retorne y
```

$$(10100010000011101010001100000111111101011001011110111000000001111111111011)_2.$$

Ou seja, n possui 74 bits e de acordo com a tabela anterior, temos que $k = 4$. Se denotarmos por M o número de multiplicações e S o número de somas necessários para calcular x^n, temos que o algoritmo 3.20 utiliza $7M + S$ operações para o pré-cálculo e $17M + 72S$ operações para a exponenciação. Em comparação, o algoritmo 3.17 utiliza $39M + 73S$ operações. Como a multiplicação tem um custo consideravelmente superior ao da soma, temos um grande ganho com o algoritmo 3.20.

A idéia anterior permite dividir o expoente em janelas de tamanho k. Agora, vamos apresentar uma idéia em que deslisamos esta janela pela representação binária do expoente, pulando sequências de zeros. Por exemplo, seja $n = 334 = (101001110)_2$. Podemos usar uma janela de tamanho 3, calculando então as potências x^3, x^5 e x^7. Os valores sucessivos de y, computados pelo algoritmo 3.20 são 1,x^5, x^{10}, x^{20}, x^{40}, x^{41}, x^{82}, x^{164} e x^{334}. Utilizando as janelas indicadas a seguir:

$$334 = (\underbrace{101}\,\underbrace{001}\,\underbrace{110})_2.$$

Mas, usando o algoritmo 3.21, obtemos a sequência de valores de y dadas por 1,x^5, x^{10}, x^{20}, x^{40}, x^{80}, x^{160}, x^{167} e x^{334}. Utilizando uma rodada a menos. Isto acontece porque utilizamos a seguinte distribuição de janelas:

$$334 = (\underbrace{101}\,00\,\underbrace{111}\,0)_2.$$

Algoritmo 3.21 Método da exponenciação por janela

ENTRADA Um elemento $x \in \mathbb{F}_{2^m}$, um parâmetro $k \geq 1$ e um inteiro não negativo $n = (n_{\ell-1}...n_0)_2$.
SAÍDA $x^n \in \mathbb{F}_{2^m}$.

```
y ← 1
i ← ℓ − 1
enquanto i ≥ 0 faça
  se n_i = 0 então
    y ← y^2
    i ← i − 1
  senão
    s ← max i − k + 1, 0
    enquanto n_s = 0 faça
      s ← s + 1
    para h = 1 até i − s + 1 faça
      y ← y^2
      u ← (n_i...n_s)_2
      y ← y × x^u
      i ← s − 1
retorne y
```

Outra idéia interessante é permitir que seja utilizado o inverso de x, de forma que se tivermos uma representação do expoente que tenha digitos negativos, então mutiplicamos por x^{-1}. Um exemplo simples é a exponenciação x^{2^k-1}. Normalmente são necessários $k-1$ multiplicações e $k-1$ quadrados. Mas é possível obter o mesmo resultado utilizando k quadrados e apenas uma multiplicação pelo inverso de x. Com essa idéia, podemos definir:

Definição 3.2.6. A *representacão com sinal* de um inteiro n é dada por

$$n = \sum_{i=0}^{\ell-1} n_i b^i, \text{com } |n_i| < b.$$

Por exemplo, podemos escolher a base $b = 2$ e os dígitos $n_i \in \{-1, 0, 1\}$. Esta representação é dita na *forma não adjacente* (FNA), se não houver dígitos não nulos vizinhos, sendo denotada por $(n_{\ell-1}...n_0)_{\text{FNA}}$. Esta representação é única e possui peso de Hamming mínimo dentre as possíveis representações com sinal de n. Em média, o número de dígitos não nulos na representação FNA é $\ell/3$.

O algoritmo 3.22 mostra como obter a representação de n na forma não adjacente.

Agora podemos combinar a idéia de representação com sinal e o método da janela [MO90]. Se considerarmos uma representação $n = (n_{\ell-1}...n_0)_2$, onde para quaisquer w

Algoritmo 3.22 Representação FNA

ENTRADA Um inteiro não negativo $n = (n_\ell ... n_0)_2$, tal que $n_\ell = n_{\ell-1} = 0$
SAÍDA A representação binária com sinal de n na forma não-adjacente $(n'_{\ell-1} ... n'_0)_{\text{FNA}}$.

$c_0 \leftarrow 0$
para $i = 0$ até $\ell - 1$ **faça**
  $c_{i+1} =\leftarrow \lfloor (c_i + n_i + n_{i+1})/2 \rfloor$
  $n'_i \leftarrow c_i + n_i - 2c_{i+1}$
retorne $(n'_{\ell-1} ... n'_0)_{\text{NAF}}$

dígitos seguidos existe apenas um dígito não nulo, e cada n_i é zero ou ímpar, tal que $|n_i| < 2^w$, então esta representação é denominada *forma não adjacente de janela de tamanho* w, e é denotada por FNA_w. O algoritmo 3.23 é uma generalização do algoritmo 3.22.

Algoritmo 3.23 Representação FNA_w

ENTRADA Um inteiro positivo n e um parâmetro $w > 1$.
SAÍDA A representação FNA_w, na forma $(n_{\ell-1} ... n_0)_{\text{FNA}_w}$ de n.

$i \leftarrow 0$
enquanto $n > 0$ **faça**
  se n é ímpar **então**
    $n_i \leftarrow n \pmod{2^w}$
    $n \leftarrow n - n_i$
  senão
    $n_i \leftarrow 0$
  $n \leftarrow n/2$
  $i \leftarrow i + 1$
retorne $(n_{\ell-1} ... n_0)_{\text{FNA}}$

Quando o expoente n é constante, é possível utilizar cadeias de adição calcular a exponenciação mais eficientemente. A obtenção de uma cadeia de adição pequena não é uma tarefa simples, mas tendo em vista a quantidade de vezes que o mesmo expoente será utilizado, é interessante computar a melhor cadeia de adições possível, para então utilizá-la para adquirir vantagem no cálculo da exponenciação.

Definição 3.2.7. Uma *cadeia de adição* para computar um inteiro n, é dada por duas sequências v e w tais que

$$v = (v_0, ..., v_s),\ v_0 = 1,\ v_s = n$$
$$v_i = v_j + v_k \text{ para todo } 1 \leq i \leq s, \text{ onde } j \text{ e } k \text{ são dados por}$$
$$w = (w_1, ..., w_s),\ w_i = (j, k) \text{ e } 0 \leq l, k \leq i - 1.$$

O *tamanho* desta cadeia de adições é s.

Com isso, uma cadeia de adições é uma sequência de inteiros tal que cada elemento é a soma de dois elementos anteriores e o último elemento da sequência é o valor de n desejado. Por exemplo, a cadeia $(1, 2, 3, 6, 12, 15)$ é uma cadeia de adições para $n = 15$.

Dado um valor de n, definimos $\ell(n)$ como sendo o menor valor de s tal que exista uma cadeia de adições de tamanho s. Determinar $\ell(n)$ é um problema difícil, mesmo para valores pequenos de n.

Como o cálculo de quadrados é mais eficiente que a multiplicação, desejamos cadeias de adições que permitiam minimizar a quantidade de multiplicações. Por exemplo, as cadeias $(1, 2, 4, 5, 6, 11)$ e $(1, 2, 3, 4, 8, 11)$ têm o mesmo tamanho, mas última permitir realizar a exponenciação utilizando uma multiplicação a menos.

Definição 3.2.8. Uma *cadeia de adição e subtração* é definida da mesma forma que uma cadeia de adição exceto que além de ter a condição $v_i = v_j + v_k$, temos também a possibilidade de ter $v_i = v_j - v_k$.

Exemplo 3.2.9. Podemos obter a cadeia de adição $(1, 2, 4, 8, 9, 19, 38, 39, 78, 156, 157, 314)$ para representar $n = 314$, mas usando uma cadeia de adição e subtração, é possível obter $(1, 2, 4, 5, 10, 20, 40, 39, 78, 156, 157, 314)$, que possui um termo a menos.

Um bom método para encontrar cadeias de adição e cadeias de adição e subtração pode ser encontrado em [KY98].

Uma vez tendo em mão uma cadeia de adição para representar o expoente n, podemos usar o algoritmo 3.24 para obter x^n.

Algoritmo 3.24 Exponenciação usando cadeia de adição

ENTRADA Um elemento x e uma cadeia de adição que represente n, como na definição 3.2.7.
SAÍDA O elemento x^n.

```
x_1 ← x
para i = 1 até s faça
    x_i ← x_j × x_k
retorne x_s
```

Se, por outro lado, mantirvermos fixa a base da exponenciação, isto é, se x for ser usado para calcular valores de x^n para diversos valores de n, então uma abordagem que traz ganho de eficiência é pré-calcular algumas potências de x.

O algoritmo 3.25, descrito em [Yao76], calcula a exponenciação baseado em potências b_i de x escolhidas livremente. Baseado nesta escolha, é preciso representar o expoente n da seguinte forma

$$n = \sum_{i=0}^{\ell-1} n_i b_i, \text{onde } 0 \leq n_i < h. \tag{3.2}$$

O valor de h também pode ser escolhido livremente. De fato, a escolha deste parâmetros vai depender da aplicação e do valor de x.

Algoritmo 3.25 Exponenciação usando o método de Yao

ENTRADA Um elemento x, um expoente n como descrito na equação 3.2 e os valores pré-calculados $x^{b_0}, ..., x^{b_{\ell-1}}$.
SAÍDA O elemento x^n.

```
y ← 1
u ← 1
j ← h − 1
enquanto j ≥ 1 faça
  para i = 0 até ℓ − 1 faça
    se n_i = j então
      u ← u × x^{b_i}
  y ← y × u
  j ← j − 1
retorne y
```

3.3 Aritmética de Corpos Binários

Nesta seção serão apresentados os algoritmos para realizar aritmética eficiente em corpos binários. Primeiramente, é preciso estabelecer uma forma de representar um elemento $x \in \mathbb{F}_{2^d}$. Existem duas representações que são frequentemente utilizadas:

- a representação polinomial, onde um elemento $x \in \mathbb{F}_{2^d}$ é representado por um polinômio de grau menor que d, $p(x) = a_0 + ... + a_{d-1}x^{d-1}$, com coeficientes $a_i \in \{0, 1\}$. A aritmética nesta representação é feita módulo um polinômio irredutível $f(x)$, de grau d.

- a representação em base normal, onde $(\beta, \beta^2, ..., \beta^{2^{d-1}})$, linearmente independentes, forma uma base para representação de elementos em $\mathbb{F}_{2^d}$. Um elemento deste corpo binário é representado por uma combinação linear dos elementos da base normal, $a_1\beta + ... + a_{d-1}\beta^{2^{d-1}}$, com coeficientes $a_i \in \{0, 1\}$.

Em ambos os casos, um elemento do corpo binário é determinado por um conjunto de d coeficientes $a_i \in \{0, 1\}$. Portanto, sua representação em um computador é dada por uma sequencia de bits, que serão agrupados em palavras tipicamente de 32 bits. Desta forma, um elemento de $\mathbb{F}_{2^{163}}$ por exemplo, pode ser representado por 163 bits, ou seja, 6 palavras de 32 bits, sendo que somente os 3 bits menos significativos da última palavra serão aproveitados.

Os cálculos em $\mathbb{F}_{2^d}$, em bases polinomiais, são realizados usando um polinômio irredútivel de grau d para reduzir os resultados a polinômios de grau menor que d. Assim, é preciso encontrar um polinômio que permita reduções modulares eficientes. Sabe-se que para qualquer valor de d, existe um polinômio irredutível de grau d, mas na prática deseja-se um trinômio ou um pentanômio, já que binômios e quadrinômios são sempre divisíveis por $x + 1$ em corpos binários.

Teorema 3.3.1. O trinômio $x^d + x^k + 1$, onde ou d ou k é ímpar, tem um número par de fatores se e somente se uma das condições abaixo for verdadeira:

- d é par, k é ímpar, $d \neq 2k$ e $dk/2 \equiv 0 \text{or } 1 \pmod 4$,
- d é ímpar, $d \equiv \pm 3 \pmod 8$, k é par e k não divide $2d$,
- d é ímpar, $d \equiv \pm 1 \pmod 8$, k é par e k divide $2d$.

Com isso, podemos deduzir que polinômios irredutíveis não existem quando $d \equiv 0 \pmod 8$ e são raros quando $d \equiv \pm 3 \pmod 8$.

3.3.1 Divisão

O algoritmo 3.26 realiza divisão de polinômio em corpos binários usando trinômios ou pentanômios. O número de somas necessárias para calcular o quociente e resto da divisão é $2b_4(d' - d + 1)$, onde d' é o grau de $f(x)$.

3.3.2 Bases Normais

Outra possibilidade é utilizar bases normais, especialmente em corpos binários, porque o quadrado de um elemento pode ser calculado através de um deslocamento circular das coordenadas do elemento. Contudo, a multiplicação é mais complicada. Na prática são usadas preferencialmente *bases normais ótimas* (BNO) e *bases normais gaussianas* (BNG).

Definição 3.3.2. Bases normais ótimas são definidas a partir de um conjunto de d elementos linearmente independentes, de forma que elementos de $\mathbb{F}_{2^d}$ possam ser representados por vetores de d elementos, cuja base é dada por $(\beta, \beta^2, ..., \beta^{2^{d-1}})$, $\beta \in \mathbb{F}_{2^d}$.

Algoritmo 3.26 Divisão de polinômios em $\mathbb{F}_{2^d}$

ENTRADA Polinômios $m(X), f(X) \in \mathbb{F}_2[X]$, onde $m(X) = X^d + a_4X^{b_4} + ... + a_0$ é um trinômio ou um pentanômio.
SAÍDA Os polinômios u e v, tais que $f = um + v$.

1: $u \leftarrow f$
2: $u \leftarrow 0$
3: **enquanto** grau $(v) \geq v$ **faça**
4: $\quad k \leftarrow \max d, \text{grau}\,(v) - d + b_4 + 1)$
5: $\quad v(X) \leftarrow u_1(X)X^k + w(X)$
6: $\quad v(X) \leftarrow w(X) - u_1(X)(m(X) - X^d)X^{k-d}$
7: $\quad u(X) \leftarrow u_1(X)X^{k-d} + u(X)$
8: **retorne** w

Em corpos binários existem dois tipo de BNOs:

- se $d+1$ é primo e 2 é um elemento primitivo de $\mathbb{F}_{d+1}$, então as raízes $(d+1)$-ésimas da unidade não triviais formam uma base normal ótima para $\mathbb{F}_{2^d}$, denominada *BNO do tipo 1*;
- se $2d+1$ é primo e
  - 2 é raíz primitiva de $\mathbb{F}_{2d+1}$ ou
  - 2 gera os resíduos quadráticos em $\mathbb{F}_{2d+1}$, isto é, $2d+1 \equiv 3 \pmod 4$ e a ordem de 2 em $\mathbb{F}_{2d+1}$ é d.

  Então há uma raíz $(2d+1)$-ésima da uidade ζ em $\mathbb{F}_{2^d}$ e $\zeta + \zeta^{-1}$ é um elemento normal que gera uma *BNO do tipo 2*.

3.3.3 Multiplicação

Agora podemos investigar algoritmos para realizar a multiplicação de elementos de um corpo binário, usando a representação polinomial ou então a representação em bases normais ótimas.

Primeiramente, para o caso de base polinomial, podemos derivar um método similar ao algoritmo 3.3, como mostra o algoritmo 3.27. Apesar da idéia do algoritmo ser a mesma, temos uma diferença crucial, pois para multiplicarmos dois inteiros de precisão arbitrária podemos usar a multiplicação de dígitos, implementada em hardware; enquanto que normalmente não existe uma instrução que seja capaz de multiplicar polinômios em corpos binários, sendo necessário utilizar explicitamente instruções de deslocamento e de manipulação lógica dos bits, como por exemplo o *ou exclusivo*.

Algoritmo 3.27 Multiplicação de polinômios em $\mathbb{F}_2[X]$

ENTRADA Polinômios $u(X), v(X) \in \mathbb{F}_2[X]$ de grau no máximo $d-1$ representados em palabvras de ℓ bits.
SAÍDA $w[X] = u(X).v(X)$ de grau no máximo $2d-2$.

```
1: w(X) ← 0
2: r ← ⌈grau (u/ℓ)⌉
3: para j = 0 até ℓ − 1 faça
4:    para i = 0 até r − 1 faça
5:       se u_i[j] = 1 então
6:          w(X) ← w(X) + v(X)X^{iℓ}
7: se j ≠ ℓ − 1 então
8:    v(X) ← v(X)X
9: retorne w
```

O algoritmo 3.28 considera uma janela de tamanho $k = 4$, para realizar a multiplicação de elementos de um corpo binário com auxílio de uma tabela. O tamanho desta janela pode variar, dependendo dos recursos computacionais disponíveis. Em uma arquitetura com palavra de 32 bits, seria desejável que houvesse uma tabela que permitisse realizar a multiplicação de polinômios de grau menor que 32 através de uma única consulta a uma tabela, mas na prática esta tabela seria muito grande, então é preciso encontrar um tamanho de janela que permita construir a tabela na memória disponível.

Algoritmo 3.28 Multiplicação de polinômios em $\mathbb{F}_2[X]$

ENTRADA Polinômios $u(X), v(X) \in \mathbb{F}_2[X]$ de grau no máximo $d-1$ representados em palabvras de ℓ bits. Produtos pré-calculados $t(X)v(X)$ para todo $t(X)$ de grau menor que k.
SAÍDA $w(X) \leftarrow u(X)v(X)$ de grau no máximo $2d-2$.

```
w(X) ← 0
r ← ⌈grau (u/ℓ)⌉
para j = ℓ/k − 1 até 0 faça
   para i = 0 até r − 1 faça
      t(X) ← t_{k−1}X^{k−1} + ... + t_0 onde t_m = u_i[jk + m]
      w(X) ← w(X) + t(X)v(X)X^{iℓ}        [t(X)v(X)] está pré-calculado
   se j ≠ 0 então
      w(X) ← w(X)X^k
retorne w
```

Usando bases normais, é preciso utilizar uma *matriz de multiplicação* $T_{\mathcal{N}}$, cujos elementos $t_{i,h}$ satisfaçam:

$$\beta^{q^i} \times \beta = \sum_{h=0}^{d-1} t_{i,h}\beta^{q^h} \text{e } \beta^{q^i}\beta^{q^j} = \sum_{h=0}^{d-1} t_{i-j,h-j}\beta^{q^h}. \tag{3.3}$$

Com isso, seja $u = (u_{d-1}...u_0)$ e $v = (v_{d-1}...u_0)$, a forma de w_h de $w = uv$ é dada por

$$w_h = \sum_{0 \le i,j < d} u_i v_j t_{i-j,h-j}. \tag{3.4}$$

Exemplo 3.3.3. Seja $m(X) = X^7 + X^6 + 1$. Dada uma raíz β de $m(X)$, obtemos a seguinte base normal:

$\beta = X$
$\beta^2 = X^2$
$\beta^{2^2} = X^4$
$\beta^{2^3} = X^6 + X + 1$
$\beta^{2^4} = X^6 + X^5 + X^4 + X^3 + X$
$\beta^{2^5} = X^5 + X^4 + X^2 + X + 1$
$\beta^{2^6} = X^4 + X^3 + 1$
$\beta^{2^7} = \beta.$

Desta maneira, os produtos $\beta^{q^i} \times \beta$ são dados por

$\beta \times \beta = X^2$
$\beta^2 \times \beta = X^3$
$\beta^{2^2} \times \beta = X^5$
$\beta^{2^3} \times \beta = X^6 + X^2 + X + 1$
$\beta^{2^4} \times \beta = X^5 + X^4 + X^2 + 1$
$\beta^{2^5} \times \beta = X^5 + X^4 + X^2 + X + 1$
$\beta^{2^6} \times \beta = X^5 + X^4 + X$

Assim, para obter $T_{\mathcal{N}}$, primeiramente calculamos:

$$\mathcal{M} = \left[\begin{array}{ccccccc|ccccccc} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{array}\right]$$

A primeira metade das colunas da matriz $\mathcal{M}$ representa as expressões para β^{q^i} e $\beta^{q^i} \times \beta$. Se realizarmos transformações nesta matriz de modo a obter a identidade na parte esquerda, então teremos a matriz $T_{\mathcal{N}}$ transposta:

$$T_{\mathcal{N}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

O número de zeros de $T_{\mathcal{N}}$ é denotado por $\sigma_{\mathcal{N}}$ e é denominado *densidade de* $T_{\mathcal{N}}$. Este é um parâmetro crucial para a velocidade da multiplicação de elementos de corpos de extensão usando bases normais ótimas. De fato, dados dois elementos pertencentes a $\mathbb{F}_{q^d}$, a multiplicação destes elementos pode ser efetuada em no máximo $2d\sigma_{\mathcal{N}}$ multiplicações e $d(\sigma_{\mathcal{N}} - 1)$ adições em $\mathbb{F}_q$. Na média, a densidade de $T_{\mathcal{N}}$ é $(q-1)d^2/q$ [BGM91], mas sabe-se que $\sigma_{\mathcal{N}} \geq 2d - 1$ [MOVW89]. Em bases normais ótimas a densidade é mínima, $\sigma_{\mathcal{N}} = 2d - 1$.

Podemos generalizar o conceito de bases normais ótimas para *bases normais de baixa complexidade*, conhecidas como *bases normais Gaussianas* (BNG).

Definição 3.3.4. Seja $p = mT + 1$ um número primo. Seja $K =< u >$, onde $u \in \mathbb{Z}_p^*$ tem ordem T. Suponha que e seja a ordem de 2 em $\mathbb{Z}_p^*$ e $(e, m) = 1$. Então, $\mathbb{Z}_p^* = \{2^i u^j | 0 \leq j < T\}$ e $K_i = K2^i$ para $0 \leq i < m$ são as classes laterais de K em $\mathbb{Z}_p^*$. Como $p|2^{mT} - 1$, temos uma raíz primitiva da unidade $\alpha \in \mathbb{F}_2^{mT}$. Os *períodos de Gauss* do tipo (m, T) são os valores $\beta_i = \sum_{j \in K_i} \alpha^j$ para $0 \leq i < m$. Seja $\beta = \beta_0$. Então $\beta_i \in \mathbb{F}_2^m$, $\beta_i = \beta^{2^i}$ e $(\beta, \beta^2, ..., \beta^{2^{m-1}})$ é uma base normal Gaussiana do tipo T.

De fato, BNGs com período $(m, 1)$ e $(m, 2)$ são BNOs do tipo 1 e 2 respectivamente. Para corpos binários onde não há uma base normal ótima, é possível utilizar uma BNG como alternativa. A tabela 3.5 mostra os valores de T recomendados pelo NIST para curvas elípticas sobre corpos binários.

Tabela 3.5: BNGs recomendadas pelo NIST para curvas elípticas sobre $\mathbb{F}_{2^m}$

m	163	233	283	409	571
T	4	2	6	4	10

3.3.4 Quadrado

O cálculo de x^2 usando bases polinomiais pode ser realizado apenas inserindo zeros entre os bits da representação de x, reduzindo o resultado final módulo $f(X)$. Esta operação pode ser eficientemente realizada em hardware. Em software, é possível usar uma tabela para a inserção de zeros. Assim, em ambos os casos a maior parte do consumo de tempo é dada pela redução modular final.

Já em bases normais, o quadrado pode ser trivialmente obtido pela instrução de deslocamento circular a esquerda. Da mesma forma, a raíz quadrada pode ser obtida pela instrução de deslocamento circular a direita. Com isso, a aritmética de algumas classes de isomorfismo em curvas elípticas supersingulares sobre corpos binários pode ser eficientemente implementada sem a utilização de coordenadas projetivas, pois, dado $P \in E$, o cálculo de $[2]P$ pode ser efetuado eficientemente devido ao mapa de Frobenius. Assim, o inverso de um elemento $x \in \mathbb{F}_{2^m}$ pode ser computado eficientemente. Além disso, curvas elípticas sobre corpos binários possuem um peso de Hamming pequeno, de modo que a quantidade de inversos que precisam ser calculados também é pequena.

3.3.5 Inversão

Para calcular o inverso de um elemento $f \in \mathbb{F}_q[X]$ em bases polinomiais é possível usar o algoritmo estendido de Euclides para cálculo do máximo divisor comum. Dados os polinômios f e $m \in \mathbb{F}_q[X]$, onde $m(X)$ é o polinômio irredutível que define o corpo binário em questão, existem polinômios únicos u, v e g, tais que $fu + mv = g$, onde $g = \text{MDC}(f, m)$, $\text{grau}\,(u) < \text{grau}\,(m)$ e $\text{grau}\,(v) < \text{grau}\,(v) < \text{grau}\,(f)$. Como $m(X)$ é irredutível, temos que $\text{MDC}(f, m) = 1$ e portanto $g = 1$ e u é o inverso de f módulo m. O algoritmo 3.29 é uma adaptação do algoritmo 3.11.

Algoritmo 3.29 Aloritmo Estendido de Euclides para MDC em corpos binários

ENTRADA Dois polinômios f e $m \in \mathbb{F}_2[X]$ de grau no máximo d.
SAÍDA Os polinômios u, v e g, tais que $fu + mv = g$ e $g = \text{MDC}(f, m)$.
  $u \leftarrow 1$, $v \leftarrow 0$ e $g \leftarrow f$
  enquanto $s \neq 0$ **faça**
    Calcule a divisão de g por s, tal que $g = qs + r$.
    $t \leftarrow u - vq$, $u \leftarrow v$, $g \leftarrow s$, $v \leftarrow t$ e $s \leftarrow r$
  $v \leftarrow (g - fu)/m$
  retorne (u, v, g)

O algoritmo 3.29 requer $O(d^2)$ operações elementares em $\mathbb{F}_q$. Para calcular o inverso em corpos binários é possível usar o algoritmo 3.30, inspirado no algoritmo 3.14.

Algoritmo 3.30 Inverso de um elemento em $\mathbb{F}_{2^d}^*$ em representação polinomial

ENTRADA Um polinômio irredutível $m(X) \in \mathbb{F}_2[X]$ de grau no máximo d e um polinômio não nulo $f(X) \in \mathbb{F}_2[X]$, tal que grau $(f) < d$.
SAÍDA $u(X) \in \mathbb{F}_2[X]$, tal que $f(X)u(X) \equiv 1 \pmod{m(X)}$.

```
u ← 1, v ← m e δ ← 0
para i = 0 até 2d faça
  se f_d = 0 então
    f(X) ← Xf(X), u(X) ← (Xu(X)) (mod m(X)) e δ ← δ + 1
  senão
    se s_d = 1 então
      s(X) ← s(X) − f(X) e v(X) ← (v(X) − u(X)) (mod m(X))
    s(X) ← Xs(X)
    se δ = 0 então
      t(X) ← f(X), f(X) ← s(X) e s(X) ← t(X)
      t(X) ← u(X), u(X) ← v(x) e v(X) ← t(X)
      u(X) ← (Xu(X)) (mod m(X))
      δ = 1
    senão
      u(X) ← (u(X)/X) (mod m(X)) e δ = δ − 1
retorne u
```

3.3.6 Exponenciação

Em representação polinomial é possível usar uma técnica que permite tornar a exponenciação significativamente mais eficiente. Seja $f(X)$ e $m(X)$ polinômios em $\mathbb{F}_q[X]$ e $g(X) = X^{q^r}$. Como o mapa de Frobenius é um automorfismo que mantém fixos os elementos do corpo base, temos que $f^{q^r} \equiv f(g) \pmod{m(X)}$. Com isso, é possível usar o algoritmo de Brent e Kung [BK78] para efetuar a exponencição.

A idéia é usar os polinômios

$$f(X) = \sum_{0 \le i < k} X^{ki} F_i(X) \text{onde } k = \lceil \text{grau } f \rceil \text{e} \sum_{0 \le j < k} f_{ik+j} X^j \tag{3.5}$$

e pré-calcular os valores $1, g, g^2, ..., g^{k-1} \pmod{m(X)}$. O algoritmo 3.31 implementa esta idéia.

Exemplo 3.3.5. Seja $m(X) = X^{15} + X + 1$ irredutível sobre $\mathbb{F}_2$, $f(X) = X^{14} + x^{13} + X^8 + X^6 + X^4 + X^3 + 1$ e $g(X) = X^{10} + X^3 + 1$. Com isso, temos que $k = 4$ e

$$f(X) = F_0(X) + X^4 F_1(X) + X^8 F_2(X) + X^{12} F_3(X)$$

Algoritmo 3.31 Composição Modular de Brent e Kung

ENTRADA Os polinômios $m, f, g \in \mathbb{F}_q[X]$ onde grau $m = d$ e grau $f, g < d$.
SAÍDA O polinômio $f(g) \pmod{m(X)}$.

$k \leftarrow \lceil \sqrt{d} \rceil$
$G[0] \leftarrow 1$
para $i = 1$ até k **faça**
  $G[i] \leftarrow (gG[i-1]) \pmod{m(X)}$
$P[0] \leftarrow 1$
para $i = 1$ até $k-1$ **faça**
  $P[i] \leftarrow (G[k]P[i-1]) \pmod{m(X)}$
para $i = 0$ até $k-1$ **faça**
  $F[i] \leftarrow \sum_{j=0}^{k-1} f_{ik+j} G[j]$
$R \leftarrow (\sum_{i=0}^{k-1} F[i]P[i]) \pmod{m(X)}$
retorne R

com

$$F_0(X) = X^3 + 1,\ F_1(X) = X^2 + 1,\ F_2(X) = 1 \text{ e } F_3(X) = X^2 + X.$$

Os valores pré-calculados de g^i e g^{ik} para $0 \leq i \leq k$ são respectivamente armazenados em vetores G e P onde $F[i]$ contém $F_i(g)$.

i	$G[i]$	$P[i]$	$F[i]$
0	(0001)	(0001)	(010101001011)
1	(010000001001)	(010000000001)	(00100000)
2	(00100001)	(01100001)	(0001)
3	(010100101010)	(010001100100)	(010001001000)

Finalmente obtemos $R = X^{13} + X^{12} + X^{11} + X^9 + X^7 + X^5 + X^3 + X^2 + X + 1$, que é equivalente a $f(g) \pmod{m(X)}$.

Agora podemos apresentar o algoritmo de Shoup [Sho94] que utiliza a base q^r para representar o expoente.

Algoritmo 3.32 Método de exponenciação de Shoup

ENTRADA Os polinômios $f, m \in \mathbb{F}_q[X]$, onde grau $m(X) = d$ e grau $f < d$. Um parâmetro r um expoente $n = (n_{\ell-1}...n_0)_{q^r}$ tal que $0 < n < q^d$.
SAÍDA O polinômio $f^n \pmod m$.
  para $i = 0$ até $\ell - 1$ **faça**
    Pré-calcule e armazene $f^{n_i} \pmod m$
  $g(X) \leftarrow X^{q^r} \pmod m$
  $y \leftarrow 1$
  para $i = \ell - 1$ até 0 **faça**
    $y \leftarrow y(g)$ Use o algoritmo 3.31
    $y \leftarrow (y \times f^{n_i}) \pmod m$
  retorne y

3.4 Aritmética de Curvas Elípticas

Nesta seção vamos apresentar os principais algoritmos para realizar aritmética de curvas elípticas. Em criptografia, a principal operação envolvida nos protocolos é a multiplicação por escalar, em que um ponto P é somado a ele próprio n vezes, onde n é um número inteiro. O algoritmo para cálculo da multiplicação por escalar é análogo ao algoritmo 3.21.

3.4.1 Aritmética de curvas elípticas sobre corpos binários

Nesta seção apresentaremos algoritmos para realizar aritmética de curvas elípticas sobre corpos binários. Para isso, vamos considerar a seguinte equação de curva elíptica

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

Curvas supersingulares

Se $a_1 = 0$, é preciso ter $a_3 \neq 0$, pois em caso contrário teríamos uma curva singular. A trasformação $x \to x + a_2$, deriva a seguinte equação de curva:

$$E : y^2 + a_3y = x^3 + a'_4x + a'_6.$$

Esta curva não possui nenhum ponto $P = (x_1, y_1)$ de ordem dois sobre $\overline{\mathbb{F}}_{2^d}$, pois neste caso teriamos $P = -P$, de forma que $y_1 = y_1 + a_3$, o que seria verdade apenas se $a_3 = 0$. Logo, $E[2] = \{P_\infty\}$ e portanto de acordo com o teorema 2.6.12, E é supersingular. Curvas supersingulares são frequentemente utilizadas em criptografia baseada em emparelhamentos por permitir a utilização de corpos de extensão com grau de mergulho pequeno, de

Algoritmo 3.33 Multiplicação escalar em curvas elípticas (janela)

ENTRADA Um ponto P em uma curva E, um inteiro positivo $n = (n_{\ell-1}...n_0)_2$, um parâmetro $k \geq 1$ e valores pré-calculados de $[3]P, [5]P, ..., [(2^k - 1)]P$.
SAÍDA O ponto $[n]P$.

1. $Q \leftarrow P_\infty$
2. $i \leftarrow i - 1$
3. **enquanto** $i \geq 0$ **faça**
4. **se** $n_i = 0$ **então**
5. $Q \leftarrow [2]Q$
6. $i \leftarrow i - 1$
7. **senão**
8. $s \leftarrow \max(i - k + 1, 0)$
9. **enquanto** $n_s = 0$ **faça**
10. $s \leftarrow s + 1$
11. **para** $h = 1$ até $i - s + 1$ **faça**
12. $Q \leftarrow [2]Q$
13. $u \leftarrow (n_i...n_s)_2$
14. $Q \leftarrow Q + [u]P$
15. $i \leftarrow s - 1$
16. **retorne** Q.

modo que a aritmética nesta extensão possa ser realizada eficientemente. Devido aos ataques MOV [MOV93] e FR [FMR99], curvas supersingulares têm sido evitadas pela comunidade científica, mas até agora não existe uma razão concreta para acreditar que essas curvas sejam menos seguras que as curvas não supersingulares.

Considerando curvas supersingulares sobre $\mathbb{F}_{2^m}$, com m ímpar, temos três classes de isomorfismos, dados por

$$E_1 : y^2 + y = x^3$$

$$E_2 : y^2 + y = x^3 + x$$

$$E_3 : y^2 + y = x^3 + x + 1.$$

O grau de mergulho para essas curvas é respectivamente 2, 4 e 4. Com o uso de emparelhamentos bilineares é possível reduzir o problema do logaritmo discreto nestas curvas para o problema do logaritmo discreto em $\mathbb{F}_{2^{km}}$. Portanto, as curvas E_2 e E_3 são mais indicadas para uso criptográfico.

A lei de adição e duplicação para as curvas elípticas E_2 e E_3, para $P + Q = (x_3, y_3)$ é dada por

$$x_3 = \begin{cases} \left(\dfrac{y_1+y_2}{x_1+x_2}\right)^2 + x_1 + x_2 & \text{se } P \neq Q, \\ x_1^4 + 1 & \text{se } P = Q \end{cases}$$

e

$$y_3 = \begin{cases} \left(\dfrac{y_1+y_2}{x_1+x_2}\right)(x_1+x_3) + y_1 + 1 & \text{se } P \neq Q, \\ x_1^4 + y_1^4 + 1 & \text{se } P = Q \end{cases}$$

Escolha das coordenadas

Usando bases normais, multiplicar um ponto por 2 pode ser realizado de forma muito eficiente, enquanto a soma de dois pontos distintos pode ser realizada com duas multiplicações e uma inversão. Deste modo, computar kP pelo método do quadrado e da multiplicação exige $2t$ multiplicações e t inversões, onde $t+1$ é o peso de Hamming de k.

O cálculo de inversos pode ser eliminado com a utilização de coordenadas projetivas. A seguir são apresentadas as fórmulas para a adição de pontos de uma curva elíptica E do tipo E_2 ou E_3. Seja $P = (x_1, y_1, 1) \in E$, $Q = (x_2, y_2, z_2) \in E$ e suponha que $P, Q \neq \infty$, $P \neq Q$ e $P \neq -Q$. Seja $P + Q = (x_3, y_3, z_3)$. Então,

$$x_3 = A^2 B z_2 + B^4$$

$$y_3 = (1 + y_1) z_3 + A^3 z_2 + A B^2 x_2$$

$$z_3 = B^3 z_2,$$

onde $A = (y_1 z_2 + y_2)$ e $B = (x_1 z_2 + x_2)$. Esta fórmula requer 9 multiplicações de elementos do corpo subjacente.

Com isso, é possível computar kP, onde P é um ponto em coordenadas afins $(x_1, y_1, 1)$, usando o método de quadrado e multiplicação. O resultado $kP = (x_k, y_k, z_k)$ pode ser convertido novamente em coordenadas afins multiplicando cada coordenada por z_k^{-1}. O número total de operações é $9t + 2$, onde $t + 1$ é o peso de Hamming de k. Logo, no cenário de curvas elípticas supersingulares sobre corpos binários, coordenadas projetivas não acresce ganho de eficiência aos algoritmos.

3.4.2 Multiplicação por escalar

Nesta seção serão descritos algoritmos para computar nP, dado um inteiro n e um ponto P da curva elíptica E. O algoritmo 3.34 é genérico e pode ser usado em qualquer tipo de curva. Como podemos notar, em cada passo são efetuadas exatamente uma soma e uma

duplicação. Com isso, é possível evitar ataques que tentam medir o consumo de tempo ou energia para cada passo, tentando deduzir os bits de n. A complexidade deste algoritmo, se forem utilizadas coordenadas afins para representação de P, é $(6M + 4S)(|n^2| - 1)$.

Algoritmo 3.34 Multiplicação escalar pelo método Montgomery

ENTRADA Um ponto $P \in E$ e um inteiro positivo $n = (n_{\ell-1}...n_0)_2$.
SAÍDA $[n]P$.

```
P1 ← P
P2 ← [2]P
para i = ℓ − 2 até 0 faça
    se n_i = 0 então
        P1 ← [2]P1
        P2 ← P1 + P2
    senão
        P1 ← P1 + P2
        P2 ← [2]P2
retorne P1
```

Em curvas supersingulares sobre corpos binários não existem pontos de 2-torsão. Assim, o mapa $P \to [2]P$ é injetor, de modo que qualquer ponto da curva pode dividido ao meio, isto é, dado um ponto $P \in E$, existe $Q \in E$ tal que $P = [2]Q$. O algoritmo 3.35 permite dividir um ponto P ao meio, dado que a curva elíptica subjacente seja tal que $|E| = 2.\ell$, para ℓ ímpar. Neste caso, temos um subgrupo $G \subset E$, tal que $|E| = 2.|G|$, onde todo ponto $P \in G$ pode ser dividido ao meio. Desta maneira, é possível inverter as fórmulas de duplicação de ponto, obtendo

$$\lambda_2^2 + \lambda = a_2 + x_1,$$

$$x_2^2 = x_1(\lambda_2 + 1) + y_1 = x_1(\lambda_2 + \lambda_1 + x_1 + 1),$$

$$y_2 = x_2(x_2 + \lambda_2).$$

Mas (x_2, λ_2) é igual a $[1/2]P$ se e somente se a equação $X^2+X = a_2+x_2$ possuir solução em $\mathbb{F}_{2^d}$. Isto é verdade se e somente se $\mathrm{Tr}(a_2^2+x^2) = 0$. Mas $\mathrm{Tr}(a_2+x_2) = \mathrm{Tr}(a_2^2+x_2^2)$, de modo que é possível evitar o cálculo de uma raíz quadrada. Então, é possível computar $w = x_1(\gamma + \lambda_1 + x_1 + 1)$, onde γ é solução de $\gamma^2 + \gamma = a_2 + x_1$. De fato, w é um candidato para x_2^2. Se $\mathrm{Tr}(a_2^2 + w) = 0$, então $\lambda_2 = \gamma$ e $x_2 = \sqrt{w}$. Caso contrário, $\lambda_2 = \gamma + 1$ e $x_2 = \sqrt{w + x_1}$.

Agora podemos explicar como é possível computar $[n]P$. Dada uma ordem ímpar $\ell_1|\ell$ e $m = \lceil \lg \ell_1 \rceil$. Se

Algoritmo 3.35 Divisão ao meio de pontos

ENTRADA Um ponto $P = (x_1, y_1) \in E$, representado como (x_1, λ_1).
SAÍDA $[1/2]P = (x_2, y_2)$, representado como (x_2, λ_2).
  Calcule γ tal que $\gamma^2 + \gamma = a_2 + x_1$
  $w \leftarrow x_1(\gamma + \lambda_1 + x_1 = 1)$
  se $\mathrm{Tr}\,(a_2^2 + w) = 1$ **então**
    $\gamma = \gamma + 1$
    $w = w + x_1$
  $\lambda_2 \leftarrow \gamma$
  $x_2 \leftarrow \sqrt{w}$
  retorne $Q = (x_2, \lambda_2)$

$$2^{m-1}n = \sum_{i=0}^{m-1} \hat{n_i} 2^i \pmod{\ell)_1}, \text{com } n_i \in \{0, 1\}$$

obtemos

$$n \equiv \sum_{i=0}^{m-1} \frac{\hat{n}_{m-1-i}}{2^i} \pmod{\ell_1}$$

e $[n]P$ pode ser computado pelo algoritmo 3.36.

As otimizações estudadas para exponenciação de elementos de um corpo finito também podem ser aplicadas neste caso. Com isso, é possível derivar algoritmos que utilizam janelas ou recodificações de n para melhorar a eficiência deste.

Algoritmo 3.36 Multiplicação por divisão ao meio e soma

ENTRADA Um ponto $P \in E(\mathbb{F}_{2^d})$ de ordem ℓ_1 e um inteiro positivo n.
SAÍDA $[n]P$.
  $m \leftarrow \lceil \lg \ell_1 \rceil$
  $\hat{n} \leftarrow (2^{m-1}n) \pmod{\ell_1}$
  $Q \leftarrow P_\infty$
  para $i = 0$ até $m - 1$ **faça**
    $Q \leftarrow [1/2]Q$
    se $\hat{n}_i = 1$ **então**
      $Q \leftarrow Q + P$
  retorne Q

3.5 Cálculo de Emparelhamentos

Seja E uma curva elíptica definida sobre um corpo finito $\mathbb{F}_q$. O grupo da classe de divisores de grau zero desta curva elíptica é denotado por $\mathrm{Pic}_0(E)$. Seja r um inteiro tal que r divide a ordem de $\mathrm{Pic}_0(E)$. Como vimos na seção 2.6.6, em curvas de genus $g = 1$, um ponto $P \in E$ pode ser mapeado para uma classe de divisores de grau zero, cujo representante é $[P] - [\infty]$, para algum ponto P da curva. Assim, denotamos por $\mathrm{Pic}_0(E)[r]$ a classe de divisores que representam pontos de n-torção, ou seja, tais que sua ordem divide r.

Seja D_1 um divisor que representa uma classe de $\mathrm{Pic}_0(E)[r]$ e D_2 um divisor cujo suporte seja disjunto do suporte de D_1. Como rD_1 é um divisor principal, existe uma função racional f_{D_1}, definida sobre E, tal que $(f_{D_1}) = rD_1$. O emparelhamento de Tate é definido com sendo um mapa da seguinte forma:

$$<,>: \mathrm{Pic}_0(E)[r] \times \mathrm{Pic}_0(E)/r\,\mathrm{Pic}_0(E) \rightarrow \mathbb{F}_{q^k}/(\mathbb{F}_{q^k})^r.$$

O inteiro k é denominado *grau de mergulho* e representa o menor inteiro tal que $r|q^k-1$, de forma que $\mathbb{F}_{q^k}$ contém as raízes r-ésimas da unidade.

Dados os divisores D_1 e D_2, definidos como anteriormente, o emparelhamento de Tate é realizado computando o valor da função f_{D_1} no divisor D_2. Para obter uma raíz r-ésima da unindade é preciso realizar uma exponenciação final, que mapeia um elemento qualquer de uma classe lateral de $\mathbb{F}_{q^k}/(\mathbb{F}_{q^k})^r$ para um raíz r-ésima da unidade.

Desta forma, o emparelhamento de Tate-Lichtenbaum é definido como sendo

$$e(D_1, D_2) = f_{D_1}(D_2)^{q^k-1/r}.$$

3.5.1 Algoritmo de Miller

O cálculo do emparelhamento de Tate pode ser efetuado em tempo polinomial através do algoritmo de Miller. Como estamos interessados apenas em curvas elípticas, ou seja, curvas de genus 1, então todos os divisores de grau zero $D \in \mathrm{Pic}_0(E)$ podem ser representados por um divisor D_P equivalente a D módulo um divisor de uma função racional definida sobre a curva, de forma que $D_P = [P] - [\infty]$. Tais divisores são denominados *divisores reduzidos*. Com isso, temos um mapa bijetivo entre pontos da curva e divisores reduzidos.

Seja $P \in E(\mathbb{F}_q)[r]$ e $Q \in E(\mathbb{F}_{q^k})$ tal que P e Q sejam linearmente independentes, ou seja, Q não pertence a mesma classe lateral definida por P no grupo quociente $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$. De fato, Q pode ser obtido através de um mapa de distorção, que mapeia um múltiplo de P em um elemento linearmente indepente de P, como desejamos. Como precisamos de divisores D_1 e D_2 de suporte disjunto, Q precisa ser representado por um divisor equivalente a $[Q] - [\infty]$, mas que não possua o símbolo $[\infty]$. Dado um

ponto aleatório $R \in E(\mathbb{F}_{q^k})$, o divisor $[Q+R]-[R]$ satisfaz as condições dadas. Portanto, desejamos computar

$$e([P]-[\infty],[Q+R]-[R]).$$

Para todo inteiro n, existe uma função racional $f_{n,P}$, tal que $(f_{n,P}) = n[P]-[nP]-[(n-1)\infty]$. O algoritmo de Miller computa a função $f_{n,P}$ usando linhas ℓ e v, resultantes da lei de adição entre os pontos nP e mP, de forma que

$$f_{n+m,P} = f_{n,P} f_{m,P} \ell / v.$$

O valor do emparelhamento $e([P]-[\infty],[Q+R]-[R])$ é dado por $f_{r,P}([Q+R]-[R])^{q^k-1/r}$.

3.5.2 Otimizações para o algoritmo de Miller

Como estamos interessados em emparelhamentos sobre curvas supersingulares, podemos citar uma série de idéias que permitem computar o emparelhamento de Tate mais eficientemente:

1. o ponto R pode ser escolhido de forma que $\ell(R)^{q^k-1/r}$ e $v(R)^{q^k-1/r}$ sejam iguais a 1. Com isso é possível ignorar o cálculo destas funções no decorrer do algoritmo de Miller, já que após a exponenciação final estes termos naturalmente desaparecem;

2. o mapa de distorção pode ser escolhido de forma que $v(Q)$ possua coordenada x em um subcorpo, de modo que $v(Q)^{q^k-1/r} = 1$. Desta maneira é possível evitar o cálculo de inversas no algoritmo de Miller;

3. o peso de Hamming de r determina o número de vezes que o algoritmo de Miller permanece em laço. Assim, a escolha de curvas elípticas onde r tenha um peso de Hamming pequeno permite computar o emparelhamento de Tate eficientemente.

Originalmente, o algoritmo de Miller foi projetado para calcular o emparelhamento de Weil [Mil86, Mil04b]. O algoritmo 3.37 pode ser usado para calcular o emparelhamento de Tate-Lichtenbaum.

Exemplo 3.5.1. Considere a curva elíptica $E : y^2 = x^3 + 6$, definida sobre $\mathbb{F}_{13}$. A ordem da curva é 7 e $E(\mathbb{F}_{13})$ é gerada pelo ponto $P = (2,1)$. Na cadeia de adição, obtemos $T = P$, $T = [2]P$, $T = [3]P$ e $T = [6]P$. O grau de mergulho é $k = 2$, já que 7 divide $13^2 - 1$ e não divide $13 - 1$. Como 2 não é um quadrado em $\mathbb{F}_{13}$, então existe α tal que $\mathbb{F}_{13^2} \equiv \mathbb{F}_{13}[\alpha]$, onde $\alpha^2 = 2$. Agora é possível calcular o emparelhamento de Tate-Lichtenbaum para P e $Q = (10+3\alpha, 11+2\alpha)$.

Algoritmo 3.37 Emparelhamento de Tate-Lichtenbaum

ENTRADA Um inteiro $r = (r_{\ell-1}...r_0)_2$. Um ponto $P = (x_1, y_1) \in E(\mathbb{F}_q)[r]$ e um ponto $Q = (x_2, y_2) \in E(\mathbb{F}_{q^k})$.
SAÍDA $e(P, Q)$.
$T \leftarrow P$, $f_1 \leftarrow 1$, $f_2 \leftarrow 1$
para $i = \ell - 2$ até 0 **faça**
$T \leftarrow [2]T$
$\lambda \leftarrow$ a inclinação da tangente de E em T
$f_1 \leftarrow f_1^2(y_2 - \lambda(x_2 - x_3) - y_3)$
$f_2 \leftarrow f_2^2(x_2 + (x_1 + x_3) - \lambda^2)$
se $\ell_i = 1$ **então**
$T \leftarrow T + P$
$\lambda \leftarrow$ a inclinação da linha que passa por T e P
$f_1 \leftarrow f_1(y_2 - \lambda(x_2 - x_3) - y_3)$
$f_2 \leftarrow f_2(x_2 + (x_1 + x_3) - \lambda^2)$
retorne $(f_1/f_2)^{q^k - 1/\ell}$

Usando o algoritmo 3.37, obtemos $f_1 = f_2 = 1$ e $T = (2, 1)$.

Para $i = 1$, temos os seguintes valores computados:

$$\lambda = 3x_3^2/2y_3 = 6,$$

$$L_1 = y - y_3 - \lambda(x - x_3) = y + 7x + 11,$$

$$L_2 = x + 2x_3 - \lambda^2 = x + 7.$$

As linhas L_1 e L_2 são resultantes da duplicação $T = [2]P$.

Calculando o valor de $f_1 = L_1(Q)$ e $f_2 = L_2(Q)$, obtemos respectivamente os elementos $1 + 10\alpha$ e $4 + 3\alpha$ de $\mathbb{F}_{13}[\alpha]$.

Como $r_1 = 1$, é preciso computar as linhas resultantes da soma de $T = [2]P$ e P, obtendo $L_1 = y + 12$ e $L_2 = x + 8$, de modo que $f_1 = f_1L_1(Q) = 11 + 11\alpha$ e $f_2 = f_2L_2(Q) = 12 + \alpha$.

Agora, temos que $T = [3]P$ e $i = 0$. Com isso, calcula-se as linhas $L_1 = y + 5x + 2$ e $L_2 = x + 11$, resultantes da duplicação de T. Assim, $f_1 = f_1^2L_1(Q) = 1 + 6\alpha$ e $f_2 = f_2^2L_2(Q) = 12 + 6\alpha$.

Como $l_0 = 1$, é preciso calcular as linhas $L_1 = x - 2$ e $L_2 = 1$, resultantes da soma de $T = [6]P$ e P, de maneira que $f_1 = 5 + 12\alpha$ e $f_2 = 12 + 6\alpha$.

Logo, o emparelhamento de Tate-Lichtenbaum $e(P, Q)$, é dado por

$$\left(\frac{5 + 12\alpha}{12 + 6\alpha}\right)^2 4 = 4 + \alpha.$$

3.6 Logaritmo Discreto

Criptossistemas baseados em emparelhamentos utilizam problemas difíceis como base para determinar o grau de segurança de seus protocolos. Com isso, deseja-se que um atacante que queira obter algum tipo de vantagem precise ser capaz de resolver tais problemas. Nesta seção serão apresentados os problemas e também algoritmos para resolvê-los, analisando a complexidade destes algoritmos. Com isso, é possível determinar condições que permitem computar estes algoritmos de forma mais eficiente. Tais condições devem ser evitadas, pois representam uma alternativa melhor para um possível atacante. Além disso, são dadas informações importantes para a escolha de parâmetros da curva elíptica e corpo finito subjacente para a obtenção de grupos onde estes problemas permanecem difíceis.

Considerando um grupo elíptico E, podemos definir os seguintes problemas [Mao04]:

- *Problema do logaritmo discreto em curvas elípticas (PLDCE):* Dados $Q, P \in E$, tais que $Q = nP$, determinar n.
- *Problema computacional de Diffie-Hellman em curvas elípticas (PCDHCE:* Dados três pontos $P, aP, bP \in E$, determinar abP.
- *Problema de decisão de Diffie-Hellman em curvas elípticas (PDDHCE):* Dados quatro elementos P, aP, bP e cP pertencentes a E, responder verdadeiro se e somente se $c \equiv ab \pmod{|E|}$.

Uma das primeiras utilizações de emparelhamentos foi feita por Joux [Jou00]. Neste artigo ele mostra como o problema de decisão de Diffie-Hellman pode ficar fácil através de mapas bilineares, com isso conseguiu produzir uma aplicação para compartilhamento de chaves entre três partes em uma única rodada.

3.6.1 Algoritmos genéricos

Dado um grupo genérico, existem técnicas que permitem calcular o logaritmo discreto em tempo exponencial no tamanho da entrada, isto é, o número de bits necessários para representar a ordem deste grupo. O principal conceito envolvido nestes algoritmos é o *paradoxo do aniversário*. Que definimos da seguinte forma:

Definição 3.6.1. Seja S um conjunto de N elementos. Sorteando-se aproximadamente $1.2\sqrt{N}$ elementos de forma aleatória, permitindo repetição, existe uma probabilidade superior a 50% de que exista pelo menos um par de elementos iguais neste sorteio.

Prova. A probabilidade de obter pelo menos um par igual no sorteio de k elementos é de

$$p(N,k) = \big(1 - \frac{1}{N}\big)...\big(1 - \frac{k-1}{N}\big).$$

De fato, é fácil mostrar que, para N grande,

$$p(N,k) = e^{-\frac{k(k+1)}{2N}}.$$

Assim, para $p(N,k) = 0.5$, temos que

$$k = \frac{1}{2} + \sqrt{\frac{1}{4} + 2N \lg 2}. \ \square$$

Portanto, dada uma função de resumo criptográfico $H : \{0,1\}^\infty \to \{0,1\}^\ell$, onde n representa o grau de segurança desta função e deve ser grande o suficiente para que não seja possível um ataque de força bruta que percorra todo o espaço $\{0,1\}^\ell$; temos que a probabilidade de encontrar uma colisão é 50% após $\sqrt{2^\ell}$ execuções de H, isto significa que um atacante consegue encontrar a colisão utilizando um espaço equivalente a $\{0,1\}^{\ell/2}$, diminuindo o grau de segurança pela metade.

O logaritmo discreto em G pode ser computado facilmente se $n = |G|$ tem apenas fatores pequenos. De fato, resolver o PLD em um grupo onde n é composto, corresponde a resolver o mesmo problema em um grupo de ordem p^α, onde p^α é o maior fator primo de n. Isto foi primeiramente observado em [PH78]. Assim, dado $n = p_1...p_r$, o PDL em G pode ser resolvido facilmente nos subgrupos de ordem $p_1, ..., p_r$, de forma que o teorema do resto chinês permite calcular r valores para o logaritmo discreto em G e alguns testes permitem verificar qual o valor correto.

Um resultado geral de Shoup [Sho97] afirma que o limite inferior para o número de operações para resolver o PLD é $2\sqrt{n}$.

Método do passo pequeno e passo grande

Este método primeiramente proposto por Shanks em [Sha71]. A sua primeira aplicação foi para computar a ordem de um elemento, mas pode ser usado para computar o logaritmo discreto. O algoritmo 3.38 computa o PLD para grupos genéricos.

A figura 3.6.2 representa o funcionamento do algoritmo 3.38. O valor do logaritmo discreto t pode ser qualquer elemento no intervalo de 0 a $n-1$, mas o método utiliza uma função de resumo H permitindo que o valor de t seja encontrado analisando apenas alguns valores do intervalo. A idéia é utilizar H para armazenar elementos que podem ser associados a possibilidade de t pertencer ao intervalo de 0 a $s-2$, isto é, os valores de j no algoritmo 3.38. Com isso, é possível varrer os possíveis valores de t dando passos grandes, ou seja, pulando $s-1$ valores a cada passo. Assim, este algoritmo consegue de fato atingir o limite de Shoup.

Algoritmo 3.38 Passo pequeno e passo grande

ENTRADA Um gerador g de um grupo G de ordem n e $h \in G$.
SAÍDA Um inteiro t tal que $[t].g = h$.

$s \leftarrow \lfloor \sqrt{n} \rfloor + 1$
para $j = 0$ até $s - 2$ **faça**
    armazene $(\beta_j \leftarrow h + [-j]g, j)$ em uma tabela de resumo H [passo pequeno]
$i \leftarrow 0$
$\gamma \leftarrow 0_G$
enquanto verdade **faça**
    se $\gamma = \beta_j$ para algum j **então**
        retorne $i(s-1) + j$
    $i \leftarrow i + 1$
    $\gamma \leftarrow \gamma + [s-1]g$ [passo grande]

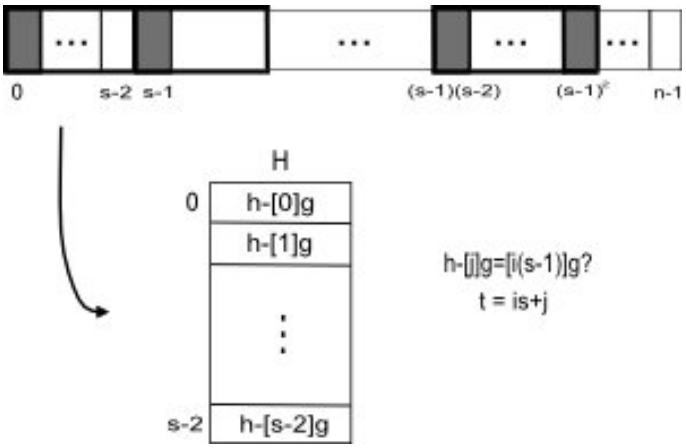


Figura 3.6.2: Diagrama do método do passo pequeno e passo grande

Este método basea-se no fato de que qualquer número inteiro entre 0 e $n-1$ pode ser expresso na forma $t = U + Vs$, para $0 \leq U, V < s$, garantindo que o algoritmo percorre todos os possíveis valores de t.

Com isso, podemos obter variações deste algoritmo utilizando maneiras diferentes de representar t. O algoritmo 3.39 dobra o tamanho do passo grande periodicamente. Ele pode ser usado para calcular a ordem de um elemento e precisa inicialmente de uma estimativa de t. Este método foi sugerido por Buchmann, Jacobson e Teske [BJT97] e tem complexidade $O(\sqrt{t})$.

Algoritmo 3.39 Passo pequeno e passo grande

ENTRADA Um gerador g de um grupo G de ordem desconhecida, uma estimatia $t \in G$ e o valor $h \in G$.
SAÍDA Um inteiro t tal que $[t].g = h$.

```
k ← ⌊√(lg t)⌋
para j = 0 até 2^{k+1} − 1 faça
    armazene (β_j ← h + [−j]g, j) em uma tabela de resumo H        [passo pequeno]
enquanto verdade faça
    para c = ⌊2^{k−1}⌋ até 2^{k+1} − 1 faça
        γ ← [2^{k+1}c]g                                            [passo grande]
        se γ = β_j para algum j, 1 ≤ j ≤ 2^{k+1} então
            retorne 2^{k+1}c + j
        k ← k + 1
    para j = 2^k até 2^{k+1} − 1 faça
        insira (β_j ← h + [−j]g, j) em H
```

Método rho de Pollard

Este método foi inicialmente proposto para resolver o PLD em corpos primos [Pol78] e é baseado no paradoxo do aniversário. Se elementos forem escolhidos aleatoriamente de um grupo $G =< g >$, de ordem n, então a quantidade de elementos que devem ser escolhidos até obter uma colisão é de aproximadamente $\sqrt{\pi n/2}$.

A idéia do método é gerar um caminho w_i, para $i \geq 0$, de forma que a seqüência de valores de w_i tenha comportamento aleatório, ou seja, tenha uma distribuição que seja o mais próximo possível de uma distribuição uniforme. É importante que esta seqüência seja determinística, permitindo a detecção da colisão sem ter que armazenar todos os elementos.

Quando a colisão acontece, está determinado um ciclo na seqüência w_i, de modo que existem inteiros k e c, tais que $w_k = w_{k+c}$. Neste caso, dizemos que a seqüência possui período c. A figura 3.6.3 mostra este ciclo, que é responsável pelo nome do método.

Existem diversos algoritmos para detecção de ciclos. De fato, estes algoritmos geralmente não dependem da estrutura do grupo G e dada a seqüência pseudo-aleatória w_i, é possível detectar ciclos para fins que vão além do uso no problema do logaritmo discreto e cálculo da ordem.

A seguir apresentaremos algoritmos de detecção de ciclos, considerando um mapa $\phi : G \to G$ que define a seqüência aleatória w_i, isto é, $\phi(w_i) = w_{i+1}$.

O algoritmo de Floyd [Knu97] não precisa comparar cada w_i com os elementos anteriores da seqüência. De acordo com Floyd basta comparar w_i com w_{2i} para $i \geq 0$. Ao

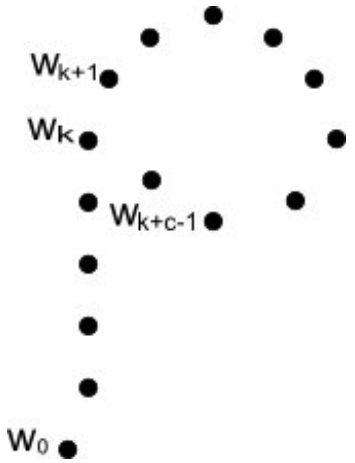


Figura 3.6.3: Seqüência w_i

invés de armazenar todos os valores de w_i, o algoritmos 3.40 armazena apenas 2 valores intermediários.

Algoritmo 3.40 Algoritmo de detecção de ciclos de Floyd

ENTRADA Um valor inicial w_0 e um mapa $\phi : G \to G$.
SAÍDA Um índice $i > 0$ tal que $\phi^i(x) = \phi^{2i}(x)$.

```
x ← φ(w_0)
y ← φ(x)
i ← 1
enquanto x ≠ y faça
    i ← i + 1
    x ← φ(x)
    y ← φ²(y)
retorne i
```

Como podemos ver na figura 3.6.3, a trajetória de w_i possui um prefixo de tamanho $k + 1$ e um ciclo de tamanho c. Considerando o paradoxo do aniversário e o fato de ϕ ser aleatório, é possível estimar que o algoritmo de detecção de ciclos de Floyd realiza a proximadamente $2,82\sqrt{n}$ aplicações de ϕ e $0,94\sqrt{n}$ iterações.

Um problema do algoritmo de Floyd é que w_i precisa ser calculado duas vezes pelo menos para i grande, em outras palavras é preciso calcular ϕ três vezes a cada rodada. O método de Brent [Bre80] usa uma variável auxiliar z, que mantém armazenado o valor de $w_{\ell(i)-1}$, onde $\ell(i)$ é a maior potência de dois contida no índice atual da seqüência w_i. Para cada novo passo da trajetória, é preciso verificar se $z = w_i$, sempre que o índice i for igual a uma potência de dois menos um, é realizada a atribuição $z = w_i$ e isto é repetido até que a colisão aconteça.

Algoritmo 3.41 Algoritmo de detecção de ciclos de Brent

ENTRADA Um valor inicial w_0 e um mapa $\phi : G \to G$.
SAÍDA Inteiros i e j tais que $\phi^i(w_0) = \phi^j(w_0)$.

```
z ← w_0
w ← w_0
i ← 0
ℓ ← 1
enquanto verdade faça
    w ← φ(w)
    i ← i + 1
    se w = z então
        pare
    se i ≥ (2ℓ − 1) então
        z ← w
        ℓ ← 2ℓ
retorne (i, j ← ℓ − 1)
```

Espera-se que o algoritmo de Brent encontre uma colisão após $1,9828\sqrt{n}$ iterações. Isto é um número maior de iterações que o necessário no algoritmo de Floyd. De fato são duas vezes mais, mas o método de Floyd calcula ϕ três vezes por iteração enquanto o método de Brent calcula uma única vez. O preço para este ganho de três vezes é compensado pelo maior número de comparações. Com isso, é possível derivar um algoritmo melhorado de Brent, que realiza um menor número de comparações. Para isso, depende do seguinte teorema:

Teorema 3.6.2. Seja i o menor índice tal que $w_i = w_{\ell(i)-1}$. Então i satisfaz $3/2\ell(i) \leq i \leq 2\ell(i)$.

A demonstração deste teorema pode ser encontrada em [Coh00]

É possível encontrar variações do algoritmo de Brent, mas outras abordagens levam soluções mais eficientes, como veremos com o método de Nivash [Niv04].

O método que veremos agora requer uma ordenação total para os elementos de G. Vamos denotar esta ordenação total pelo símbolo $>$, desde que não aja ambigüidade. O método funciona da seguinte maneira: mantenha uma pilha de pares (w_j, j), onde tanto w_j quanto j são estritamente maiores que seus antecessores na pilha. Inicialmente a pilha está vazia e na rodada i são removidas da pilha todas as entradas (w_k, k), tais que $w_k > w_i$. Se uma colisão $w_i = w_k$ é encontrada, então o algoritmo termina com sucesso e o ciclo tem tamanho $i - j$. Caso contrário, o par (w_i, i) deve ser empilhado e $w_{i+1} = \phi(w_i)$ é computado.

O número esperado de aplicações de ϕ no algoritmo 3.43 é de $k + c(1 + 1/2)$, ou

Algoritmo 3.42 Algoritmo melhorado de detecção de ciclos de Brent

ENTRADA Um valor inicial w_0 e um mapa $\phi : G \to G$.
SAÍDA Inteiros i e j tais que $\phi^i(w_0) = \phi^j(w_0)$.

```
z ← w_0
w ← w_0
i ← 0
ℓ ← 1
enquanto verdadeiro faça
    w ← φ(w)
    i ← i + 1
    se w = z então
        pare
    se i ≥ (2ℓ − 1) então
        z ← w
        ℓ ← 2ℓ
        enquanto i < (3/2ℓ − 1) faça
            w ← φ(w)
            i ← i + 1
retorne (i, j ← ℓ − 1)
```

seja, aproximadamente $1,5666\sqrt{n}$. Nivash provou que o tamanho esperado da pilha é $\lg h + O(1)$, onde h é justamente o número de aplicações de ϕ. Portanto, é necessário apenas uma quantidade logarítmica de memória. Para aumentar a probabilidade de encontrar uma colisão perto do começo do segundo ciclo é possível utilizar a seguinte idéia: dividir os elementos de G em m classes, de forma que o algoritmo passe a organizar os elementos da trajetória aleatória em m pilhas, utilizando a mesma ordenação $>$ dentro de cada classe e as mesmas condições de manipulação das pilhas, finalizando quando for encontrada a primeira colisão em qualquer uma das classes. Com isso, é esperado que o algoritmo encontre a colisão antes de $k + c(1 + 1/(m+1))$ aplicações de ϕ. O tamanho de cada pilha é de $\lg h - \lg m + O(1)$, de modo que o uso de memória é multiplicado por m. O algoritmo 3.43 é especialmente útil para sistemas mono-processados com memória grande.

Como vimos no método do passo pequeno e passo grande, é possível aproveitar o fato do logaritmo discreto pertencer a um intervalo específico de G. Da mesma forma, Pollard [Pol00] descreve um método que atinge o mesmo propósito, com a mesma eficiência e vantagem com relação ao uso de memória que o método de rho de Pollard. A idéia é ter várias trajetórias aleatórias ao mesmo tempo verificando quando elas colidirem, como as seqüências são determinísticas, não existe a necessidade de verificar colisão dentro de cada seqüência. As trajetórias podem ser vistas como cangurus pulando dentro do grupo em consideração. A primeira versão do algoritmo de Pollard trabalha com *cangurus*

Algoritmo 3.43 Algoritmo de detecção de ciclos de Nivash

ENTRADA Um valor inicial w_0 e um mapa $\phi : G \to G$, um número K de pilhas e uma classe de funções $\delta : G \to [0, ..., K-1]$.
SAÍDA Inteiros i e j tais que $\phi^i(w_0) = \phi^j(w_0)$.

```
crie as pilhas S[0], ..., S[K − 1]
inicialize os índices p_0, ..., p_{k−1} para −1
x ← w_0
enquanto verdadeiro faça
    κ ← δ(x)
    encontre o menor índice t ≤ p_κ, tal que S(t) ≥ x ou faça t = −1
    se t ≠ −1 e S(t) = x então
        pare
    i ← i + 1
    p_κ ← t + 1
    se p_κ é muio grande então
        redimensione S(κ)
    S(κ) ← (x, i)
    x ← φ(x)
retorne (i, índice de S(t))
```

mansos (M) e *cangurus selvagens (S)* com respectivos pontos iniciais $w_0(M) = [b]g$ e $w_0(S) = h$. O primeiro canguru é chamado de manso metaforicamente porque sabe-se previamente a sua posição. Ambos permanecem pulando dentro do grupo e conseguem lembrar exatamente o caminho que foi feito. Quando uma colisão ocorre, interpretamos que M capturou S. Se o caminho dos cangurus se cruza, então as trajetórias coincidirão a partir deste evento. M monta armadilhas depois de um certo número de pulos e, se assim conseguir capturar S, pode-se usar esta informação para calcular a posição de S, ou seja, o logaritmo discreto com relação a g.

Esta primeira versão é conhecida como *método lambda* e funciona da seguinte maneira: seja $r > 1$ um inteiro e $v : G \to \{1, ..., r\}$. Os cangurus seguem r trajetórias da forma

$$w_{i+1}(K) = w_i(K) + M(v(w_i(K))),$$

onde $K = M$ ou $k = S$.

A função M multiplica o gerador g do grupo G por tamanhos de salto s_j previamente estabelecidos. Assim, $M(j) = [s_j]g$ tem tamanho $O(\sqrt{a-b})$, onde a e b são limites conhecidos para n. As distâncias viajadas para cada canguru são

$$d_0(K) = 0 \text{ e } d_{i+1}(K) = d_i(K) + s_v(w_i(K)), \text{com } i \in \mathbb{N}$$

e são armazenadas. Note que $w_j(M) = [b + d_j(M)]g$ e $w_j(S) = h + [d_j(S)]g$. O canguru

manso inicia o processo e instala uma armadilha depois de L saltos, de modo que sua distância com relação ao início é $d_L(M)$. Então, o canguru selvagem começa a pular e após cada salto é preciso verificar se caiu em alguma armadilha, isto é, $w_L(M) = w_L(S)$. Depois de um número determinado de passos, S é parado, pois acredita-se que já foi longe o suficiente para permanecer em um lugar seguro (em outras palavras S deve estar em um ciclo que não contém armadilhas). Um novo canguru começa a saltar, a partir de uma posição inicial $w_0(S) = hg^z$, com z pequeno e crescendo para cada novo canguru. Podemos imaginar então que os cangurus iniciam caminhos paralelos na esperança de que pelo menos um seja capturado. Uma representação gráfica deste fenômeno assemelha-se a letra grega λ que dâ nome ao método.

Van Oorschot e Wiener [OW99] mostraram que o número esperado de operações é mínimo se a média dos valores s_j for $\sqrt{a-b}/2$ e S salta $0.7\sqrt{a-b}$ vezes antes de instalar uma armadilha, sendo assim esperado $3.3\sqrt{a-b}$ operações no grupo G. Usando mais memória é possível encontrar um algoritmo que realize $2\sqrt{a-b}$ operações.

3.6.2 Cálculo de índices

De acordo com o teorema de Shoup, a respeito do limite inferior para o número de operações para resolver o PLD, é possível concluir que os algoritmos que tenham complexidade inferior aos métodos apresentados na seção anterior não são genéricos. Isto vale tanto para algoritmos de complexidade subexponencial como para algoritmos de complexidade $O(|G|^C)$, para $C < 1/2$. Nesta seção discutiremos métodos que funcionam em grupos como $\mathbb{F}_p^\times$ ou $E(\mathbb{F}_p)$. Esta família de algoritmos é denominada *cálculo de índices*.

A idéia geral do método é a seguinte: dado um grupo multiplicativo G, deseja-se construir uma extensão K de G, tal que o problema possa ser resolvido em K e posteriormente reduzido para o grupo G através de uma relação de equivalência $\sim$. Para isso, é preciso que o grupo K possua uma *base suave de primos* , isto é, um conjunto de elementos $P = \{p_1, ..., p_r\}$, denominados *primos* tais que para $a \in K$, então

$$a = \prod_{i=1}^{r} p_1^{e_1} ... p_r^{e_r}.$$

Estes primos precisam ter um *limite suave*, isto é, serem tais que sua norma em G é menor que um determinado B. Um elemento de G pode ser visto como a composição destes primos juntamente com uma relação de equivalência $\sim$, que mapeia elementos de K em elementos de G.

Desta forma, para resolver $g^n = h$ no grupo G, são escolhidos os valores g^i e hg^j em K, encontrando representações suaves (produtos de pequenos primos), tornando possível eliminar relações de modo a encontrar uma solução em K, que podemos reduzir a G.

Exemplo 3.6.3. Seja $G = \mathbb{F}_p^{\times}$. Então, G é isoformo a $(\mathbb{Z}, \times)/\sim$, onde a relação de equivalência é dada pela congruência $n_1 \equiv n_2 \pmod{p}$. Desta forma, o conjunto P é dado pelos primeiros números primos. Neste caso, o limite suave é determinado por um número inteiro B, de forma que cada primo pertencente a P é menor que B.

Exemplo 3.6.4. No caso de corpos de extensão, $\mathbb{F}_{p^d}^{\times}$, podemos representar os elementos do grupo multiplicativo através de polinômios de grau d. O conjunto P é formado por polinômios irredutíveis e um elemento é denominado *B-suave* se puder ser fatorado como a multiplicação de polinômios irredutíveis de grau menor que B.

O algoritmo 3.44 apresenta uma forma genérica de aplicação deste método para um grupo G.

Para um grupo G específico é possível encontrar pequenas variações do método apresentado no algoritmo 3.44, principalmente com relação a forma como são encontradas as relações e os métodos de álgebra linear responsáveis por encontrar a solução.

Definição 3.6.5. Seja N um número natural. Então

$$L_N(\alpha, c) = \exp((c + o(1))(\ln N)^{\alpha}(\ln \ln N)^{1-\alpha}),$$

onde $0 \leq \alpha \leq 1$ e $c > 0$. $L_N(\alpha, c)$ varia entre complexidade polinomial para $\alpha = 0$ e exponencial para $\alpha = 1$. Para $\alpha < 1$ a complexidade é denominada *subexponencial*.

Duas variações do algoritmo 3.44 podem ser destacadas para resolver o PLD em corpos $\mathbb{F}_{p^d}^{\times}$:

- se $d < \sqrt{\ln p}$, então o método da *armadilha do corpo numérico* pode ser utilizado. O tempo de execução esperado é de $L_{p^d}(1/3, 1, 923)$,
- se $d > (\ln p)^2$, então o método da *armadilha do corpo de funções*, de Coppersmith, pode ser utilizado com corpos de característica pequena, tais como $\mathbb{F}_{2^d}$, com tempo de execução esperado de $L_{2^d}(1/3, 1, 588)$.

Atualmente não se conhece algoritmo subexponencial capaz de resolver o PLD em corpos finitos onde $\sqrt{\ln p} < d < (\ln p)^2$.

3.6.3 Cálculo de índices em curvas elípticas

O método do cálculo de índices funciona em alguns casos especiais de curvas elípticas, alcançando complexidade subexponencial, mas dados uma curvas e pontos genéricos, os melhores algoritmos são lentos, com complexidade exponencial $O(\sqrt{p})$.

Algoritmo 3.44 Cálculo de índices

ENTRADA Um grupo G de ordem N, dois elementos $g, h \in G$, com $h \in <g>$.
SAÍDA Um inteiro t tal que $h = [t]g$.

Construção da base de fatores
Escolha um limite suave B e construa uma base de fatores dada por $P_B = \{\pi_1, ..., \pi_{n_B}\}$, onde π_i é um primo suave.

Produção de relações
Encontre relações da forma

$$[a_i]g + [b_i]h = \sum_{j=1}^{n_B} [e_{i,j}]\pi_j, \text{para } i = 1, 2, ...$$

Faça $c = n_B$. Construa a matriz A com c linhas definidas como vetores

$$(e_{i,1}, e_{i,2}, ..., e_{i,c}), \text{para } i = 1, ..., c.$$

Armazene os vetores $a = (a_1, ..., a_c)$ e $b = (b_1, ..., b_c)$.
Faça

$$v = (e_{c+1,1}, e_{c+1,2}, ..., e_{c+1,c}).$$

Processe a matriz A para tentar obter um valor de c menor.

Álgebra linear
Calcule a solução de $xA = v \pmod N$ ou encontre uma solução para $xA = 0 \pmod N$.

Extração da solução
A matriz A e os vetores satisfazem a seguinte relação em G:

$$(a^t, b^t) \times \begin{bmatrix} g \\ h \end{bmatrix} = A \times \Pi, \text{onde } (a^t, b^t) = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_c & b_c \end{bmatrix} \text{ e } \Pi = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \vdots \\ \pi_c \end{bmatrix}.$$

Seja $x = (x_1, ..., x_c)$ uma solução encontrada no passo três. Seja $\alpha = xa^t$ e $\beta = xb^t$.
se $xA = 0$ e $(\beta, N) = 1$ **então**
    retorne $\log_g(h) = -\frac{\alpha}{\beta} \pmod N$
se $xA = v$ **então**
    retorne $\log_g(h) = -\frac{\alpha - a_{c+1}}{\beta - b_{c+1}} \pmod N$

Nesta seção serão apresentadas as razões pelas quais estes ataques não funcionam em curvas genéricas. Primeiramente, é preciso encontrar uma forma de estender o grupo elíptico $E(\mathbb{F}_q)$. Este problema é definido com maior detalhe da seguinte maneira [Sil07]:

Definição 3.6.6. Seja $E/\mathbb{F}_q$, $H, G \in E(\mathbb{F}_q)$ tal que $H = nG$. O problema de estender a curva E é encontrar os seguintes valores

- um anel R contido num corpo K,
- um ideal $\mathbf{p}$ de R tal que $R/\mathbf{p} = \mathbb{F}_{\mathbf{q}}$,
- uma curva elíptica $\hat{E}/K$ que satisfaça $\hat{E} \equiv E \pmod{\mathbf{p}}$,
- pontos $\hat{S}, \hat{T} \in \hat{E}(K)$, tais que

$$\hat{S} \equiv S \pmod{\mathbf{p}} \text{ e } \hat{\mathbf{T}} \equiv \mathbf{T} \pmod{\mathbf{p}}.$$

Silverman [Sil07] classificou as tentativas de resolver o problema conforme o tipo do anel R, podendo este ser um *anel local* (por exemplo o anel dos números p-ádicos $\mathbb{Z}_p$) ou um *anel global* (por exemplo o anel dos números inteiros $\mathbb{Z}$) e o tipo dos pontos $\hat{S}, \hat{T}$, podendo estes serem pontos de torção ou não.

Desta forma é possível dividir as tentativas de ataques à curvas elípticas usando o método do cálculo de índices em quatro categorias:

1. extensão local sem pontos de torção,
2. extensão local com pontos de torção,
3. extensão global sem pontos de torção,
4. extensão global com pontos de torção.

Extensão local sem pontos de torção

Esta primeira categoria tem como objetivo estender um ponto $Q \in E(\mathbb{F}_q)$ para um ponto q-ádico $\hat{Q} \in \hat{E}(\mathbb{Q}_q)$. Isto pode ser feito através do lema de Hensel. A idéia é iniciar o processo com um ponto $Q = Q_1$ e primeiramente calcular

$$Q_2 \pmod{q^2},$$

seguido do cálculo do ponto

$$Q_3 \pmod{q^3},$$

e assim por diante.

Para resolver o problema do logaritmo discreto em $E(\mathbb{F}_q)$, supõe-se que foi possível estender $S, T \in E(\mathbb{F}_q)$ aos pontos $\hat{S}, \hat{T} \in \hat{E}(K)$. Multiplicando-se por $N = |E(\mathbb{F}_q)|$, os pontos $N\hat{T}$ e $N\hat{S}$ estão no chamado *grupo formal* de $E(\mathbb{Q}_q)$. Este grupo tem uma função $\hat{\log}$ eficiente para cálculo do logaritmo discreto. Com isso, é possível computar

$$m = \frac{\hat{\log}(N\hat{T})}{\hat{\log}(N\hat{S})} \in \mathbb{Q}_q.$$

Mas para obter o valor de $\hat{S}$ e $\hat{T}$, é preciso calcular a seqüência $S_1, S_2, ...$ e $T_1, T_2, ...$, da mesma forma que foi realizado para obter a seqüência $Q_1, Q_2, ...$ Mas além disso, é preciso que as relações $T_i = mS_i$ sejam mantidas pelo método, de forma que ao final do processo, $\hat{T} = m\hat{S}$. Mas a cada etapa deste processo, existem q escolhas para S_i e T_i, embora para um determinado valor de S_i, exista apenas um valor para T_i tal que $T_i = mS_i$ e não existe nenhum método conhecido para resolver este problema sem que o valor de m seja conhecido.

Extensão local com pontos de torção

Seja $Q \in E(\mathbb{F}_q)$ um ponto de ordem n (tal que $n \neq p$). É preciso estender Q para um ponto q-ádico $\hat{Q} \in E(\mathbb{Q}_q)$. Isto pode ser feito de diversas maneiras, mas é possível demonstrar que existe uma única maneira de fazer esta extensão de modo que $\hat{Q}$ seja também um ponto de n-torção em $E(\mathbb{Q}_q)$. Ou seja, existe um único valor de $\hat{Q}$ tal que

$$n\hat{Q} = \hat{\infty} \text{ e } \hat{Q} \equiv Q \pmod{\mathbf{p}}.$$

De fato, é fácil computar valores de $Q_k \pmod{\mathbf{q^k}}$, de modo que é possível obter os valores de $\hat{S}$ e $\hat{T}$, tais que $\hat{T} = m\hat{S}$. Contudo, para resolver o problema do logaritmo discreto em K é preciso multiplicar por $N = |E(\mathbb{F}_q)|$, para ser possível trabalhar no grupo formal. Mas isto faz com que a relação $\hat{T} = m\hat{S}$ seja transformada em $\hat{\infty} = m\hat{\infty}$ e não se conhece uma forma de resolver este problema sem ser no grupo formal.

Extensão global sem pontos de torção

Este método pode ser dividido em duas abordagens distintas:

- método da extensão fácil,
- método da extensão difícil.

No *método da extensão fácil* , nós podemos escolher a curva estendida $\hat{E}$ e os pontos estendidos $\hat{Q}_1, ..., \hat{Q}_r \in \hat{E}(\mathbb{Q})$ simultaneamente usando métodos elementares tais como

álgebra linear. No *método da extensão difícil* , nós podemos usar métodos elementares para estender a curvas e até mesmo alguns pontos. Então nós precisamos estender pontos adicionais que não foram considerados na construção da extensão original.

1. **Método da extensão fácil.**

   Se considerarmos os coeficientes da curva elíptica como variáveis, então podemos facilmente determinar a curva $\hat{E} \equiv E \pmod{q}$ e os pontos $\hat{S} \equiv S \pmod{q}$ e $\hat{T} \equiv T \pmod{q}$. Assim, se os pontos $\hat{S}$ e $\hat{T}$ forem linearmente dependentes, então é simples obter a relação $\hat{T} = m\hat{S}$ e a redução módulo q resolve o problema do logaritmo discreto.

   De maneira geral, podemos encontrar combinações lineares aleatórias da forma

   $$Q_i = a_i S - b_i T,$$

   para $i = 1, 2, ..., r$.

   Observe que se pudermos estabelecer a relação linear

   $$n_1 Q_1 + ... + n_r Q_r = \infty,$$

   então provavelmente conseguimos resolver o problema do logaritmo discreto usando

   $$(n_1 a_1 + ... + n_r a_r)S = (n_1 b_1 + ... + n_r b_r)T.$$

   Uma forma cúbica genérica $F(X, Y, Z) = 0$, tem dez coeficientes. Com isso, podemos usar álgebra linear para estender E em até nove pontos $\hat{Q}_1, ..., \hat{Q}_9 \in \hat{E}(\mathbb{Q})$.

   Infelizmente o teorema 3.6.7 estabelece condições que permitem concluir que a probabilidade de encontrar pontos estendidos linearmente dependentes é menor que $1/q$, para curvas elípticas $E/\mathbb{F}_q$.

   Teorema 3.6.7. (Masser) Seja E_U uma família parametrizada de curvas elípticas, onde $U = (U_1, ..., U_n)$ e seja $Q_{1,U}, ..., Q_{r,U}$ uma família parametrizada de pontos linearmente independentes. Então

   $$\{u \in \mathbb{Q}^n : Q_{1,u}, ..., Q_{r,u} \text{ são pontos linearmente dependentes em } E_U(\mathbb{Q})\}$$

   é um conjunto pequeno (de dendidade zero).

Logo, este método não representa um ganho para a solução do problema do logaritmo discreto em curvas elípticas. Algumas idéias permitem escolher curvas de forma cuidadosa na tentatica de aumentar a probabilidade de encontrar pontos linearmente independentes, como por exemplo a heurística sugerida pela conjectura de Birch-Swinnerton-Dyer, mas mesmo neste caso o ganho não é grande o suficiente.

2. **Método da extensão difícil.**

   Seja a curva elíptica $E/\mathbb{F}_q$ e $S \in E(\mathbb{F}_q)$. Seja $\hat{S} \in \hat{E}/\mathbb{Q}$ a extensão de S. Suponha que exista a extensão $\hat{T} \in \hat{E}(\mathbb{Q})$. Estamos procurando por m tal que $T = mS$ e $\hat{T} = m\hat{S}$. Mas é possível provar que é preciso $O(q^2)$ bits para representar $\hat{T}$. Logo, o método da extensão difícil é inviável em curvas elípticas para fins criptográficos, pois em geral temos que $q \sim 2^{160}$.

Extensão global com pontos de torção

Sejam $S, T \in E(\mathbb{F}_q)$. Vamos estender S e T para os pontos $\hat{S}, \hat{T} \in \hat{E}(\mathbb{Q}_{\text{tor}})$. Com isso, a relação $\hat{T} = m\hat{S}$ é mantida e de fato é muito simples encontrar m. Por exemplo, é possível ter $\hat{T} = m\hat{S} \pmod{p}$, para pequenos primos $P = \{3, 5, 7, ...\}$ e usar o teorema do resto chinês para obter o valor de m.

Mas este método também não funciona pois $\hat{E}(\mathbb{Q}_{\text{tor}})$ é conjunto com número pequeno de elementos. Em geral temos que

Teorema 3.6.8. (Mazur) Para curvas elípticas $\hat{E}/\mathbb{Q}$, temos que $|\hat{E}(\mathbb{Q}_{\text{torção}})| \leq 16$.

Mas podemos pensar em extensões para corpos distintos de $\mathbb{Q}$. Se considerarmos um corpo numérico $K/\mathbb{Q}$, então podemos sempre encontrar pontos $\hat{S}, \hat{T} \in E(K)_{\text{torção}}$ e um ideal $\mathbf{p}$ do anel de inteiros R_K tal que $\hat{S} = S \pmod{\mathbf{p}}$ e $\hat{T} = T \pmod{\mathbf{p}}$. Então o problema do logaritmo discreto pode ser resolvido em $E(K)$. Para fazer isso é preciso trabalhar no corpo K. Se $K/\mathbb{Q}$ tem grau pequeno isto é possível, mas o teorema 3.6.9 estabelece um limite inferior para $[K : \mathbb{Q}]$.

Teorema 3.6.9. (Serre) Seja $\hat{E}/\mathbb{Q}$ e $T \in E(K)$ um ponto de ordem n. Então geralmente temos que

$$[K : \mathbb{Q}] \geq C,$$

onde $C \sim cn^4$, para uma constante c.

Resumo

Nesta seção vimos de forma simplificada as razões pelas quais não foi possível aplicar o ataques de cálculo de índices em grupos elípticos. Podemos resumir estas razões na seguinte tabela:

Tabela 3.6: **Dificuldades para resolver o PLD em curvas elípticas**

Extensão local sem pontos de torção	falha porque a extensão para pontos em $\hat{E}(\mathbb{Q}_p)$ perde a relação $\hat{T} = m\hat{S}$
Extensão local com pontos de torção	falha porque a extensão de pontos em $\hat{E}(\mathbb{Q}_p)_{\text{tor}}$ mantêm a relação $\hat{T} = m\hat{S}$, mas é impossível determinar m
Extensão global sem pontos de torção	método de extensão fácil: falha porque os pontos estendidos são independentes método da extensão díficil: falha porque não existe um método para estender o ponto $\hat{T}$
Extensão global com pontos de torção	a extensão para $\hat{E}(\mathbb{Q})_{\text{tor}}$ ou $\hat{E}(K)_{\text{tor}}$ falha porque $\lvert E_{\mathbb{Q}_{\text{tor}}}\rvert$ é muito pequeno ou porque $[K : \mathbb{Q}]$ é muito grande

3.6.4 Transferência de logaritmo discreto

Nesta seção serão apresentado três métodos para transferir o problema do logaritmo discreto em curvas elípticas para grupos onde este problema seja mais fácil de resolver. Com o estudo destas técnicas é possível determinar condições que devem ser evitadas na escolha de parâmetros de criptossistemas baseados em curvas elípticas.

Transferência para espaços vetoriais sobre $\mathbb{F}_q$

Seja $C/\mathbb{F}_q$, onde $q = p^d$, uma curva hiperelíptica de genus g e assuma que p divide $|\operatorname{Pic}^0_C|$. Para este caso, é possível construir um mapa de $\operatorname{Pic}^0_C[p]$ para $\Omega^0(C)$, o espaço vetorial sobre $\mathbb{F}_q$ de holomorfismos diferenciais em C. Este espaço é isomorfo a $\mathbb{F}_q^{2g-1}$. Embora computar o mapa exija algum esforço, a complexidade de avaliar este mapa é $O(\lg q)$. Com isso, é possível transferir o problema do logaritmo discreto em $\operatorname{Pic}^0_C[p]$ para $\mathbb{F}_q^{2g-1}$, onde este problema pode ser resolvido através de método como o algoritmo de Euclides para $g = 1$, ou técnicas genéricas de álgebra linear para o caso geral.

Estes métodos têm complexidade $O((2g-1)\lg(q)^k)$, para uma constante pequena k. O caso geral para esta técnica é apresentado em [Rüc99] e especificamente para curvas elípticas em [SA98, Sem98, Sma99].

Transferência através de emparelhamentos bilineares

Seja $C/\mathbb{F}_q$, onde $q = p^d$, uma curva hiperelíptica de genus g. Considerando o problema do logaritmo discreto em um subgrupo de Pic_C^0 de ordem ℓ. Seja k um inteiro tal que $\ell | q^k - 1$. O emparelhamento de Tate-Lichtembaum é um mapa da forma

$$T_\ell : J_C(\mathbb{F}_q)[\ell] \times J_C(\mathbb{F}_{q^k}) \to \mathbb{F}_{q^k}^\times[\ell].$$

Com isso, é possível transferir o problema do logaritmo discreto em curvas hiperelípticas para o grupo multiplicativo $\mathbb{F}_{q^k}^\times$. Se k for pequeno o suficiente, podemos resolver o PLD neste grupo ao invés de $\mathrm{Pic}_C^0[\ell]$.

No caso de curvas elípticas supersingulares, como é o caso das curvas elípticas usadas neste trabalho, sabe-se que o valor de k é pequeno, de modo que um cuidado maior na escolha dos tamanhos dos grupos precisa ser tomada, tendo em vista um valor de k tal que o PLD no grupo $\mathbb{F}_{q^k}^\times$ seja difícil de ser resolvido.

Seja a curva elíptica $E/\mathbb{F}_q$ e e_ℓ um emparelhamento bilinear definido sobre pontos de ordem ℓ desta curva. De fato, dado um valor ponto aleatório $R \in E(\mathbb{F}_q)$ e os pontos $S, T \in E(\mathbb{F}_q)$, tais que $T = nS$, temos que $e_\ell(S, R) = \mu$, onde μ é uma raíz ℓ-ésima da unidade. Assim, temos que

$$e_\ell(T, R) = e_\ell(S, R)^n.$$

Logo, é possível computar o valor de n usando o método do cálculo de índices, através de algoritmos de complexidade menor que aqueles conhecidos para o PLD em curvas elípticas, isto é, de complexidade subexponencial.

Transferência através da descida de Weil

A descida de Weil, também conhecida por restrição escalar, é uma técnica bastante usada em geometria algébrica e pode ser utilizada em objetos geométricos como curvas, diferenciais e grupos de Picard.

A idéia do método é relacionar elementos de dimensão t de um corpo separável L, de grau d, em elementos de dimensão td do corpo base K. Um exemplo clássico é a transformação de curvas sobre os números complexos $\mathbb{C}$ em superfícies sobre os números reais $\mathbb{R}$.

Em curvas elípticas a técnica pode ser usada em corpos compostos, isto é, corpos $\mathbb{F}_{q^k}$, para $k > 1$. Neste caso, é possível usar variedades sobre $\mathbb{F}_q$, tentando usar estruturas adicionais desta variedade de dimensão maior para obter algum tipo de vantagem.

O algoritmo GHS [GS99] é o mais conhecido para o uso do método da descida de Weil. Este método aplica-se a corpos binários $\mathbb{F}_{2^d}$, onde d é um número composto. Galbraith, Hess e Smart [GHS02b] aplicaram o método GHS em outras curvas usando isogenias de grau pequeno. Neste trabalho, os autores mostraram que para o corpo $\mathbb{F}_{2^{155}}$, existem por volta de 2^{123} classes de isogenia de curvas elípticas $E/\mathbb{F}_{2^{155}}$, levando a curvas D, tais que o logaritmo discreto em $D/\mathbb{F}_{2^5}$, pode ser resolvido mais eficientemente que em $E/\mathbb{F}_{2^{155}}$.

Capítulo 4

Criptografia baseada em emparelhamentos

> O segredo é uma rede: rota uma das malhas, rompem-se facilmente as outras.
>
> Victor Hugo

4.1 Introdução

Existem dois emparelhamentos conhecidos hoje em dia, são eles os emparelhamentos de Weil e o de Tate. A primeira aplicação de emparelhamentos foi para reduzir o problema do logaritmo discreto em curvas elípticas supersingulares para o problemas do logaritmo discreto em corpos finitos. Esta redução é conhecida como ataque MOV usando o emparelhamento de Weil e ataque FR, usando o emparelhamento de Tate.

A primeira utilização de emparelhamento para prover serviços criptográficos foi realizada por Joux [Jou00], em 2000, para construir um protocolo de acordo de chaves entre três partes em uma única rodada. Em 2001, Boneh e Franklim [BF01] utilizaram emparelhamentos para criar um protocolo de ciframento baseado em identidades, cuja segurança e eficiência foi comprovada.

Neste capítulo serão definidos os modelos de criptografia baseada em identidades, criptografia sem certificados e cifrassinatura, permitindo o estudo de protocolos que implementem esses modelos de forma eficiente usando emparelhamentos bilineares.

4.1.1 Modelo de atacante

Ao analisar a segurança de um criptossistema é preciso levar em conta o poder do atacante, isto é, que tipo de ações este pode executar, o acesso a informações e o grau de interatividade do atacante com o procotolo a ser atacado.

Definição 4.1.1. Um algoritmo *probabilístico em tempo polinomial* é aquele que tem acesso a uma fonte aleatória de informação para tomar decisões em tempo polinomial.

Ao analisar a segurança de um protocolo criptográfico é preciso considerar três perguntas:

1. *Que informação é conhecida pelo atacante?* Isto é, ele tem acesso a textos cifrados apenas ou também a pares de textos em claro e cifrado?
2. *O adversário tem poder de escolher a informação a que tem acesso?* Ou seja, o atacante tem controle sobre o texto claro correspondente a um texto cifrado?
3. Em caso afirmativo para a pergunta anterior, *o adversário tem capacidade de fazê-lo de forma adaptativa?* Neste caso, o atacante pode escolher as mensagens no decorrer do ataque ao invés de ter que fazê-lo *a priori*.

Dependendo das respostas às perguntas anteriores é possível ter os seguintes tipos de ataques:

(i) ataques de texto cifrado conhecido;

(ii) ataques de texto em claro conhecido;

(iii) ataques de texto em claro escolhido;

(iv) ataques de texto cifrado escolhido;

(v) ataques adaptativos de texto em claro escolhido (CPA);

(vi) ataques adaptativos de texto cifrado escolhido (CCA).

A demonstração de segurança contra atacantes ativos exige estabelecer uma forma inteligente de lidar com a capacidade de adaptação do adversário. Uma maneira natural de representar esta situação é através de *jogos*. Através deste método, é possível estabelecer regras para um jogo entre o usuário e o adversário, refletindo a situação do uso do criptossistema na prática.

O idéia do jogo consiste em definir regras tais que o adversário ganha se consegue vantagem probabilística para determinar se uma cifra corresponde a uma mensagem m_0

ou m_1. A escolha de qual mensagem será cifrada no jogo deve ser feita pelo usuário de forma aleatória, de modo que uma vantagem no jogo representa uma fraqueza do criptossistema. A demonstração de segurança termina ao estabelecer um simulador S que seja capaz de resolver um problema difícil se e somente se o adversário tem vantagem no jogo. Com isso, a resolução do problema difícil fica atrelada a insegurança do protocolo.

Para ser mais específico, defini-se o adversário através do par de algoritmos (A_1, A_2). O primeiro recebe os parâmetros do sistema e gera o par de mensagens m_0 e m_1, além de uma cifra c, correspondente a m_0 ou m_1 com probabilidades idênticas, ou seja, $P[c = e(m_0)] = P[c = e(m_1)] = 1/2$. O algoritmo A_2 recebe as duas mensagens e a cifra c e deve retornar $i = 0$ ou $i = 1$. O adversário ganha se m_i foi a mensagem escolhida por A_1.

Definição 4.1.2. No contexto de assinaturas digitais, uma *falsificação existencial* é definida como uma dupla (m, σ), onde m é uma mensagem qualquer e σ é uma assinatura válida. A mensagem m não pode ter sido assinada previamente por um usuário legítimo e o conteúdo de m pode ser desprovido de qualquer significado.

4.2 Criptografia Baseada em identidades

4.2.1 Modelo

O advento da criptografia assimétrica representou um grande avanço na segurança dos computadores, especialmente porque solucionou o problema da troca de chaves para algoritmos de criptografia simétrica. Mas ataques surgiram aproveitando o fato de não haver garantia sobre quem é o verdadeiro dono de uma chave pública, de forma que um usuário pode se fazer passar por outro facilmente, tornando necessário um mecanismo de associação entre a chave pública e seu dono.

Para resolver este problema foi criado o mecanismo de certificação digital, que utiliza uma estrutura hierárquica de autoridades certificadoras, capazes de garantir adequadamente a posse de uma dada chave pública. Este mecanismo funciona muito bem em organizações abertas como a internet. Em 1984 um modelo de criptografia baseada em identidades (CBI) foi proposto por Shamir [Sha85]. Este modelo tinha como objetivo evitar o uso da certificação digital, utilizando a própria identidade do usuário como sua chave pública. Esta identidade poderia ser um endereço de e-mail, CPF, nome completo, ou uma combinação destes elementos. A chave privada seria obtida através de uma autoridade de confiança (TA - trust authoraty). Com isso, certificados digitais seriam necessários apenas na identificação desta autoridade central, reduzindo drasticamente o seu uso. Um problema que existe nesta idéia é o conhecimento da chave privada pela autoridade central, sendo necessária uma confiança total pelo usuário, o que exige uma série de cuidados do ponto de vista prático e legal. Por outro lado, não é necessária toda

a infra-estrutura hierárquica de autoridades para o gerenciamento das chaves, tornando o modelo mais simples e adequado para organições onde a hierarquia é natural e seus limites são bem controlados.

Shamir desenvolveu um esquema de *assinatura baseada em identidades*, cujo funcionamento é parecido com o RSA. Ele também conjecturou sobre a existência de um esquema para ciframento, problema que foi resolvido na prática pelos criptossistemas de Boneh e Franklim [BF01], cuja segurança foi rigorosamente demonstrada e Sakai e Kasahara [SOK00].

Esquema de Assinatura Baseado em Identidades de Shamir

O esquema de assinatura baseado em identidades de Shamir é o modelo que outros esquemas de CBI seguem, sendo divido em quatro etapas:

1. **Configuração:** esta etapa é realizada pela autoridade de confiança para gerar os parâmetros globais do sistema e a chave mestra, que garantirá que apenas a TA consegue gerar as chaves privadas.

2. **Geração de chave privada:** este algoritmo recebe como entrada os parâmetros globais do sistema, a chave mestra e a identidade de um usuário, retornando a chave privada associada.

3. **Assinatura:** dada uma chave privada e uma mensagem, o algoritmo retorna a assinatura.

4. **Verificação:** dada uma identidade, uma mensagem e uma assinatura, o algoritmo retorna verdadeiro caso a assinatura daquela mensagem corresponda à identidade fornecida, e retorna falso caso contrário.

Este modelo pode ser implementado de diversas formas, e ultimamente inúmeros artigos surgiram propondo uma maneira segura para realizar criptografia baseada em identidades, a grande maioria utilizando emparelhamentos.

Com isso temos duas formas de associar uma chave pública a seu dono: certificados digitais e criptografia baseada em identidades. As duas soluções parecem ser complementares, isto é, dependendo do ambiente em que está sendo considerado e suas características, é necessários escolher entre as vantagens e desvantagens de cada um dos sistemas. Uma possibilidade que parece bastante interessante é a utilização conjunta destas soluções, para tentar atingir um equilibrio. A seguir serão descritas com mais detalhes estas diferenças entre as duas soluções.

4.2.2 Esquemas

Nesta seção serão apresentados esquemas de criptografia baseada em identidades para assinatura digital, ciframento e acordo de chaves. As demonstrações de segurança podem ser encontradas com detalhes nos trabalhos originais dos respectivos autores. Os protocolos foram escolhidos levando em consideração a eficiência das operações e a demonstração de segurança segundo um modelo de atacante plausível.

Assinatura

Será descrito agora um protocolo eficiente para realizar assinatura baseada em identidades (ABI) [BLMQ05]. Este esquema utiliza apenas um emparelhemento na verificação, todavia precisa de modelo de segurança com suposições mais fortes que de outros criptossistemas baseados em identidades de ABI com segurança demonstrável.

Com isso, define-se as seguintes noções de segurança:

Definição 4.2.1. Sejam os grupos G_1, G_2 e G_3 grupos utilizados para o cálculo de emparelhamentos bilineares da forma $e : G_1 \times G_2 \rightarrow G_3$. Sejam $P \in G_1$ e $Q \in G_2$ geradores dos respectivos grupos.

O problema *q-forte de Diffie-Hellman* (q-FDH) nos grupos G_1 e G_2 consiste em, dados $(q+2)$ pontos $(P, Q, \alpha Q, ..., \alpha^q Q)$ como entrada, encontrar o par $(c, 1/(c+\alpha P))$ com $c \in \mathbb{Z}_p^*$.

O problema *q-bilinear de inversão de Diffie-Hellman* (q-BIDH) nos grupos G_1, G_2 e G_3 consiste em, dados os pontos $(P, Q, \alpha Q, ..., \alpha^q Q)$ calcular $e(P, Q)^{1/\alpha} \in G_3$.

Desta forma, o protocolo de assinatura digital baseada em identidades pode ser descrito do seguinte modo:

Configuração: Dado k, o PKG escolhe grupos de mapas bilineares, G_1, G_2 e G_t, de ordem prima $p > 2^k$ e geradores $Q \in G_2$, $P = \psi(Q) \in G_1$, $g = e(P, Q) \in G_t$. Escolha um s aleatório pertencente a Z_p^*, uma chave pública do sistema $Q_{pub} = sQ \in G_2$, funções de resumo criptográfico $H_1 : \{0,1\}^* \leftarrow \mathbb{Z}_p^*$, $H_2 : \{0,1\}^* \times G_T \leftarrow \mathbb{Z}_p^*$.

Geração da chave privada: Para uma identidade ID, a chave privada é $S_{ID} = \frac{1}{H_1(ID)+s} P$.

Assinatura: Dada uma mensagem $M \in \{0,1\}^*$ e a identidade ID, utiliza-se x aleatório pertencente a $\mathbb{Z}_p^*$ para calcular $r = g^x$ e $h = H_2(M, r) \in \mathbb{Z}_p^*$. Calcula-se $S = (x + h)S_{ID_d}$. A assinatura é dada por $\delta = (h, S)$.

Verificação: Dada a assinatura $\delta = (h, S)$ sobre a mensagem M e a identidade ID. Aceite a mensagem se $h = H_2(M, e(S, H_1(ID)Q + Q_{pub})g^{-h})$.

Definição 4.2.2. Um esquema de ABI é seguro segundo o conceito de inforjabilidade existencial sobre ataques adaptativos de texto claro escolhido se nenhum adversário com acesso a algoritmos probabilísticos em tempo polinomial consegue vantagem não negligível no seguinte jogo:

1. O desafiante executa o algoritmo de configuração para gerar os parâmetros do sistema e os envia ao adversário.

2. O adversário executa uma série de consultas aos seguintes oráculos:

   - Extração de chave: retorna a chave privada de identidades arbitrárias
   - Assinatura: produz assinaturas de mensagens arbitrárias usando a chave privada de identidades arbitrárias

3. O adversário produz o trio (ID^*, M^*, σ^*), de modo que nenhuma consulta sobre a chave privada de ID^* foi realizada nas etapas anteriores e (M^*, ID^*) não foi submetida ao oráculo de assinaturas. O adversário ganha se (ID^*, M^*, σ^*) for aceito pelo algoritmo de verificação.

Teorema 4.2.3. Suponha que A seja um adversário que realiza q_{h_1} e q_{h_2} consultas aos oráculos referentes às funções de resumo criptográfico H_1 e H_2 respectivamente. Suponha que A realiza q_s consultas ao oráculo de assinaturas. Assuma que em um tempo t, A produz uma forja com probabilidade $\epsilon \geq 10(q_s + 1)(q_s + q_{h_2})/2^k$. Então, existe um algoritmo B que é capaz de resolver o problema q-FDH para $q = q_{h_1}$ com o seguinte tempo esperado

$$t' \leq 120686 q_{h_1} q_{h_2}(t + O(q_s \tau_p))/(\epsilon(1 - q/2^k)) + O(q^2 \tau_{\text{mult}}),$$

onde τ_p e τ_{mult} representam o custo das operações de cálculo do emparelhamento e multiplicação por escalar em G_2.

Ciframento

O esquema de ciframento baseado em identidades de Boneh e Franklin [BF01] é importante por ser o primeiro esquema desse tipo, resolvendo um problema em aberto sugerido por Shamir. Uma extensão deste método, com demonstração de segurança com relação a ataques de texto cifrado escolhido, pode ser encontrada em [BF03].

A descrição do modelo é dada a seguir:

Configuração: Dado k, o PKG escolhe grupos de mapas bilineares, G_1, G_2 e G_t, de ordem prima $p > 2^k$ e geradores $Q \in G_2$, $P = \psi(Q) \in G_1$, $g = e(P, Q) \in G_t$. Escolha

um s aleatório pertencente a Z_p^*, uma chave pública do sistema $Q_{pub} = sQ \in G_2$, funções de resumo criptográfico $H_1 : \{0,1\}^* \leftarrow G_1$, $H_2 : G_t \leftarrow \{0,1\}$.

Geração da chave privada: Para uma identidade ID, a chave privada é $S_{ID} = H_1(ID)P$.

Ciframento: Dada uma mensagem $M \in \{0,1\}^*$ e a identidade ID, utiliza-se r aleatório pertencente a $\mathbb{Z}_p^*$ para calcular $R = rP$ e $S = e(Q_{pub}, H_1(ID))$. A cifra é dada por (R, c), onde $c = m \oplus H_2(rS)$.

Deciframento: Dada a cifra (R, c) sobre a mensagem M e a identidade ID. Calcule $T = e(R, S_{ID})$ e $M = c \oplus H_2(T)$.

Acordo de Chaves

Em [BM05] é proposto um esquema eficiente de acordo de chaves entre duas partes, onde o problema da custódia da chave privada pode ser resolvido sem custo extra. A seguir é descrito o esquema de acordo de chaves com custódia de chaves:

Configuração: Dado k, o PKG escolhe grupos de mapas bilineares, G_1, G_2 e G_t, de ordem prima $p > 2^k$ e geradores $Q \in G_2$, $P = \psi(Q) \in G_1$, $g = e(P, Q) \in G_t$. Escolha um s aleatório pertencente a Z_p^*, uma chave pública do sistema $Q_{pub} = sQ \in G_2$, funções de resumo criptográfico $H_1 : \{0,1\}^* \leftarrow \mathbb{Z}_p^*$, $H_2 : G_t \leftarrow \{0,1\}$.

Geração da chave privada: Para uma identidade ID e $a \in \mathbb{Z}_p^*$ a chave pública é dada por $(a + s)Q$, que pode ser calculada por $aQ + sQ$. A chave privada é $S_{ID} = (a + s)^{-1}Q$.

Acordo de chaves: Suponha que Alice e Bob tenham identidade representada por ID_A e ID_B respectivamente. Alice gera $x_A \in \mathbb{Z}_p^*$ e calcula $A_K = x_a(bQ + sQ)$. Bob gera $x_b \in \mathbb{Z}_p^*$ e calcula $B_K = x_b(aQ + sQ)$. A chave compartilhada é dada por $K = e(B_K, S_{ID_A})^{x_a} = e(A_k, S_{ID_b})^{x_b}$.

Este esquema possui um par de chaves não tão elegante como no caso do criptossistema de Boneh e Franklin, porque a chave pública não mais depende unicamente da identidade do usuário. O esquema é consistente pois temos que

$$e(B_K, S_{ID_A})^{x_a} = e(Q, Q)^{x_a x_b} = e(A_K, S_{ID_B})^{x_b}.$$

Além disso, a custódia da chave existe porque a TA consegue computar

$$x_a Q = (s + b)^{-1} A_K,$$

$$x_b Q = (s + a)^{-1} B_K,$$

$$K = e(x_a Q, x_b Q).$$

Um esquema parecido pode ser derivado, sem a necessidade de custódia de chave privada. Para isso, é preciso considerar um mapa de distorção ψ, como é possível observar na descrição dada a seguir:

Configuração: Dado k, o PKG escolhe grupos de mapas bilineares, G_1, G_2 e G_t, de ordem prima $p > 2^k$ e geradores $Q \in G_2$, $P = \psi(Q) \in G_1$, $g = e(P, Q) \in G_t$. Escolha um s aleatório pertencente a Z_p^*, uma chave pública do sistema $Q_{pub} = sQ \in G_2$, funções de resumo criptográfico $H_1 : \{0,1\}^* \leftarrow G_1$ e $H_2 : \{0,1\}^* \leftarrow G_2$, $H_2 : G_t \leftarrow \{0,1\}$.

Geração da chave privada: Para uma identidade ID e $a \in \mathbb{Z}_p^*$ a chave pública é dada por $(a + s)Q$, que pode ser calculada por $aQ + sQ$. A chave privada é $S_{ID} = (a + s)^{-1} P$.

Acordo de chaves: Suponha que Alice e Bob tenha identidade representada por ID_A e ID_B respectivamente. Alice gera $x_A \in \mathbb{Z}_p^*$ e calcula $A_K = x_a(bQ + sQ)$. Bob gera $x_b \in \mathbb{Z}_p^*$ e calcula $B_K = x_b(aQ + sQ)$. A chave compartilhada é dada por $K = e(B_K, S_{ID_A})^{x_a} = e(A_k, S_{ID_b})^{x_b}$.

O usuário pode comprovar que a TA gerou os parâmetros de forma consistente e ainda tem certeza que a TA não consegue computar a chave compartilhada de sessão derivada a partir desse esquema. Além disso é possível estabelecer um esquema similar, onde o acordo de chaves pode ser realizado mesmo entre membros de hierarquias distintas, sem custo extra.

4.3 Criptografia sem Certificados

Com o surgimento da criptografia sem certificados (CSC), é possível resolver o problema da custódia da chave privada em sistemas de criptografia baseada em identidades. De fato, CSC representa uma alternativa intermediária, que resolve problemas existentes em CBI e não possui um custo tão caro de gerenciamento como em certificação digital. Infelizmente, CSC ainda possui problemas a serem resolvidos, principalmente com relação a revogação de chave, que implica na troca da identidade do usuário e também o problema da distribuição de chaves públicas, pois é preciso evitar que um atacante publique uma chave falsa. Com isso, é possível afirmar que a criptografia sem certificados ainda é recente

demais para ser usada na prática e ainda existem questões a serem definidas para que isto seja possível.

4.3.1 Definição

A definição de esquemas de criptografia sem certificados foi introduzida por Al-Riyami e Paterson [AR04, ARP03]. De forma geral, a criptografia sem certificados pode ser definida através de sete algoritmos, descritos a seguir:

Configuração: Através deste algoritmo são determinados parâmetros como a curva elíptica e o corpo finito subjacente, as funções de resumo criptográfico necessárias para mapear por exemplo identidades em pontos e principalmente a chave mestra s, que é responsável pela segurança do sistema. Além disso, é gerada uma chave pública do sistema.

Extração da chave privada parcial: Este algoritmo é executado pela TA com o objetivo de gerar uma chave privada parcial, com a qual o usuário pode computar o seu próprio par de chaves.

Estabelecimento de valor secreto: Este algoritmo é executado pelo usuário e utiliza a sua identidade, a chave pública do sistema e um valor aleatório para determinar um valor secreto.

Estabelecimento de chave privada: Este algoritmo utiliza a chave privada parcial, a chave pública do sistema e o valor secreto determinado pelo algoritmo anterior para computar a sua chave privada.

Estabelecimento de chave pública: Este algoritmo utiliza como entrada a chave pública do sistema e o seu valor secreto para determinar uma chave pública que pode ser distribuída livremente.

Ciframento ou Assinatura: Este algoritmo recebe como entrada a chave pública do sistema, uma identidade, a chave pública ou privada referente a esta identidade e uma mensagem, que poderá ser cifrada ou assinada (no caso de cifrassinatura são necessárias duas identidades e as respectivas chaves).

Deciframento ou Verificação: Este algoritmo precisa ser capaz de decifrar ou verificar uma mensagem enviada através do algoritmo anterior, sendo necessário para isso a chave pública do sistema, as identidades envolvidas e as suas respectivas chaves, levando em consideração que para decifrar é necessária a chave privada do usuário, enquanto que para verificar é necessária a sua chave pública.

Em criptografia sem certificados a chave privada do usuário é composta de uma parte gerada pela autoridade de confiança (TA - *trust authority*) e uma parte gerada pelo próprio usuário. Com isso, o problema da custódia da chave em CBI é resolvido, enquanto nenhuma certificação é necessária.

4.3.2 Assinatura agregada sem certificados

Para ilustrar um exemplo de protocolo de criptografia sem certificados será apresentado um esquema de assinatura agregada eficiente e com demonstração de segurança [CD07]. Assinaturas agregadas [BGSL03] são uma generalização de multiassinaturas, permitindo assinaturas múltiplas em mensagens arbitrárias e com o tamanho de uma assinatura simples. Neste esquema de assinatura agregada, Rafael Castro e Ricardo Dahab apresentam um novo serviço para o cenário de criptografia sem certificados.

A forma mais geral de assinaturas agregadas permite que múltiplos usuários assinem múltiplas mensagens, ou seja, dados os n usuários $(U_1, U_2, ..., U_n)$ e as n mensagens $(M_1, M_2, ..., M_n)$, é possível computar γ, tal que γ pode ser usado para verificar que o usuário U_i assinou a mensagem M_i.

Deste modo, em criptografia sem certificados, além dos sete algoritmos citados em 4.3.1, são necessários outros dois algoritmos, descritos a seguir:

Agregamento: Pode ser executado por qualquer usuário. Recebe como entrada a lista $\delta = (\delta_1, \delta_2, ..., \delta_n)$ de assinaturas, onde δ_i é a assinatura de M_i pelo usuário U_i. A saída do algoritmo é a assinatura agregada γ.

Verificação de assinatura agregada: Recebe como entrada γ, uma lista $\mathbb{U}$ de usuários e $\mathbb{M}$ de mensagens. O algoritmo aceita γ se e somente se γ foi gerado pelo conjunto correto de assinaturas individuais $\delta = (\delta_1, \delta_2, ..., \delta_n)$.

A segurança de assinaturas sem certificados é analisada segundo o modelo de falsificação existencial em ataques adaptativos de texto em claro escolhido. Como não há certificação da chave pública, é preciso considerar atacantes com capacidade de publicar chaves públicas falsas. Por outro lado, é preciso considerar a possibilidade do atacante ter acesso à chave mestra s. Assim, devem ser analisados dois tipos de adversários, descritos a seguir

- **Tipo I:** O adversário pode falsificar chaves públicas mas não tem acesso à chave mestra s.
- **Tipo II:** O adversário tem acesso à chave mestra s mas não é capaz de falsificar chaves públicas.

Com isso, a segurança do esquema é baseada em dois jogos, considerando adversários A_I e A_{II}, no modelo do oráculo aleatório, dando acesso ao adversário aos seguintes oráculos:

- `CriarUsuario;`
- `RevelarChaveParcial;`
- `RevelarValorSecreto;`
- `RevelarChavePublica;`
- `Assinar;`
- `SolicitarResumo;`
- `TrocarChavePublica.`

O adversário não pode usar um oráculo para ter informações sobre o usuário que será atacado, portanto, no contexto de assinaturas agregadas, o adversário não tem a permissão de solicitar informações secretas sobre algum dos usuários pertencentes à lista $(U_1, U_2, ..., U_n)$. Assim, é possível definir os seguintes jogos:

Definição 4.3.1. Jogo 1: Seja C o algoritmo desafiante e k o parâmetro de segurança. Com isso, o jogo 1 é dado pelas seguintes etapas:

1. C gera seu par de chaves, usando os algoritmos descritos em 4.3.1 e o parâmetro k.

2. C roda A_I, onde A_I tem acesso aos oráculos:
   - `RevelarChavePublica;`
   - `RevelarChaveParcial;`
   - `RevelarValorSecreto;`
   - `TrocarChavePublica;`
   - `SolicitarResumo;`
   - `Assinar.`
3. A_I retorna (ID, M, δ)

A_I ganha se a assinatura δ for válida para a mensagem M e a identidade ID e as condições seguintes forem válidas:

1. O oráculo `Assinar` nunca foi utilizado para assinar a mensagem M com a identidade ID;
2. O oráculo `RevelarChaveParcial` nunca foi utilizado com a identidade ID.

Definição 4.3.2. Jogo 2: Seja C o algoritmo desafiante e k o parâmetro de segurança. Com isso, o jogo 2 é dado pelas seguintes etapas:

1. C gera seu par de chaves, usando os algoritmos descritos em 4.3.1 e o parâmetro k.

2. C roda A_{II}, onde A_{II} tem acesso aos oráculos:

- `RevelarChavePublica`;
- `RevelarChaveParcial`;
- `RevelarValorSecreto`;
- `TrocarChavePublica`;
- `SolicitarResumo`;
- `Assinar`.

3. A_I retorna (ID, M, δ)

A_I ganha se a assinatura δ for válida para a mensagem M e a identidade ID e as condições seguintes forem válidas:

1. O oráculo `Assinar` nunca foi utilizado para assinar a mensagem M com a identidade ID;
2. O oráculo `RevelarChaveParcial` nunca foi utilizado com a identidade ID;
3. O oráculo `TrocarChavePublica` nunca foi utilizado com a identidade ID.

Para o caso de assinaturas agregadas estes jogos precisam apenas de pequenas alterações. A primeira alteração é o retorno do adversário no passo 3, que deve ser uma assinatura agregada dada por γ. A segunda alteração é com relação ao algoritmo de verificação que precisa verificar a assinatura agregada para cada par de mensagem e usuário (M_i, U_i).

Configuração: Dado k, o PKG escolhe grupos de mapas bilineares, G_1, G_2 e G_t, de ordem prima $p > 2^k$ e geradores $Q \in G_2$, $P = \psi(Q) \in G_1$, $g = e(P, Q) \in G_t$. Escolha um s aleatório pertencente a Z_p^*, uma chave pública do sistema $Q_{pub} = sQ \in G_2$, funções de resumo criptográfico H_1, H_2 e H_3.

Extração de chave privada parcial: Este algoritmo é executado pela TA, que deve escolher um r aleatório pertencente a $\mathbb{Z}_p^*$. A TA computa $R = rP$ e retorna $d = (r + sH_1(ID, R))$ e o valor R.

Estabelecimento de valor secreto: Este algoritmo é executado pelo usuário, que escolhe um x aleatoriamente em $\mathbb{Z}_p^*$ e x é o seu valor secreto.

Estabelecimento de chave privada: A chave privada do usuário é dada pelo par (d, x), onde d foi computada pela TA e x é o valor secreto escolhido aleatoriamente pelo usuário.

Estabelecimento de chave pública: O usuário computa $P_{ID} = xP$. A pública do usuário é dada pelo par (R, P_{ID}).

Assinatura: Dada uma mensagem M, a identidade ID e a chave privada (d, x), a assinatura é dada por $\delta = dH_2(M, ID, P_{ID}, R) + xH_3(M, ID, P_{ID}, R)$.

Verificação: Dada a assinatura δ, a mensagem M, a identidade ID com sua chave pública (R, P_{ID}), calcula-se $h_1 = H_1(ID, R)$, $h_2 = H_2(M, ID, P_{ID}, R)$ e $h_3 = H_3(M, ID, P_{ID}, R)$. A assinatura δ é aceita se e somente se

$$e(P, \delta) = e(h_2, R + h_1 Q_{pub}) e(h_3, P_{ID}).$$

Note que a chave pública é composta de uma parte gerada pela TA e uma parte gerada pelo usuário. Um adversário capaz de falsificar chaves públicas é capaz de falsificar ambos os componentes.

Para utilizar este esquema para assinaturas agregadas, basta adicionar os algoritmos descritos a seguir:

Agregamento: Dado o conjunto $\delta = \{\delta_1, \delta_2, ..., \delta_3\}$ de assinaturas a serem agregadas, retorne

$$\gamma = \sum_{\forall i} \delta_i.$$

Verificação de assinatura agregada: Seja um conjunto de mensagens $\mathbb{M} = \{M_1, M_2, ..., M_n\}$ e um conjunto de usuários $\mathbb{U} = \{u_1, u_2, ..., u_n\}$. Seja u_i um usuário pertencente a $\mathbb{U}$, então sua identidade é dada por ID_{u_i}, sua chave pública é dada por $(R_{u_i}, P_{ID_{u_i}})$. Seja $h_{u_i} = H_1(ID_{u_i}, R_{u_i})$. Compute os valores

$$\gamma_1 = \prod_{1 \leq i \leq n} e(\sum H_2(M_i, ID_{u_i}, P_{ID_{u_i}}, R_{u_i}), R_{u_i} + h_{u_i} Q_{pub}).$$
$$\gamma_2 = \prod_{1 \leq i \leq n} e(\sum H_3(M_i, ID_{u_i}, P_{ID_{u_i}}, R_{u_i}), P_{ID_{u_i}}).$$

Aceite a assinatura agregada se e somente se

$$e(P, \gamma) = \gamma_1 \gamma_2.$$

Para otimizar o algoritmo de verificação é possível reunir mensagens assinadas por um mesmo usuário e usar a bilinearidade do emparelhamento para que a verificação seja feita utilizando apenas um emparelhamento por usuário e não mais um emparelhamento por mensagem. Para isso, basta somar o resultado dos resumos criptográficos aplicados às mensagens assinadas pelo usuário e calcular o emparelhamento com o valor obtido.

A demonstração de segurança deste esquema pode ser encontrada com detalhes em [CD07] e considera adversários do tipo I e II. A demonstração é realizada apresentando

um algoritmo C, tal que se um adversário consegue vantagem no respectivo jogo, definido em 4.3.1 ou 4.3.2, então o algoritmo C é capaz de resolver o problema PCDHCE. Como assume-se que o PCDHCE é difícil, então conclui-se que não existe um adversário capaz de atacar o esquema.

4.4 Cifrassinatura

O conceito de cifrassinatura foi proposto pela primeira vez por Zheng [Zhe97]. Através desta primitiva criptográfica é possível prover confidencialidade, autenticidade e não-repúdio em um único passo, permitindo maior eficiência da operação e também maior segurança. Como em criptografia convencional, a recuperação de um texto claro a partir de um texto cifrado deve ser computacionalmente inviável sem a chave privada do destinatário. Além disso, cifrassinar um texto sem a chave privada do remetente deve ser também computacionalmente inviável.

4.4.1 Cifrassinatura Baseada em Identidades

Existem muitas propostas de protocolos de cifrassinatura baseada em identidades, como por exemplo [Boy03, CML05, CYHC03, CYH+05, LQ03a, YW03], que possuem demonstração de segurança segundo o modelo do oráculo aleatório. Dentre eles, a proposta de Chen e Malone-Lee [CML05] é a mais eficiente.

Barreto [BLMQ05] propôs em 2005 um protocolo de cifrassinatura baseada em identidades. Sua proposta tem grande importância porque utiliza apenas o cálculo de dois emparelhamentos bilineares na verificação. Mas além de ser uma alternativa eficiente, o protocolo possui demonstração de segurança mais forte que as outras propostas. Segue abaixo a descrição com maiores detalhes do protocolo escolhido como foco deste projeto.

Configuração: Dado k, o PKG escolhe grupos de mapas bilineares, G_1, G_2 e G_t, de ordem prima $p > 2^k$ e geradores $Q \in G_2$, $P = \psi(Q) \in G_1$, $g = e(P, Q) \in G_t$. Escolha um s aleatório pertencente a Z_p^*, uma chave pública do sistema $Q_{pub} = sQ \in G_2$, funções de resumo criptográfico H_1, H_2 e H_3.

Geração de Par de Chaves: Para uma identidade ID, a chave privada é $S_{ID} = \frac{1}{H_1(ID)+s}Q \in G_2$.

Cifrassinatura: Dada uma mensagem M, a identidade do remetente ID_r e a identidade do destinatário ID_d, utiliza-se x aleatório pertencente a Z_p^* para calcular $r = g^x$, $c = M \oplus H_3(r)$ e $h = H_2(M, r)$. Calcula-se $S = (x+h)\psi(S_{ID_d})$ e $T = x(H_1(ID_r)P + \psi(Q_{pub}))$. A cifrassinatura é a tripla (c, S, T).

Deciframento e Verificação: Dada a tripla (c, S, T) e a identidade do remetente ID_r, calcula-se $r = e(T, S_{ID_d})$, $M = c \oplus H_3(r)$ e $h = H_2(M, r)$. Aceite a mensagem se $r = e(S, H - 1(ID_r)Q + Q_{pub})g^{-h}$, neste caso a mensagem M e a assinatura (h, S) são retornadas.

Barreto [BLMQ05] demonstrou que este esquema de cifrassinatura é semanticamente seguro, não estando sujeito aos ataques que ocorrem quando são utilizadas algumas otimizações dos emparelhamentos de Weil e Tate. Este esquema mostra detalhadamente como o uso de emparelhamentos permite a construção de um protocolo de criptografia baseada em identidades.

Para prover confidencialidade, autenticidade e não-repúdio, esquemas de cifrassinatura devem resistir a dois tipos de ataques:

- confidencialidade de mensagens em ataques adaptativos de texto cifrado escolhido;
- falsificação existencial em ataques adaptativos de texto em claro escolhido.

Para demonstrar a segurança do esquema de cifrassinatura é preciso considerar dois jogos G_1 e G_2, definidos a seguir:

Definição 4.4.1. Jogo G_1: Um esquema de cifrassinatura baseada em identidades satisfaz a propriedade de confidencialidade de mensagens em ataques de texto cifrado escolhido se nenhum adversário de tempo polinomial for capaz de obter vantagem não negligível no seguinte jogo, onde C é o algoritmo desafiante e k o parâmetro de segurança.

1. C executa a configuração do esquema de cifrassinatura baseada em identidades e gera o parâmetro k.
2. C roda o algoritmo A, onde A tem acesso aos oráculos:
   - `GerarChavePrivada;`
   - `Cifrassinar;`
   - `Decifrassinar;`
3. A produz dois textos claros M_0 e M_1 e identidades ID_S e ID_R e recebe $C =$ `Signcrypt`(M_b, S_{ID_S}, ID_R), onde b é escolhido aleatoriamente no conjunto $\{0, 1\}$;
4. Nesta fase A pode usar novamente os mesmos oráculos do passo 2;
5. Finalmente, A retorna um bit b' e vence o jogo se e somente se $b' = b$.

Durante o jogo, as seguintes condições precisam ser válidas:

1. O oráculo `GerarChavePrivada` nunca foi utilizado com a identidade ID_R;

2. O oráculo `Decifrassinar` nunca foi utilizado com a identidade ID_R e a cifrassinatura C.

Definição 4.4.2. Jogo G_2: Um esquema de cifrassinatura baseada em identidades deve resistir falsificação existencial em ataques de texto claro escolhido se nenhum adversário de tempo polinomial for capaz de obter vantagem não negligível no seguinte jogo, onde C é o algoritmo desafiante e k o parâmetro de segurança.

1. C executa a configuração do esquema de cifrassinatura baseada em identidades e o parâmetro k.
2. C roda o algoritmo A, onde A tem acesso aos oráculos:
   - `GerarChavePrivada;`
   - `Cifrassinar;`
   - `Decifrassinar;`
3. A retorna (ID, M, δ)

A ganha se a assinatura δ for válida para a mensagem M e a identidade ID e as condições seguintes forem válidas:

1. O oráculo `Cifrassinar` nunca foi utilizado para assinar a mensagem M com a identidade ID;
2. O oráculo `GerarChavePrivada` nunca foi utilizado com a identidade ID.

Em ambas as definições são considerados ataques internos. De fato, na confidencialidade de mensagens, o adversário pode ser desafiado com uma cifrassinatura gerada usando uma chave privada de um remetente corrompido. Além disso, no contexto de não-repúdio, o atacante pode gerar cifrassinaturas usando uma chave privada de uma destinatário corrompido.

Capítulo 5

Implementação

5.1 Introdução

Com o intuito de comprovar a relevância do custo de emparelhamentos bilineares nos protocolos estudados, usando como base a bibilioteca Miracl, foram realizadas implementações dos procotolos de cifrassinatura baseada em identidades, e também cifra-mento+assinatura baseada em identidades. No decorrer do processo de implementação, um colega do Laboratório de Criptografia Aplicada, Diego Aranha, sugeriu um protocolo para cifrassinatura sem certificados. Com esta proposta em mãos, Rafael Castro e eu sugerimos algumas alterações até chegar na proposta de cifrassinatura sem certificados apresentada neste capítulo. A implementação desta proposta também foi realizada, permitindo a medição de tempo dos protocolos, podendo verificar que o emparelhamento representa a maior parcela de custo. A medição foi realizada em um computador Pentium Centrino 1.70 MHz, com 512 Mb de memória RAM.

5.2 Escolha de parâmetros

Existem diversos fatores a serem considerados na hora de projetar um criptossisstema baseado em curvas elípticas. A escolha de parâmetros deve levar em conta não só a segurança contra ataques de cálculo de índices, como também ataques ao corpo finito resultante da transferência do problema do logaritmo discreto em curvas elípticas através de emparelhamentos bilineares ou outras idéias como por exemplo usando espaços vetorais sobre $\mathbb{F}_q$ ou então pela técnica da descida de Weil. Por outro lado, existem circunstâncias que impõe restrições à escolha de parâmetros, seja por limitações de recursos computacionais, como em arquiteturas embarcadas de pouco poder de processamento e principalmente com pouco poder de armazenamento, seja por peculiaridades da plataforma que per-

mitem otimizações específicas para uma determinada escolha de parâmetros, como por exemplo instruções especiais para manipulação de polinômios; ou mesmo por mecanismos de proteção contra ataques de canal lateral, ou qualquer outro mecanismo que permita o uso de parâmetros que em outros sistemas não poderiam ser selecionados.

Existem diversas estruturas algébricas que podem ser usadas juntamente com o problema do logaritmo discreto para projetar criptossistemas, desde corpos finitos até corpos de funções. Como foi apresentado anteriormente, o PLD em corpos finitos pode ser resolvido em tempo subexponencial. Outro caso em que temos algoritmos subexponenciais para resolver o PLD são os corpos numéricos, isto é, extensões algébricas dos racionais, como por exemplo o corpos constituídos de elementos da forma $a + b\sqrt{2}$, para $a, b \in \mathbb{Q}$.

Assim, pode-se usar variedades algébricas, onde não são conhecidos ataques por cálculo de índices em tempo subexponencial. Portanto, podemos escolher uma curva C de genus g, sobre um corpo finito $\mathbb{F}_q$, onde $q = p^d$. O grupo de Picard Pic^0_C é canonicamente isomorfo à variedade Jacobiana J_C de C, que é uma variedade abeliana. Além disso, existe um divisor da forma $\sum_{i=1}^{r} P_i - rP_\infty$, para $r \leq g$, que representa um elemento de J_C. Em curvas elípticas, que são variedades algébricas de genus 1, temos que para cada ponto $P \in E$, existe um divisor $D \sim |P| - |\infty|$.

Mas existem ataques a curvas de genus $g = 3$ e 4, capazes de resolver o problema do logaritmo discreto em Pic^0_C respectivamente em complexidade $O(|\,\mathrm{Pic}^0_C\,|^{0.375})$ e $O(|\,\mathrm{Pic}^0_C\,|^{0.44})$. De forma geral, são utilizadas apenas curvas hiperelípticas de genus $g = 1, 2$ e 3 e mesmo o caso $g = 3$ precisa levar em conta o tamanho do grupo para não sofrer ataques de cálculo de índices

Além disso, precisamos tomar cuidado para evitar a transferência do problema do logaritmo discreto. Ou seja, é preciso garantir as seguintes condições:

1. o grau de imersão k grande o suficiente para que o PLD em $\mathbb{F}_{q^k}$ seja difícil de resolver. Com isso, evita-se ataques através de emparelhamentos bilineares;

2. se o corpo finito subjacente é $\mathbb{F}_{p^d}$, então deve-se evitar que d tenha um divisor d_0, não permitindo o uso do método da descida de Weil. Ou seja, deve-se usar corpos primos, ou então corpos binários e ternários de grau d, que não seja um primo de Mersenne ou de Fermat.

Além de tudo, os parâmetros devem ser escolhidos de modo que as operações necessárias para executar os protocolos sejam eficientes e exijam pouco espaço de armazenamento.

5.2.1 Escolha do corpo finito

Devido a eficiência da aritmética, geralmente são utilizados corpos primos $\mathbb{F}_p$, ou corpos binários $\mathbb{F}_{2^d}$. Mas existem casos em que corpos de extensão podem oferecer certas vantagens. Também existem muitos estudos realizados para corpos ternários.

No caso de corpos primos, geralmente são escolhido primos p com pequeno peso de Hamming, ou seja, $p = 2^k + c$, onde c é pequeno, permitindo calcular reduções modulares eficientemente. A multiplicação pode ser feita usando o método de Karatsuba. A representação Montgomery também pode representar um ganho de eficiência.

Para corpos binários $\mathbb{F}_{2^d}$, é importante que d seja um primo grande. O uso de bases normais convém em casos em que são necessárias mais operações de quadrado que multiplicação, ou então em casos em que é necessário calcular raíz quadrada, pois estas operações são aplicações do mapa de Fröbenius, que resumem-se a deslocamentos circulares.

5.2.2 Escolha da curva elíptica

A escolha da curva elíptica deve levar em conta a eficiência das principais operações, como a multiplicação por escalar e portanto a soma e a duplicação. Para isso, sabendo-se previamente qual é o corpo finito subjacente, é possível escolher o sistema de coordenadas afins, ou então o sistema de coordenadas projetivas, que podem ter pesos não triviais, como por exemplo em coordenadas jacobianas e López-Dahab. Outro fato a ser considerado é se a curva elíptica em questão é supersingular ou não-supersingular. Em curvas supersingulares, temos sempre um grau de imersão pequeno, de forma que é preciso tomar cuidado para não ser possível transferir o problema do logaritmo discreto através de emparelhamentos bilineares.

Existem também métodos para gerar curvas aleatórias, como por exemplo o método da *multiplicação complexa* . Uma grande quantidade de curvas elípticas estão descritas detalhadamente nos padrões NIST, SEC-G, FIPS186-2, etc. Estes padrões apresentam todos os parâmetros necessários para implementar um criptossistema de forma segura. Com isso, é possível analisar os requisitos de segurança necessários e os recursos computacionais disponíveis para a melhor escolha de parâmetros.

5.3 Proposta de cifrassinatura sem certificados

De acordo com as seções anteriores, as vantagens do uso de cifrassinatura como primitiva criptográfica e as soluções encontradas no modelo de criptografia sem certificados motivam a busca por um protocolo eficiente e seguro para construção de uma proposta

para cifrassinatura sem certificados. Esta é uma área ainda pouco explorada, onde existem poucas propostas, como por exemplo em [WZM07], que é uma proposta específica para o contexto de cifrassinatura em anel para comunicação anônima e privada. Assim, propomos nesta seção, um protocolo de cifrassinatura sem certificados para um cenário genérico.

Segue abaixo a descrição detalhada do protocolo proposto neste trabalho:

Configuração: Dado k, o PKG escolhe grupos de mapas bilineares, G_1, G_2 e G_t, de ordem prima $p > 2^k$ e geradores $Q \in G_2$, $P = \psi(Q) \in G_1$, $g = e(P, Q) \in G_t$. Escolha um s aleatório pertencente a Z_p^*, uma chave pública do sistema $Q_{pub} = sQ \in G_2$, funções de resumo criptográfico H_1, H_2, H_3 e H_4.

Extração de chave privada parcial: Dada uma identidade ID, computar o valor $D_{ID} = (H_1(ID) + s)^{-1}P$.

Extração de par de chaves: Dada uma identidade ID, selecionar aleatoriamente $x_{ID} \in \mathbb{Z}_p^*$. Computar a chave privada $S_{ID} = (x_{ID})^{-1}D_{ID}$. Computar a chave pública $P_{ID} = x_{ID}(H_1(ID)P + P_0)$. O par de chaves de ID é dado por (S_{ID}, P_{ID}).

Cifrassinatura: Dada uma mensagem M, a identidade do remetente ID_r e a identidade do destinatário ID_d, utiliza-se d aleatório pertencente a Z_p^* para calcular $r = H_3(d, M)$, $c = M \oplus H_2(rP_A, g^{r^{-1}})$ e $h = H_4(g^{r^{-1}}, rP_A, r^{-1}P_B)$. Calcula-se $T = (r + h)^{-1}S_A$ e a cifrassinatura é dada por $S = (c, rP_A, r^{-1}P_B, T)$.

Deciframento e Verificação: Dada a cifrassinatura (c, R, S, T) e a identidade do remetente ID_r, calcula-se $r' = e(S, S_B)$, $h = H_4(r', R, S)$, $V = e(R + hP_A, T)$, $M = c \oplus H_2(R, r')$. Aceite a mensagem se $V = g$, neste caso a mensagem M é retornada, caso contrário a assinatura é recusada.

A descrição deste protocolo utiliza apenas cinco algoritmos, pois na prática são estes algoritmos que são implementados. Conceitualmente, é possível distinguir sete algoritmos, como descrito na seção 4.3.1.

5.4 Parâmetros

A escolha de parâmetros foi realizada tendo em vista a dificuldade do problema do logaritmo discreto na curva elíptica subjacente, assim como na dificuldade do problema do logaritmo discreto no corpo de extensão. Assim, de acordo com esses critérios, foi possível escolher dois conjuntos de parâmetros, cuja equação da curva elíptica é comum a ambos e é dada por $y^2 + y = x^3 + x + 1$, com cofator $h = 1$ e grau de mergulho $k = 4$.

A diferença está na escolha do corpo finito subjacente. O primeiro utiliza o corpo finito $\mathbb{F}_{2^241}$ [GHS02a], equivalente a segurança de 964 bits no RSA, com polinômio irredutível dado por $f(x) = x^{241} + x^{70} + 1$, enquanto o segundo utiliza o corpo finito $\mathbb{F}_{2^457}$ [BBF+06], com polinômio irredutível dado por $f(x) = x^{457} + x^{16} + 1$, equivalente a segurança de 1828 bits no RSA.

5.5 Implementação

Para comprovar a importância da proposta de cifrassinatura sem certificados, foram implementados sete protocolos, enumerados a seguir:

1. cifrassinatura baseada em identidades [BLMQ05];
2. ciframento baseado em identidades [BF03];
3. assinatura baseada em identidades [BLMQ05];
4. ciframento sem certificados com demonstração de segurança [LQ06];
5. assinatura sem certificados com demonstração de segurança [CD07];
6. assinatura sem certificados sem demonstração de segurança [CPHL07];
7. cifrassinatura sem certificados sem demonstração de segurança, proposto neste trabalho.

O protocolo de ciframento sem certificados proposto em [LQ06] é eficiente em relação a outros protocolos encontrados na literatura e possui demonstração de segurança, portanto foi utilizado tanto para o caso em que a segurança demonstrável era necessária, quanto no caso contrário.

Estes protocolos foram implementados sobre a biblioteca *miracl*, utilizando os parâmetros descritos na seção anterior.

5.5.1 Funções de Resumo Criptográfico

Para a implementação de alguns protocolos foi necessário utilizar uma função de resumo criptográfico, que mapea identidades em pontos da curva elíptica. Na biblioteca *miracl* existe uma versão desta função para o caso de curvas elípticas sobre corpos primos. Esta função utiliza a identidade para gerar um número $x = H(ID)$, tal que $x \in \mathbb{F}_p$. Então o algoritmo entra em um laço que utiliza x para gerar uma seqüência determinística, até que seja possível encontrar um y tal que (x, y) seja um ponto da curva elíptica. Esta função foi adaptada para o caso de curvas supersingulares sobre corpos binários.

5.6 Resultados

Nesta seção serão apresentados os tempos dos protocolos implementados, comprovando a eficiência do protocolo de cifrassinatura baseada em identidades (CBI) escolhido e do protocolo de cifrassinatura sem certificados (CSC) proposto. É importante lembrar que estão sendo considerados os tempos totais dos protocolos, ou seja, o tempo para cifrassinar e decifrassinar uma mensagem, dadas as identidades e chaves necessárias, além do ciframento, deciframento, assinatura e verificação, utilizados para analisar o ganho da cifrassinatura perante a abordagem padrão.

Nesta primeira tabela, o protocolo de cifrassinatura baseada em identidades é de Barreto [BLMQ05], enquanto o protocolo de ciframento baseado em identidades é uma extensão do protocolo de Boneh e Franklin [BF03] e o protocolo de assinatura baseada em identidades é de Barreto [BLMQ05]. Com isso, a última coluna corresponde a soma dos tempos dos protocolos de ciframento e assinatura citados.

Tabela 5.1: **Cifrassinatura Baseada em Identidades**

Parâmetros	Cifrassinatura	Ciframento + Assinatura
241 bits	117,6 ms	172,8 ms
457 bits	151,4 ms	263,2 ms

Para analisar a eficiência da nossa proposta de cifrassinatura sem certificados serão utilizadas duas tabelas, apresentando o tempo de ciframento e assinatura sem certificados com demonstração de segurança e sem demonstração de segurança. É importante ressaltar que a demonstração de segurança da nossa proposta não foi concluída.

O protocolo de cifrassinatura está descrito na seção 4.4.1. Já o protocolo de ciframento sem certificados é descrito em [LQ06] e o protocolo de assinatura sem certificados é descrito em [CD07].

Tabela 5.2: **Cifrassinatura sem Certificados com demonstração de segurança**

Parâmetros	Cifrassinatura	Ciframento + Assinatura
241 bits	123,9 ms	225,9 ms
457 bits	159,0 ms	281,7 ms

Nesta segunda tabela, o objetivo é comparar nosso protocolo de cifrassinatura sem certificados com o ciframento seguido da assinatura, no contexto de criptografia sem certificados, mas sem exigir demonstração de segurança, ou seja, permitindo protocolos mais

eficientes. O protocolo de cifrassinatura portanto é o mesmo da tabela anterior, assim como o protocolo de ciframento sem certificados, já que esta proposta é eficiente. Já a assinatura sem certificados é uma proposta diferente, que utiliza um único emparelhamento no processo de verificação da assinatura, descrito em [CPHL07].

Tabela 5.3: **Cifrassinatura sem Certificados sem demonstração de segurança**

Parâmetros	Cifrassinatura	Ciframento + Assinatura
241 bits	123,9 ms	125,1 ms
457 bits	159,0 ms	167,2 ms

Capítulo 6

Conclusão

Neste trabalho foi proposto um protocolo eficiente de cifrassinatura sem certificados. A eficiência foi comprovada através da comparação entre o tempo de execução de propostas equivalentes e o protocolo proposto no capítulo anterior. É importante ressaltar que o principal custo no tempo de execução são os emparelhamentos bilineares. De fato, cada emparelhamento leva 51,2 ms para 241 bits e 71,3 ms para 457 bits. Desta forma, conclúi-se que o número de emparelhamentos bilineares é determinante no tempo de execução dos protocolos de criptografia baseada em identidades e criptografia sem certificados.

A escolha de parâmetros é parte essencial do processo de implementação, pois estabelece um ponto de equilíbrio no compromisso entre segurança e eficiência. Curvas supersingulares foram escolhidas porque não foi possível comprovar até agora que curvas elípticas supersingulares não sejam seguras. O único cuidado necessário é o de utilizar parâmentros tais que o problema do logaritmo discreto na curva e no corpo de extensão sejam difíceis, ou seja, a curva deve ter ordem superior a 160 bits e o corpo de extensão deve ter ordem superior a 512 bits.

O modelo de criptografia baseada em identidades tem vantagens interessantes quando comparado ao modelo de certificação digital, mas existem problemas como a custódia da chave. Este problema pode ser resolvido pelo modelo de criptografia sem certificados de forma eficiente. Mas ainda existem questões abertas que precisam ser resolvidas para tornar a criptografia sem certificados um modelo adequado para ser usado na prática, substituindo a certificação digital. Dentre esses problemas estão a revogação da chave de um usuário, que acarreta na necessidades de trocar a sua identidade, o que obviamente não é desejável. Além disso, é preciso um mecanismo para evitar a publicação falsa de chaves públicas.

A demonstração de segurança de protocolos criptográficos ainda é um assunto relativamente novo e precisa estabelecer métodos e modelos que representem melhor o comportamento das operações criptográficas e a capacidade de atuação do atacante, pois ainda

não se sabe até que ponto é necessário abrir mão de eficiência para obter um protocolo mais seguro, porque não é possível saber se o uso de modelos mais restritos e rígidos realmente levam a um ganho de segurança na prática.

Neste trabalho foram estudados os conceitos necessários para a construção de emparelhamentos bilineares, de forma a atingir o estado da arte no assunto, permitindo propor contribuições mais sólidas e relevantes. Com isso, os estudos realizados durante a construção desta tese servem como base para o doutorado.

6.1 Trabalhos Futuros

Como trabalho futuro será realizado um estudo mais aprofundado das técnicas de demonstração de segurança, principalmente no modelo do oráculo aleatório, na tentativa de provar a segurança da nossa proposta, já que esta demonstração representaria um resultado importante para criptografia sem certificados.

Referências Bibliográficas

[AR04] S. Al-Riyami. Cryptographic schemes based on elliptic curve pairings. 2004.

[ARP03] S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. 2894 of Lecture Notes in Computer Science:452–473, 2003.

[BBF+06] G. Bertoni, L. Breveglieri, P. Fragneto, G. Pelosi, and L. Sportiello. Software implementation of tate pairing over gf(2^m). In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 7–11, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[Ber04] D. J. Bernstein. Scaled remainder trees. *preprint*, 2004.

[BF01] D. Boneh and M. Franklin. Identity based encryption from the weil pairing. In *Advances in Cryptology – CRYPTO 2001: 21st Annual International Cryptology Conference*, LNCS 2139, pages 213–229. Springer-Verlag, 2001.

[BF03] D. Boneh and M. Franklin. Identity based encryption from the weil pairing. In *SIAM J. Comput.*, volume 32, pages 586–615, 2003.

[BGhS04] P. S. L. M. Barreto, Steven Galbraith, Colm Ó hÉigeartaigh, and Michael Scott. Efficient pairing computation on supersingular abelian varieties. In *Cryptology ePrint Arquive*, 2004.

[BGM91] T. Beth, W. Geiselmann, and F. Meyer. Finding (good) normal bases in finite fields. In *ISSAC '91: Proceedings of the 1991 international symposium on Symbolic and algebraic computation*, pages 173–178, New York, NY, USA, 1991. ACM Press.

[BGSL03] D. Boneh, C. Gentry, H. Shachan, and B. Lynn. Aggregate and verifiably encrypted signatures from bilinear maps. LNCS 2656:416–432, 2003.

[BJT97] J. Buchmann, M. J. Jacobson, and E. Teske. On some computacional problems in finite abelian groups. *Math. Comp.*, 66:1663–1687, 1997.

[BK78] R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *J. ACM*, 25(4):581–595, 1978.

[BLMQ05] P. S. L. M. Barreto, Benoît Libert, Noel McCullagh, and Jean Jacques Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In *Advances in Crytology – Asiacrypt'2005*, LNCS. Springer-Verlag, 2005.

[BM05] P. S. L. M Barreto and N. McCullagh. A new two-party identity-based authenticated key agreement. 2005.

[BNN04] M. Bellare, C. Namprempe, and G. Neven. Security proofs for identity-based identification and signature schemes. *Eurocrypt'04*, 3027:268–286, 2004.

[Boy03] X. Boyen. Multipurpose identity-based signcryption: A swiss army knife for identity-based cryptography. *Crypto'03*, 2729:382–399, 2003.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[Bre80] R. P. Brent. An improved monte carlo factorization algorithm. *BIT*, 20:176–184, 1980.

[CD07] Rafael Castro and Ricardo Dahab. Efficient certificateless signatures suitable for aggregation. Cryptology ePrint Archive, Report 2007/454, 2007.

[CF06] H. Cohen and G. Frey. *Handbook of Elliptic and Hyperelliptic curve criptography*. Chapman & Hall/CRC, Boca Raton, 1 edition, 2006.

[CML05] L. Chen and J. Malone-Lee. Improved identity-based signcryption. *PKC'05*, 3386:362–379, 2005.

[Coh00] H. Cohen. *A course in Computational Algebraic Number Theory*. Springer-Verlag, Berlin, 4 edition, 2000.

[Col80] G. E. Collins. Lecture notes on arithmetic algorithms. 1980.

[CP01] J. J. Cannon and C. Playoust. *An introduction to Algebraic Programming with Magma*. Springer-Verlag, February 2001.

[CPHL07] K. Choi, J. Park, J. Hwang, and D. Lee. Efficient certificateless signature schemes. pages 443–458, 2007.

[Cra92] R. Crandall. Method and apparatus for public key exchange in a cryptographic system. volume 632 of *159*, October 1992.

[CYH+05] S. S. M. Chow, T. H. Yuen, L. C. S. Hui, , and S. M. Yiu. Signcryption in hierarchical identity based cryptosystem. IFIP TC11, 2005.

[CYHC03] S. S. M. Chow, S. M. Yiu, L. C. K. Hui, and K. P. Chow. Efficient forward and provably secure id-based signcryption scheme with public verifiability and public ciphertext authenticity. 2971 of LNCS:352–369, 2003.

[DL03] I. Duursma and H.-S. Lee. Tate pairing implemantation for hyperelliptic curves $y^2 = x^p - x + d$. In Springer-Verlag, editor, *ASIACRYPT 2003*, LNCS 2894, pages 111–123, 2003.

[Eis99] D. Eisenbud. *Commutative Algebra toward a view of Algebraic Geometry.* Springer-Verlag, 1999.

[FMR99] G. Frey, M. Müller, and H. G. Rück. The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Trans. on Inform. Theory*, 45(5):1717–1719, 1999.

[GHS02a] S. D. Galbraith, K. Harrison, and D. Soldera. Implementing the tate pairing. In *ANTS-V: Proceedings of the 5th International Symposium on Algorithmic Number Theory*, pages 324–337, London, UK, 2002. Springer-Verlag.

[GHS02b] S. D Galbraith, F. Hess, and N. P. Smart. Extending the ghs weil-descent attack. *Lecture Notes in Comput. Sci.*, 2332:29–44, 2002.

[GS99] S. D Galbraith and N. P. Smart. A cryptographic application of weil descent. *Lecture Notes in Comput. Sci.*, 1746:191–200, 1999.

[Har77] R. Hartshorne. *Algebraic Geometry.* Springer-Verlag, 1977.

[HC02] J. L. Hill and D. E. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, 2002.

[Jeb93] T. Jebelean. Improving the multiprecision euclidean algorithm. volume 722, pages 45–58, Berlin, 1993. Springer-Verlag.

[Jou00] A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Algorithmic Number Theory, IV-Symposium (ANTS IV)*, LNCS 1838, pages 385–394. Springer-Verlag, 2000.

[Kar95] A. A. Karatsuba. The complexiy of computations. *Trudy Mat. Inst. Steklov*, 211:186–202, 1995.

[Knu97] D. E. Knuth. *The art of computer programming - Seminumerical algorihtms*, volume 2. Addison-Wesley Publishing Company, MA, 3 edition, 1997.

[KO63] A. Karatsuba and Y. Ofman. Multiplication of Multidigit Numbers on Automata. *Soviet Physics Doklady*, 7:595–596, January 1963.

[Kwo05] S. Kwon. Efficient Tate pairing computation for elliptic curves over binary fields. In C. Boyd and J. M. González Neto, editors, *ACISP 2005*, pages 134–145, Berlim Heidelberg 2005, 2005. Springer-Verlag.

[KY98] N. Kunihiro and H. Yamamoto. Window and extended window methods for addition chain and addition-subtraction chain. *IEICE Trans. Fundamentals*, E81(1), january 1998.

[Lan03] S. Lang. *Complex Analysis*. Springer-Verlag, 2003.

[LD99] J. López and R. Dahab. Improved algorithms for elliptic curve arithmetic if GF(2^n). In *Selected Areas in Cryptography: 5th Annual International Workshop. SAC'98. Kingston. Ontario, Canada, August 1998. Proceedings.*, LNCS 1556, pages 201–212. Springer-Verlag, 1999.

[Ler97] R. Lercier. *Algorithmique des courbes elliptiques dans les corps finis*. École Polytechnique, 1997.

[LN97] R. Lidl and H. Niederreiter. *Finites Fields*. Cambridge, The Pitt Building, Trumpington Street, Cambridge, Unigted Kingdom, 2 edition, 1997.

[LQ03a] B. Libert and J.-J. Quisquater. New identity based signcryption schemes from pairings. In *IEEE Information Theory Workshop*, 2003.

[LQ03b] B. Linbert and J.-J. Quisquater. New identity based signcryption schemes from pairings. *IEEE Information Theory Workshop*, 2003.

[LQ06] B. Libert and J.-J. Quisquater. On constructing certificateless cryptosystems from identity based encryption. In *Public Key Cryptography 2006 (PKC'06)*, LNCS 3958, pages 474–490, 2006.

[Mao04] W. Mao. *Modern Cryptography - theory and practice*. Prentice Hall, 2004.

[MB72] R. Moenck and A. Borodin. Fast modular transforms via division. In *FOCS*, pages 90–96, 1972.

[Mil86] V. S. Miller. Short programs for functions on curves. *IBM, Thomas J. Watson Reseach Center*, 1986.

[Mil04a] V. S. Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4):235–261, 2004.

[Mil04b] V. S. Miller. The weil pairing, and its efficient calculation. *J. Cryptology*, 17:235–1234, 2004.

[ML03] J. Malone-Lee. Identity-based signcryption. *Cryptology ePrint Archive*, 2003.

[MO90] Morain and Olivos. Speeding up the computations on an elliptic curve using addition-subtraction chains. *RAIRO: R. A. I. R. O. Informatique Theorique et Applications/Theoretical Informatics and Applications*, 24, 1990.

[Mon85] P. L. Montgomery. Modular multiplication without trial division. In *Mathematics of Computation*, volume 44, pages 519–521, 1985.

[MOV93] A. J. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to a finite field. *IEEE Trans. on Inform. Theory*, 39:1639–1646, 1993.

[MOVW89] R. C. Mullin, I. M. Onyszchuk, S. A. Vanstone, and R. M. Wilson. Optimal normal bases in gf(pn). *Discrete Appl. Math.*, 22(2):149–161, 1989.

[MWS04] D. J. Malan, M. Welsh, and M. D. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *1st IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON'04)*, Santa Clara, California, October 2004.

[Niv04] G. Nivash. Cycle detection using a stack. *Inform. Process. Lett.*, 90(3):135–140, 2004.

[NR03] D. Nalla and K. C. Reddy. Signcryption scheme for identity-based cryptosystems. *Cryptology ePrint Archive*, 2003.

[OW99] P. Van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *J. Cryptology*, 12:1–28, 1999.

[PH78] S. Pohlig and M. Hellman. An improving algorithm for computing logarithms over gf_p and it cryptographic singnificance. *IEEE Trans. Inform. Theory*, IT-24:106–110, 1978.

[Pol78] J. M. Pollard. Monte carlo methods for index computation (mod p). *Math. Comp.*, 32:918–924, 1978.

[Pol00] J. M. Pollard. Kangaroos, monopoly and discrete logarithms. *J. Cryptology*, 13:437–447, 2000.

[Rüc99] H.-G. Rück. On the discrete logarithm problem in the divisor class group of curves. *Math. Comp.*, 68:805–806, 1999.

[SA98] T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete logarithm for anomalous elliptic curves. *Comm. Math. Univ. Sancti Pauli*, 47:81–92, 1998.

[Sem98] I. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of a elliptic curve in characteristic p. *Math. Comp.*, 67:353–356, 1998.

[Sha71] D. Shanks. Class number, a theory of factorization and genera. *Proc. Symp. Pure Math*, 20:415–440, 1971.

[Sha85] A. Shamir. Identity-based cryptosystems and signature squemes. In G R Blakley and David Chaum, editors, *Proceedings of CRYPTO 84 on Advances in cryptology*, LNCS 491, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[Sho94] V. Shoup. Exponentiation fo irreducible $gf(2^n)$ using polynomials over finite fileds. *preprint*, 1994.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. *Adances in Cryptology - Eurocrypt, Lecture Notes in Comput. Sci.*, 1233:256–266, 1997.

[Sil07] J. H. Silverman. The four faces os lifting for the elliptic curve discrete logarithm problem. *ECC07*, 2007.

[Sma99] N. P. Smart. The discrete logarithm problem on elliptic curves of trace one. *J. Cryptology*, 12:141–151, 1999.

[Sma02] N. P. Smart. An identity based authenticated key agreement protocol based on pairing. *Eletronic Letters*, 38:630–632, 2002.

[SOK00] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystem based on pairing. In *Symposium on Cryptography and Information Security, Okinawa, Japan*, January 2000.

[Was03] L. C. Washington. *Elliptic Curves: Number Theory and Cryptography*. Chapman & Hall/CRC, 2003.

[WZM07] L. Wang, G. Zhang, and C. Ma. A secure ring signcryption scheme for private and anonymous communication. pages 107–111, 2007.

[Yao76] A. C. Yao. On the evaluation of powers. *SIAM Journal on Computing*, 5(1):100–103, 1976.

[YW03] T. H. Yuen and V. K. Wei. Fast and proven secure blind identity-based signcryption from pairings. In *CT-RSA'05*, volume 3376 of LNCS, pages 305–322, 2003.

[Zhe97] Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) ¡¡ cost(signature) + cost(encryption). *Crypto'97*, 1294:165–179, 1997.

Índice Remissivo