Marina Barcia Schenque

Curvas Elípticas e Suas Aplicações na Criptografia Moderna

Rio Grande, Rio Grande do Sul, Brasil

dezembro, 2025

Marina Barcia Schenque

Curvas Elípticas e Suas Aplicações na Criptografia Moderna

Trabalho de Conclusão de Curso, Matemática Aplicada Bacharelado, submetido por Marina Barcia Schenque junto ao Instituto de Matemática, Estatística e Física da Universidade Federal do Rio Grande.

Universidade Federal do Rio Grande - FURG

Instituto de Matemática, Estatística e Física - IMEF

Curso de Matemática Aplicada Bacharelado

Orientador: Dr. Adilson da Silva Nunes

Rio Grande, Rio Grande do Sul, Brasil

dezembro, 2025

Universidade Federal do Rio Grande – FURG
Instituto de Matemática, Estatística e Física
Curso de Bacharelado em Matemática Aplicada
Av. Itália km 8 Bairro Carreiros
Rio Grande-RS CEP: 96.203-900 Fone (53)3293.5411
e-mail: imef@furg.br Sítio: www.imef.furg.br

Ata de Defesa de Monografia

No décimo sétimo dia do mês de dezembro de 2025, às 17h, no Auditório Prof. Luiz Augusto Andreoli de Moraes do IMEF, foi realizada a defesa do Trabalho de Conclusão de Curso da acadêmica **Marina Barcia Schenque** intitulada **Curvas Elípticas e suas Aplicações na Criptografia Moderna**, sob orientação do Prof. Dr. Adilson da Silva Nunes, deste instituto. A banca avaliadora foi composta pelo Prof. Dr. Luverci do Nascimento Ferreira – IMEF/FURG e o pelo Prof. Dr. Rodrigo Barbosa Soares – IMEF/FURG. A candidata foi: (X) aprovada por unanimidade; ( ) aprovada somente após satisfazer as exigências que constam na folha de modificações, no prazo fixado pela banca; ( ) reprovada. Na forma regulamentar, foi lavrada a presente ata que é abaixo assinada pelos membros da banca, na ordem acima relacionada.

Adilson da Silva Nunes

Prof. Dr. Adilson da Silva Nunes

Orientador

Prof. Dr. Luverci do Nascimento Ferreira

Rodrigo Soares

Prof. Dr. Rodrigo Barbosa Soares

Agradecimentos

Agradeço aos meus pais, Angelo e Márcia Schenque, que me ensinaram a contar mesmo antes de eu saber ler. Que caminhavam pelas ruas da cidade me mostrando os números das casas, e que entregavam as comandas dos restaurantes para que eu somasse os valores "de cabeça". Obrigado pelos abraços, pelos afagos, pelas palavras de conselho, pelo "vai dar tudo certo". Vocês fazem parte da minha jornada desde a primeira vez que me levaram de mãos dadas para o colégio, e continuarão até os últimos dias que eu for um ser humano pensante.

Ao meu orientador, Prof. Dr. Adilson da Silva Nunes, não só pela grande orientação neste Trabalho de Conclusão de Curso, como também pelo "conjunto da obra": por todo o apoio que me forneceu durante os quatro anos de curso, me guiando e me fazendo confiar no meu trabalho. Devo a você muito mais do que um café por tudo o que já fez por mim.

Aos meus amigos de curso: Alana Baldez, Bernardo Schettini, Jhonatan Biller e Luís Fernandes. Obrigado por terem feito o processo de estudar matemática ser menos solitário e mais divertido. Estar com vocês sempre me incentivou a não desistir e a persistir nesse caminho. Também agradeço por toda a ajuda emocional que me proporcionaram, quando sentia que nada iria dar certo.

Aos demais amigos, em especial, à Esther Batista e Miguel Fagundes, que atuaram como uma segunda família para mim nos momentos mais difíceis, me acolhendo e ouvindo meus desabafos. Sou muito grata de ter encontrado vocês pela FURG, e espero que possa levá-los para a vida inteira.

Por fim, agradeço à Universidade Federal do Rio Grande, pelo acesso público e de grandessíssima qualidade à Educação Superior.

"Tudo é possível. O impossível apenas demora mais."
(Dan Brown, Fortaleza Digital, p. 21, 1998)

Resumo

Em um mundo onde milhares de transações de dados ocorrem via meios inseguros (como a internet) a todo instante, o uso da criptografia torna-se essencial para a segurança e privacidade dos usuários, bem como o estudo de diferentes métodos que proporcionem sua melhor eficiência.

A Criptografia de Curvas Elípticas, proposta independentemente por Miller e Koblitz em meados de 1986, propõe a utilização de curvas elípticas para esta finalidade, utilizando-se do fato de que essas curvas, quando aplicadas em corpos finitos, geram grupos abelianos com estruturas que tornam sua decriptação mais lenta do que a de outras criptografias geralmente utilizadas, como a RSA e a de Diffie-Hellman.

Neste trabalho, é proposto o estudo da Criptografia de Curvas Elípticas e suas aplicações, adentrando conceitos da Álgebra e da Geometria Algébrica.

Palavras-chaves: Criptografia, Curvas Elípticas, Problema do Logaritmo Discreto.

Abstract

In a world where thousands of data transactions occur via unsafe means (for instance, the internet) all the time, the use of cryptography becomes essential to the security and privacy of users, as well as the study of different methods that provide its efficiency.

Elliptic Curve Cryptography, proposed independently by Miller and Koblitz around 1986, offers the use of elliptic curves with this goal. It uses the fact that these curves, when applied in finite fields, generate abelian groups with structures that make the decryption slower than that of its counterparts that are generally used, for instance, RSA and Diffie-Hellman.

In this work, it's proposed the study of Eliptic Curve Cryptography and its applications, entering concepts of algebra and algebraic geometry.

Key-words: Cryptography, Elliptic Curves, Discrete Logarithm Problem.

Lista de ilustrações

Lista de tabelas

Sumário

Introdução

A criptografia (do grego, *cryptos*: secreto, oculto e *-grafia*: escrita) consiste na prática de métodos para codificar uma mensagem de modo que apenas o destinatário real consiga decifrá-la. Sua evolução está profundamente ligada à existência da criptoanálise, que tem por objetivo a decriptação destes sistemas criptográficos por meio de análises linguísticas, matemáticas, entre outras.

Sinais de métodos similares à criptografia como forma de comunicação podem ser observados desde 2000 a.C., no Egito e na Mesopotâmia. (COSTA; FIGUEIREDO, 2006) Nesta época, as mensagens eram apenas ocultadas, sem que seu conteúdo fosse modificado. Isto era possível pela utilização de diferentes alfabetos, e sua segurança baseava-se somente na esperança de que a mensagem não entrasse em contato com alguém que tivesse os conhecimentos necessários para decifrá-la. Por conta de sua natureza, este método ainda não pode ser descrito como criptografia, mas sim, como esteganografia.

O início da criptografia clássica é, usualmente, atribuído ao ditador romano Júlio César (100 – 44 a.E.C.), que utilizava um código simples para comunicar-se em combate. A Cifra de César consistia em substituir cada letra do alfabeto por n posições à sua frente, permutando-as. A chave de criptografia, neste caso, era conhecer o valor de n para que fosse possível, posteriormente, permutá-las n posições para trás, retornando ao alfabeto original. Como o alfabeto português possui 26 letras, é possível obter 25 chaves diferentes.

A criptoanálise, neste período, resumia-se a testar por "força bruta"todas as possibilidades de permutações, o que, para os dias atuais, pode parecer uma tarefa relativamente simples. É preciso levar em conta, no entanto, que a grande maioria das populações não sabia ler ou escrever. Na tabela 1, observa-se um exemplo de uma permutação para a Cifra de César com $n = 3$, em que partimos do alfabeto original da primeira linha e o ciframos com base na segunda linha.

Tabela 1 – Exemplo de permutação para a Cifra de César

Alfabeto original	...	V	W	X	Y	Z	A	B	C	D	E	...
Alfabeto permutado	...	Y	Z	A	B	C	D	E	F	G	H	...

Fonte: Elaborado pela autora.

Durante a Idade Média, a utilização de mensagens secretas era atribuída à magia negra ou à bruxaria, e, portanto, amplamente desconsiderada. Os poucos resquícios que existem de criptografia desta época são, também, de cifragem monoalfabética, em que cada letra do alfabeto corresponde a um único símbolo e vice-versa.

Com a idade de ouro da civilização árabe, houve avanços na criptoanálise que tornaram necessária a evolução da criptografia. No século VII, al-Khalil decifrou antigos criptogramas bizantinos baseando-se na ideia de que o título provavelmente seria "Em nome de Deus", método conhecido como "método da palavra provável".

Outro problema que as criptografias monoalfabéticas apresentam, de forma que tornam-se facilmente decifráveis, mesmo não sendo o destinatário legítimo da mensagem, é o fato de que a frequência média em que uma letra específica aparece em um texto de uma determinada língua é mais ou menos constante. Assim, utilizando-se de uma análise de frequência das mesmas, é possível decifrar a mensagem. (COUTINHO, 2016)

Tabela 2 – Frequência média das letras na língua portuguesa (BRAGA, 2003)

Letra	%	Letra	%	Letra	%	Letra	%
A	14,64	G	1,30	N	5,05	T	4,34
B	1,04	H	1,28	O	10,73	U	4,64
C	3,88	I	6,18	P	2,52	V	1,70
D	4,10	J	0,40	Q	1,20	X	0,21
E	12,57	L	2,78	R	6,53	Z	0,47
F	1,02	M	4,75	S	7,81		

Na língua portuguesa, por exemplo, a letra A corresponde à letra de maior frequência média (14,64%), seguida pela letra E (12,57%). Utilizando-se desta informação, é fácil admitir que, quaisquer que sejam as letras que corresponderão às mesmas no novo código, serão as que aparecerão com maior frequência, e, assim, não se torna uma tarefa complicada a de quebrar uma Cifra de César.

Apesar desses conhecimentos árabes, os europeus continuaram utilizando criptografias de fácil decifragem até a Idade Moderna, em que iniciou-se o processo de utilizar cifras homofônicas, nas quais cada vogal poderia ser representada por múltiplos símbolos distintos. Esse método também era utilizado em conjunto com a transposição de letras, como na Cifra de César. Na tabela 3 observa-se um exemplo de uma cifra homofônica.

Tabela 3 – Exemplo de cifra homofônica

Alfabeto original	A	B	C	D	E	F	G	...
Alfabeto cifrado	$\star$	Z	@	M	S	!	∇	...
	X				†			
	?				μ			
	Q							

Fonte: Elaborado pela autora.

Ao longo do tempo, algumas dessas cifras passaram a contar com mais de 500 símbolos diferentes, utilizando diversas representações para vogais, consoantes, dígrafos e sílabas comuns da língua. Apesar do esforço, todas estas cifras eram, eventualmente, quebradas por criptoanalistas da época ou acabavam caindo em desuso devido à alta dificuldade para memorizar seus padrões.

Com o advento do telégrafo, em 1844, a encriptação de mensagens tornou-se um tópico importante para o público geral. Isso ocorreu porque a mensagem precisa ser lida por uma terceira pessoa: o operador do telégrafo. Essas necessidades apenas intensificaram-se com a invenção do rádio.

Durante a Primeira Guerra Mundial, foi utilizada principalmente a cifra ADFGVX (POULTER; KULP, 2017), que utilizava uma combinação de técnicas de transposição e substituição. Essas mensagens cifradas eram comunicadas via rádio, o que as tornava sujeitas a interceptações a todo momento. A decriptação dessas mensagens foi essencial para a resolução de muitas batalhas, uma vez que, sabendo onde o inimigo pretendia atacar, eliminava-se o fator surpresa.

No entanto, a história mais marcante envolvendo criptografia do século XX ocorre durante a Segunda Guerra Mundial, com a máquina de cifragens alemã conhecida como Enigma (SMART, 2016). A máquina, que lembrava uma máquina de escrever, utilizava uma chave que dependia da configuração de montagem, isto é, a ordem e posição dos misturadores, conexão dos cabos emparelhando pares de letras no painel frontal e a posição do refletor. Por questões de segurança, ainda, a chave era trocada após cada mensagem.

Com os avanços da tecnologia, a criptografia tornou-se um elemento essencial para a sociedade. Não mais utilizada apenas em estratégias de guerras, a criptografia está atualmente presente desde mensagens enviadas pela internet, como também em transações financeiras e compras on-line. Isso significa que, diferente dos tempos de Júlio César, em que a chave de codificação poderia ser combinada previamente em um ambiente seguro, as criptografias modernas precisam compartilhar essa chave em um ambiente inseguro, que pode estar sujeito a ataques externos.

Uma das grandes limitações da criptografia, até então, era a crença de que só seria possível utilizar métodos de chave simétricas, ou seja, situações em que a chave para encriptar e decriptar a mensagem fosse a mesma.

Para que a segurança dessas informações fosse alcançada de maneira eficiente, surgiu a necessidade do que é chamado de "criptografia de chave pública"ou criptografia assimétrica, um modelo de encriptação que consiste na utilização de uma chave pública, conhecida por todos os usuários do meio, e uma chave privada, que possui tempo computacional inviável para ser descoberta. Um criptossistema de chave pública deve conter um esquema público de encriptação E e um esquema privado de decodificação D, tal que,

para uma mensagem M,

$$D(E(M)) = M = E(D(M))$$

Em seu artigo "New Directions in Cryptography"(DIFFIE; HELLMAN, 1976), Whitfield Diffie e Martin Hellman apontam, como solução desse problema, uma combinação de função exponencial com aritmética modular, que pôde ser denominada como uma função de mão única. Sua utilização teve início na criptografia RSA, criada por Rivest, Shamir e Adleman (1978), três pesquisadores do Massachusetts Institute of Technology.

Na cifra RSA, para que Alice e Bob enviem mensagens sem que Eva consiga interceptá-las, de alguma forma, são realizadas as seguintes operações:

1. Alice procura pela chave pública de Bob, que está disponível para todos os usuários do meio.
2. Ela encontra o número n, o qual utilizará para cifrar sua mensagem e enviá-la para Bob.
3. A mensagem criptografada chega até Bob, que utiliza sua chave secreta para decifrá-la e ler a mensagem.

Eva, como usuária do meio, conhece a chave n, mas será incapaz de decifrar a mensagem por não ter conhecimento da chave secreta, pois as operações utilizadas para a codificação da mensagem são feitas de maneira que seja computacionalmente inviável encontrar um dos valores a partir do outro.

A teoria que fundamenta a RSA, a qual entraremos em mais detalhes futuramente, utiliza a multiplicação de dois números primos p e q grandes o suficiente que gerem n, e suas operações serão feitas com os números primos em si, que, mesmo com n conhecido, serão desconhecidos. A teoria estabelece que, se um número n tem mais do que 10^{160} dígitos e é obtido como produto de dois números primos, cada qual com mais de 10^{70} dígitos, então o tempo computacional para encontrar estes fatores é maior do que a idade do universo.

Outro modelo de criptografia assimétrica promissor é a Criptografia de Curvas Elípticas (ou ECC), independentemente proposta por (MILLER, 1986) e (KOBLITZ, 1987). Esta cifragem utiliza da estrutura de grupos abelianos que podem ser gerados a partir das curvas elípticas. Enquanto a segurança do modelo RSA é baseada no problema da fatoração de inteiros suficientemente grandes, a segurança da ECC baseia-se no problema do logaritmo discreto.

A escolha, em certos casos, da utilização da ECC, deve-se ao fato de que o problema da fatoração de inteiros pode ser resolvido por um algoritmo de tempo subexponencial, enquanto que o problema do logaritmo discreto demora tempo exponencial completo. Isso

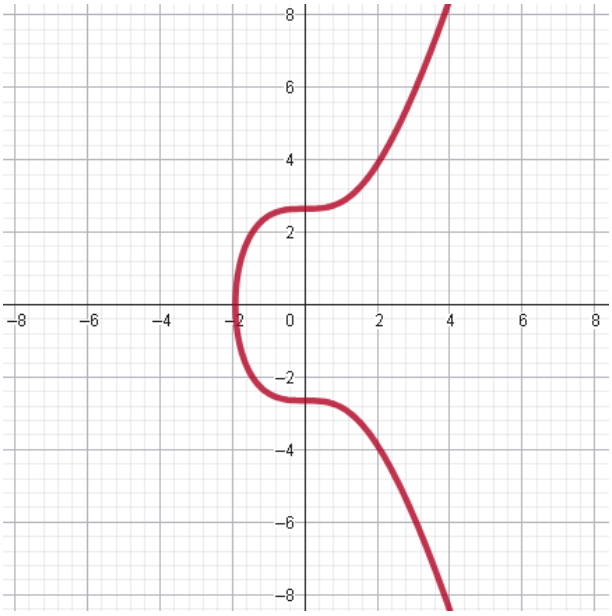

Figura 1 – Exemplo de curva elíptica utilizada para o Bitcoin ($y^2 = x^3 + 7$)

Fonte: Elaborado pela autora.

implicará que, para atingir o mesmo nível de segurança em ambos os métodos, a ECC exigirá parâmetros menores.

Devido ao fato de que os sistemas computacionais estão se tornando cada vez menores e mais restritos, mas, ao mesmo tempo, possuem uma necessidade maior de segurança, a ECC surge como uma alternativa viável em certas ocasiões. Na tabela 4, comparamos o comprimento da chave de criptografia necessária, em bits, para obter a mesma quantidade de bits de segurança.

Um valor n de bits de segurança, neste caso, significa que seriam necessárias 2^n operações para quebrar os códigos. Percebe-se que a ECC apresenta maior eficiência, visto que, para obter uma chave com 256 bits de segurança, precisa de aproximadamente metade do tamanho de uma chave RSA capaz de fornecer apenas 80 bits de segurança.

Tabela 4 – Comparação do comprimento da chave de criptografia (em bits) (BARKER, 2020)

Bits de segurança	RSA	ECC
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

A ECC utiliza, para sua formulação, as curvas elípticas. Essas curvas são descritas

pela equação

$$y^2 = x^3 + ax + b,$$

onde ainda são excluídas as situações em que $4a^3+27b^2 = 0$, a fim de evitar singularidades. A vantagem na utilização dessas curvas deve-se ao fato de que as curvas elípticas são capazes de formar grupos abelianos.

Essas curvas são aplicadas sobre campos finitos $\mathbb{F}_p$, isto é, o conjunto de inteiros módulo p, onde p é um número primo. Dessa forma, as vantagens em sua utilização são evidenciadas pelas estruturas algébricas que as compõem.

Por fim, a segurança do modelo será baseada no problema do logaritmo discreto. Este problema pode ser explicado como: conhecidos os pontos P e Q, encontrar k tal que $Q = kP$, o que pode ser traduzido sobre os campos finitos $\mathbb{F}_p$ para, conhecidos a e b, desejamos encontrar k tal que $b = a^k \ mod \ p$. Desde que se evite curvas supersingulares e também curvas cuja ordem não tenha nenhum fator primo grande, não é conhecido nenhum algoritmo sub-exponencial que possa quebrar o sistema.

1 Objetivos

1.1 Objetivos gerais

O presente trabalho tem como objetivo principal estudar a Criptografia de Curvas Elípticas (ECC), bem como analisar suas vantagens e desvantagens em relação a outras técnicas criptográficas amplamente utilizadas, como a Criptografia RSA. Esse objetivo foi alcançado por meio do estudo de temas como Aritmética e Geometria Algébrica, que fundamentam a formulação matemática desses modelos.

1.2 Objetivos específicos

Buscou-se compreender algoritmos baseados na ECC, como o Diffie–Hellman de Curvas Elípticas (ECDH) e o Algoritmo de Assinatura Digital de Curvas Elípticas (ECDSA). Esses algoritmos utilizam a ECC em formato híbrido, incorporando a ela um componente de chave simétrica.

Para a conclusão do trabalho, estabeleceu-se ainda o objetivo de elaborar um código em Python exemplificando o uso tanto da Criptografia de Curvas Elípticas quanto da Criptografia RSA para criptografar e decriptar mensagens, de modo a possibilitar a comparação do tempo computacional entre ambas.

O trabalho está disposto da seguinte forma: o capítulo 2 apresenta conceitos de Teoria dos Números e Aritmética que serão necessários para o capítulo 3, que introduz a Criptografia RSA. Em seguida, o mesmo ocorre nos capítulos 4 e 5, no qual o primeiro introduz a Álgebra necessária para a compreensão do segundo, que trata-se da Criptografia de Curvas Elípticas. O capítulo 6 apresenta a implementação do que foi estudado em Python e o capítulo 7 conclui o trabalho.

2 Teoria dos Números e Aritmética

A Aritmética é conhecida como a parte elementar da Teoria dos Números, e teve como marco inicial a obra Os Elementos, de Euclides, aproximadamente 300 anos a.E.C.. No entanto, ela apenas se tornaria um dos pilares na matemática com os estudos de Pierre de Fermat (1601-1665) e Leonhard Euler (1707-1783).

Estas áreas são identificadas pelo estudo das propriedades e relações que os números possuem entre si, e terão extrema importância para o estudo da Criptografia RSA, já que a mesma necessita da aritmética modular para criptografar suas chaves.

2.1 Teoria dos Números

2.1.1 Divisilibidade

Definição 2.1.1. Dados $a, b \in \mathbb{Z}$. Dizemos que a divide b quando existir algum inteiro $c \in \mathbb{Z}$ tal que

$$b = a * c$$

Neste caso, diremos também que a é um divisor ou um fator de b, ou ainda, que b é divisível por a e c é o quociente.

A notação para indicar que a divide b é $a|b$, enquanto que, para a não divide b, a notação será $a \nmid b$. A negação dessa sentença indica que não existe nenhum número inteiro c tal que $b = c * a$.

Exemplo 2.1.1. $2|6, 3|6, 5 \nmid 6$.

Teorema 2.1.1. Se $a, b, c \in \mathbb{N}$, com $a \neq 0$ são tais que $a|b$ e $a|c$. Dados $x, y \in \mathbb{N}$, $a|(xb + yc)$, e, se $xb \geq yc$, então $a|(xb - yc)$.

A demonstração pode ser encontrada em (HEFEZ, 2006).

Teorema 2.1.2 (Divisão Euclidiana)**.** Sejam a e b dois números naturais com $0 < a < b$. Existem dois únicos números naturais r e q tais que:

$$b = a * q + r,\ r < a$$

A demonstração pode ser encontrada em (HEFEZ, 2006).

Exemplo 2.1.2. Vamos encontrar o quociente e o resto da divisão de 22 por 6.

Considere as diferenças sucessivas:

$$22 - 6 = 16, 22 - 2 * 6 = 10, 22 - 3 * 6 = 4 < 6$$

Portanto, $q = 3$ e $r = 4$.

Definição 2.1.2. Sejam $a, b \in \mathbb{Z}$, não simultaneamente nulos, o máximo divisor comum entre os números inteiros a e b é o maior inteiro positivo d que satisfaz as seguintes condições:

- (i) d é um divisor comum de a e b;
- (ii) Se d' é um divisor comum de a e b, então $d'|d$.

Definição 2.1.3. Sejam $a, b \in \mathbb{N}^*$. Definimos o conjunto

$$J(a, b) = \{x \in \mathbb{N}^*; \exists u, v \in \mathbb{N}, x = ua - vb\}$$

Teorema 2.1.3. Sejam $a, b \in \mathbb{N}^*$ e seja $d = min\ J(a, b)$. Temos que d é o mdc de a e b.

Demonstração. Suponhamos que c divide a e b. Logo, c divide todos os números da forma $ua - vb$, portanto, divide todos os elementos de $J(a, b)$, logo, $c|d$.

Mostraremos que d divide todos os elementos de $J(a, b)$. Seja $x \in J(a, b)$ e, supomos por absurdo, que $d \not| x$. Assim, pela Divisão Euclidiana, $x = dq + r$, com $0 < r < d$.

Como $x = ua - vb$ e $d = mb - na$, para alguns $u, v, m, n \in \mathbb{N}$, segue-se que $r = (u + qn)a - (v + qm)b \in J(a, b)$. Mas isso é um absurdo, pois $d = min\ J(a, b)$ e $r < d$. Em particular, $d|a$ e $d|b$.

□

Proposição 2.1.1. Dois números naturais a e b são primos entre si se, e somente se, existem números naturais m e n tais que $na - mb = 1$.

Demonstração. Suponhamos a e b primos entre si. Logo, $mdc(a, b) = 1$. Como já visto, temos que existem n e m tais que $na - mb = mdc(a, b) = 1$, segue-se a primeira parte da preposição.

Reciprocamente, suponhamos que existam números naturais n e m tais que $na - mb = 1$. Se $d = mdc(a, b)$, temos que $d|(na - mb)$, ou seja $d|1$, e, portanto, $d = 1$.

□

Teorema 2.1.4. Sejam a, b, c números naturais. Se $a|b * c$ e $mdc(a, b) = 1$, então $a|c$.

Demonstração. Se $a|b * c$, então existe $e \in \mathbb{N}$ tal que $bc = ae$. Se $mdc(a, b) = 1$, então, temos que existem $m, n \in \mathbb{N}$, tais que $na - mb = 1$. Multiplicando por c, obtemos

$$c = nac - mbc$$

Mas $bc = ae$, então $c = nac - mae = a(nc - ne)$. Portanto, $a|c$.

□

2.1.2 Primalidade

Definição 2.1.4 (Número primo)**.** Um número natural maior do que 1 e divisível apenas por 1 e por ele mesmo é chamado de número primo.

Ocorre da definição acima que, dados p e q números primos e a um número natural qualquer, será verdade que:

1. Se $p|q$, então $p = q$.
2. Se $p \nmid a$, então $mdc(p, a) = 1$.

A primeira afirmação surge do fato de que, como $p|q$ e q é primo, só será possível $p = 1$ ou $p = q$. Como p é primo, tem-se que $p > 1$, da onde sai que $p = q$.

A segunda afirmação ocorre pois, se $mdc(p, a) = d$, então $d|p$ e $d|a$. Então $d = 1$ ou $d = p$. Mas $d \neq p$, pois $p \nmid a$, do que se pode concluir que $mdc(p, a) = 1$.

Proposição 2.1.2. Sejam $a, b, p \in \mathbb{N}^*$, com p primo. Se $p|ab$, então $p|a$ ou $p|b$.

Demonstração. Basta provar que, se $p|ab$ e $p \nmid a$, então $p|b$. Mas, se $p \nmid a$, então $mdc(p, a) = 1$, e segue que $p|b$.

□

Corolário 2.1.1. Se $p, p_1, ..., p_n$ são números primos e, se $p|p_1...p_n$, então $p = p_i$ para algum $i = 1, ..., n$.

Demonstração. Usando a proposição anterior, utilizamos a indução sobre n e o fato de que, se $p|p_i$, então $p = p_i$.

□

Teorema 2.1.5 (Teorema Fundamental da Aritmética)**.** (GAUSS, 1801) Todo número natural maior do que 1 ou é primo ou se escreve, de modo único, como um produto de números primos.

Demonstração. Se $n = 2$, o resultado é automaticamente verificado.

Supomos, então, que este resultado é válido para todo número natural menor do que n, e provaremos que vale para n: se o número n é primo, nada temos a demonstrar. Suponhamos, então, que n seja composto.

Logo, existem números naturais n_1 e n_2 tais que $n = n_1n_2$, com $1 < n_1 < n$ e $1 < n_2 < n$. Pela hipótese, existem números primos $p_1, ..., p_r$ e $q_1, ..., q_s$ tais que $n_1 = p_1...p_r$ e $n_2 = q_1...q_s$. Portanto, $n = p_1...p_rq_1...q_s$.

Mostraremos, também, a unicidade da escrita. Suponha que $n = p_1...p_r = q_1...q_s$, onde p_i e p_j são números primos. Como $p_1|q_1...q_s$, pelo corolário anterior, temos que $p_1 = q_j$ para algum j que, após reordenar $q_1...q_s$, podemos assumir que seja q_1. Portanto, $p_2...p_r = q_2...q_s$. Como $p_2...p_r < n$, a hipótese de indução conclui que $r = s$ e os p_i e q_j são iguais a seus pares. □

Ordenando os primos em ordem crescente e contabilizando os fatores repetidos, este resultado também pode ser escrito da seguinte forma:

Teorema 2.1.6. Dado um número $n \in \mathbb{N}, n > 1$, existem primos $p_1 < ... < p_r$ e $\alpha_1, ...\alpha_r \in \mathbb{N}^*$, univocamente determinados, tais que

$$n = p_1^{\alpha_1}...p_r^{\alpha_r}$$

A demonstração pode ser encontrada em (HEFEZ, 2006).

Teorema 2.1.7. Existem infinitos números primos.

Demonstração. Suponha que exista apenas um número finito de números primos $p_1, ...p_r$.

Considere o número natural $n = p_1p_2...p_r + 1$. O número n possui um fator primo p que, portanto, deve ser um dos $p_1, ..., p_r$ e, por consequência, p divide o produto $p_1p_2...p_r$.

Mas isso implica que p divide 1. Absurdo!

□

Teorema 2.1.8 (Lema de Eratóstenes)**.** Se um número inteiro $n > 1$ não é divisível por nenhum primo p tal que $p^2 \leq n$, então ele é primo.

Demonstração. Por absurdo, suponhamos que n não seja divisível por nenhum número primo p tal que $p^2 \leq n$ e que n não seja primo.

Se n for composto, segue que existe algum primo q, o menor número primo que divide n. Isto é, $n = q * n_1$ para algum $n_1 \in \mathbb{Z}$ com $q \leq n_1$, pois q é o menor primo que divide n.

Ao multiplicar a desigualdade por q, segue daí que $q^2 \leq qn_1 = n$. Logo, n é divisível por um número primo q tal que $q^2 \leq n$. Absurdo!

□

Baseado neste último Lema, foi criado o Crivo de Eratóstenes, um método para descobrir todos os números primos até um certo número natural n. Este método consiste em, partindo do número 2, excluir todos os valores da tabela que sejam múltiplos de 2, depois os múltiplos de 3, e assim por diante. De acordo com o Lema, apenas será necessário fazer este processo até o valor de $\sqrt{n}$.

Na tabela 5, obtivemos o Crivo de Eratóstenes para $n = 120$. Para ela, apenas foi necessário testar os valores até $\sqrt{n} = \sqrt{120}$, ou seja, até 7, visto que o próximo número primo, 11, ultrapassa a raíz quadrada, e os valores entre 7 e 11 já estariam previamente excluídos.

2.2 Aritmética Modular

Definição 2.2.1. Dizemos que dois números inteiros a e b são congruentes módulo n se os restos de sua divisão euclidiana de a e b por n são iguais. Escrevemos isso como:

$$a \equiv b(mod\ n)$$

Tabela 5 – Crivo de Eratóstenes para $n = 120$

	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~	11	~~12~~
13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~	~~21~~	~~22~~	23	~~24~~
~~25~~	~~26~~	~~27~~	~~28~~	~~29~~	~~30~~	31	~~32~~	~~33~~	~~34~~	~~35~~	~~36~~
37	~~38~~	~~39~~	~~40~~	41	~~42~~	43	~~44~~	~~45~~	~~46~~	47	~~48~~
~~49~~	~~50~~	~~51~~	~~52~~	53	~~54~~	~~55~~	~~56~~	~~57~~	~~58~~	59	~~60~~
61	~~62~~	~~63~~	~~64~~	~~65~~	~~66~~	67	~~68~~	~~69~~	~~70~~	71	~~72~~
73	~~74~~	~~75~~	~~76~~	~~77~~	~~78~~	79	~~80~~	~~81~~	~~82~~	83	~~84~~
~~85~~	~~86~~	~~87~~	~~88~~	89	~~90~~	~~91~~	~~92~~	~~93~~	~~94~~	~~95~~	~~96~~
97	~~98~~	~~99~~	~~100~~	101	~~102~~	103	~~104~~	~~105~~	~~106~~	107	~~108~~
109	~~110~~	~~111~~	~~112~~	113	~~114~~	~~115~~	~~116~~	~~117~~	~~118~~	~~119~~	~~120~~

Propriedades da Congruência Modular:

1. Todo número é congruente módulo n a si próprio, ou seja, $a \equiv a \; mod \; n$;
2. Se $a \equiv b \; mod \; n$, então $b \equiv a \; mod \; n$;
3. Se $a \equiv b \; mod \; n$ e $b \equiv c \; mod \; n$, então $a \equiv c \; mod \; n$;
4. Se $a \equiv a' \; mod \; n$ e $b \equiv b' \; mod \; n$, então $a + b \equiv a' + b' \; mod \; n$ e $a * b \equiv a' * b' \; mod \; n$;
5. Em particular, $a^k \equiv (a')^k \; mod \; n$, para qualquer $k \geq 0$.

Proposição 2.2.1. Suponha $a, b \in \mathbb{N}$ tais que $b \geq a$. Tem-se que $a \equiv b \; mod \; m$ se, e somente se, $m|b - a$.

Demonstração. Pela Divisão Euclidiana, temos que $a = mq + r$, com $r < m$ e $b = mq' + r'$, com $r' < m'$. Logo,

$$b - a = \begin{cases} m(q' - q) + (r' - r), & \text{se } r' \geq r \\ m(q' - q) - (r' - r), & \text{se } r \geq r' \end{cases}$$

onde $r' - r < m$ ou $r - r' < m$. Portanto, $a \equiv b \; mod \; m$ se, e somente se, $r = r'$, o que equivale a dizer que $m|b - a$.

□

Definição 2.2.2 (Sistema Completo de Resíduos)**.** Chamamos de sistema completo de resíduos módulo m todo conjunto de números naturais cujos restos pela divisão por m são os números $0, 1, ..., m-1$, sem repetições e numa ordem qualquer. Assim, sistema completo de resíduos módulo m possui m elementos.

Proposição 2.2.2. Sejam $a, m \in \mathbb{N}$, com $m > 1$. A congruência $aX \equiv 1 \; mod \; n$ possui uma solução x_0 se, e somente se, $mdc(a, m) = 1$. Além disso, x é uma solução da congruência se, e somente se, $x \equiv x_0 \; mod \; m$.

Demonstração. A congruência terá solução x_0 se, e somente se, $m|ax_0 - 1$, o que equivale a dizer que a equação diofantina $aX - mY = 1$ possui solução em números naturais. Isso ocorre se, e somente se, $mdc(a, m) = 1$.

Por outro lado, se x_0 e x são equações da congruência $aX \equiv 1 \; mod \; n$, então $ax \equiv ax_0 \; mod \; m$, o que implica que $x \equiv x_0 \; mod \; m$. Se x_0 é solução da congruência e $x \equiv x_0 \; mod \; m$, então x também é solução da congruência, pois

$$ax \equiv ax_0 \equiv 1 \; mod \; m$$

Se considerarmos que duas soluções congruentes módulo m são, essencialmente, a mesma, temos ainda a unicidade da solução da congruência $aX \equiv 1 \; mod \; n$.

□

2.2.1 Função $\phi(m)$ de Euler

Definição 2.2.3. Designaremos por $\phi(m)$ (ou função phi de Euler) o número de elementos de um sistema reduzido de resíduos módulo m que corresponde à quantidade de números naturais entre 0 e $n-1$ que são primos com m.

$$\phi : \mathbb{N}^* \to \mathbb{N}$$

Note que $\phi(m) \leq m - 1$, sendo que $\phi(m) = m - 1 \Leftrightarrow$ m é primo.

Teorema 2.2.1 (Teorema de Euler)**.** Sejam $m, a \in \mathbb{Z}$ com $m > 1$ e $mdc(a, m) = 1$. Então,

$$a^{\phi(m)} \equiv 1 \; (mod \; m)$$

Demonstração. Seja $r_1, ..., r_{\phi(m)}$ um sistema reduzido de resíduos módulo m. Logo, $a * r_1, ..., a * r_{\phi(m)}$ também formam um sistema reduzido de resíduos módulo m. Assim,

$$a^{\phi(m)} r_1 * r_2 * ... * r_{\phi(m)} = ar_1 * ar_2 * ... * ar_{\phi(m)} \equiv r_1 * r_2 * ... * r_{\phi(m)} \; mod \; m$$

Como $mdc(r_1 * r_2 * ... * r_{\phi(m)}, m) = 1$, $a^{\phi(m)} \equiv 1 \; (mod \; n)$. □

Teorema 2.2.2 (Pequeno Teorema de de Fermat)**.** Sejam $a \in \mathbb{Z}$ e p um número primo tais $mdc(a, p) = 1$. Se $p \nmid a$, então:

$$a^{p-1} \equiv 1 \; (mod \; p)$$

Demonstração. Basta utilizar o Teorema de Euler e notar que, p sendo primo, $\phi(p) = p - 1$. □

Teorema 2.2.3 (Teorema do Resto Chinês)**.** (HEFEZ, 2006) O sistema

$$X \equiv c_1 \; mod \; n_1$$

$$X \equiv c_2 \; mod \; n_2$$

$$...$$

$$X \equiv c_r \; mod \; n_r,$$

onde $mdc(n_i, n_j) = 1$, para todo par n_i, n_j com $i \neq j$, possui uma única solução módulo $N = n_1 * n_2 * ... * n_r$. Tal solução pode ser obtida desta forma:

$$x = N_1 * y_1 * c_1 + ... + N_r * y_r * c_r,$$

onde $N_i = \frac{N}{n_i}$ e y_i é a solução de $N_i Y \equiv 1 \; mod \; n_i, i = 1, ..., r$.

Demonstração. ($\Rightarrow$) Inicialmente, provamos que x é uma solução simultânea do sistema. De fato, como $n_i | N_j$, se $i \neq j$ e $N_i y_i \equiv 1 \; mod \; n_i$, segue-se que

$$x = N_1 * y_1 * c_1 + ... + N_r * y_r * c_r \equiv N_i * y_i * c_i \equiv c_i \; mod \; n_i.$$

($\Leftarrow$) Por outro lado, se x' é outra solução do sistema, então

$$x \equiv x' \; mod \; n_i, \forall i, i = 1, ..., r.$$

Como $mdc(n_i, n_j) = 1$, para $i \neq j$, segue-se que $[n_1, ..., n_r] = n_1 * ... * n_r = N$ e, consequentemente, temos que $x \equiv x' \; mod \; N$. □

Proposição 2.2.3. Sejam $m, m' \in \mathbb{N}$, com $m > 1$, $m' > 1$ e $mdc(m, m') = 1$,

$$\phi(m * m') = \phi(m) * \phi(m')$$

Demonstração. Consideramos a seguinte tabela formada pelos números naturais de 1 até $m * m'$:

1	2	...	k	...	m'
m' + 1	m'+2	...	m'+k	...	2m'
⋮	⋮		⋮		⋮
(m-1)m'+1	(m-1)m'+2	...	(m-1)m'+k	...	mm'

Como $mdc(t, m * m') = 1$ se, e somente se, $mdc(t, m) = mdc(t, m') = 1$, para calcular $\phi(m * m')$, determinamos os números da tabela que são simultaneamente primos com m e com m'.

Se o primeiro elemento da coluna não for primo com m', todos os elementos não são. Dessa forma, os elementos primos com m' estão obrigatoriamente nas colunas restantes, que são, em número, $\phi(m')$.

Vejamos quais são os elementos primos com m destas colunas.

Como $mdc(m, m') = 1$, a sequência

$$k, m' + k, ..., (m - 1)m' + k$$

forma um sistema completo de resíduos módulo m, e, portanto, $\phi(m)$ desses elementos são primos com m.

Portanto, o número de elementos simultaneamente primos com m' e m é $\phi(m) * \phi(m')$.

□

Lema 2.2.1. Se p é um número primo e r é um número natural, tem-se que

$$\phi(p^r) = p^r - p^{r-1} = p^r(1 - \frac{1}{p}).$$

Demonstração. De 1 até p^r, temos p^r números naturais. Excluiremos destes os que não forem primos com p^r, ou seja, todos os múltiplos de p, que são $p, 2p, ..., p^{n-1}p$, de modo que são exatamente p^{n-1} elementos. Portanto, $\phi(p^r) = p^r - p^{r-1}$.

□

Teorema 2.2.4. Se $m = p_1^{\alpha_1}...p_n^{\alpha_n}$ é a decomposição de m em fatores primos, então

$$\phi(m) = p_1^{\alpha_1}...p_n^{\alpha_n}(1 - \frac{1}{p_1})...(1 - \frac{1}{p_n}),$$

que também pode ser escrito como

$$\phi(m) = p_1^{\alpha_1 - 1}...p_n^{\alpha_n - 1}(p_1 - 1)...(p_n - 1).$$

Demonstração. O resultado decorre do Lema acima.

□

3 Critografia RSA

Como já visto anteriormente, com a grande utilização de meios inseguros para a comunicação nos dias de hoje, urge a necessidade de que estas trocas de mensagem possuam duas propriedades importantes: a privacidade do conteúdo e a possibilidade de assiná-lo de alguma forma, para que exista uma comprovação do autor real da mensagem.

Para isso, Whitfield Diffie e Martin Hellman desenvolveram, como solução para os problemas que a criptografia enfrentava há séculos, a chamada "Criptografia de Chave Pública". Esse método consiste na utilização de duas chaves diferentes: uma chave pública, conhecida por todos do meio; e uma chave privada, conhecida apenas pelo destinatário da mensagem. (HELLMAN, 1978)

A Criptografia RSA (RIVEST; SHAMIR; ADLEMAN, 1978), datada dos anos 70, é um dos primeiros métodos de chave pública amplamente utilizados. Ela utiliza uma combinação de função exponencial com aritmética modular, que é conhecida como uma função de mão única, de tal modo que torna-se computacionalmente inviável tentar decriptar a mensagem sem ter a chave privada.

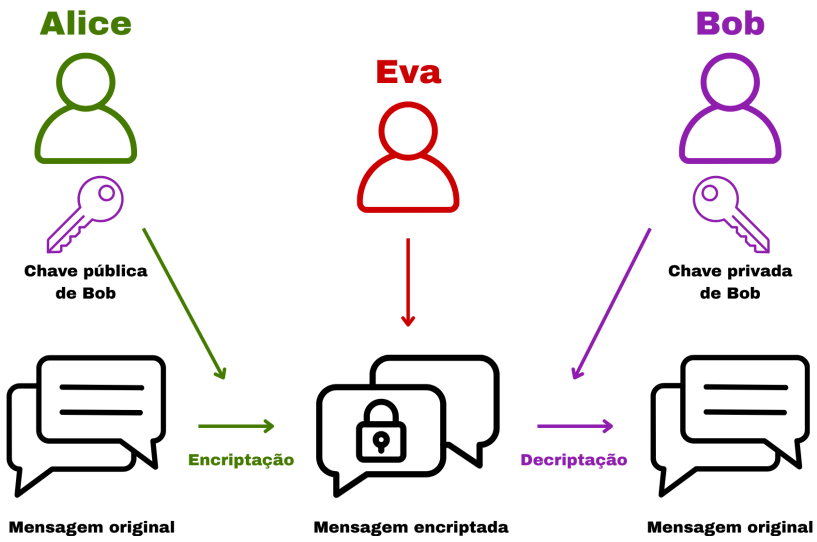

Figura 2 – Exemplo de encriptação de mensagem com a criptografia de chave pública

Fonte: Elaborado pela autora.

Em uma criptografia de chave pública, ao enviar uma mensagem para Bob, Alice utiliza a chave pública dele para encriptar o conteúdo, de modo que, caso a mensagem seja

interceptada por Eva, ela não será capaz de compreendê-la. Em seguida, ocorre o processo de decriptação, no qual Bob utiliza sua chave privada para ler a mensagem verdadeira.

3.1 Método de funcionamento da RSA

Tomando uma mensagem M, o RSA utiliza uma chave pública (e, n) e uma chave privada d para realizar os cálculos necessários. Abaixo, será brevemente explicado como é dado seu funcionamento.

1. Inicialmente, a mensagem a ser enviada é representada como um inteiro entre 0 e $n-1$ (caso necessário, é preciso quebrar a mensagem em blocos), tomando n como o produto de dois primos p e q.
2. A partir deste número, obteremos a função phi de Euler, $\Phi(n) = (p-1)(q-1)$, do qual extrairemos a outra chave pública e e a chave privada d.
3. Estes valores serão obtidos tomando d como um número coprimo de $\Phi(n)$, e e de modo que $ed \equiv 1 \bmod \Phi(n)$.
4. O resultado criptografado que será enviado para o segundo usuário será $C \equiv M^e \bmod n$.
5. Para decifrar a mensagem, ele deverá, então, realizar a operação com a chave privada $M \equiv C^d \bmod n$.

Exemplo 3.1.1. Tomando n como o produto dos primos $p = 11$ e $q = 17$, obteremos

$$n = 11 * 17 = 187$$

e, por sua vez,

$$\Phi(n) = (11-1) * (17-1) = 160.$$

Podemos utilizar $d = 7$, visto que $mdc(7, 160) = 1$, e, portanto d e $\Phi(n)$ são coprimos. Deste modo, ao calcular o valor de e, obteremos $e = 23$, pois

$$ed = 7 * 23 = 161 \equiv 1 \; mod \; 160.$$

Então, encontramos que $n = 187, d = 7, e = 23$.

Supondo que a mensagem a ser encriptada seja $M = 8$, teremos

$$C \equiv M^e \; mod \; n \equiv 8^{23} \; mod \; 187$$

$$\equiv 59029581035870565 1712 \; mod \; 187 \equiv 83 \; mod \; 187.$$

Logo, o destinatário receberá C = 83.

Após isso, será utilizado o valor privado de $d = 7$ para decriptar a mensagem, e teremos

$$M \equiv C^d \ mod \ n \equiv 83^7 \ mod \ 187$$

$$\equiv 27136050989627 \ mod \ 187 \equiv 8 \ mod \ 187$$

Deste modo, o destinatário finalmente receberá a mensagem original $M = 8$.

Exemplo 3.1.2. É claro que, comumente, será necessário também encriptar mensagens mais longas, bem como palavras ou textos. Como exemplo deste processo para uma mensagem maior, iremos criptografar a palavra **MATEMATICA**. Para a simplificação do processo, consideraremos todas as letras em maiúsculo, sem a presença de acentos.

Para isso, será necessário, inicialmente, converter todas as letras em números. Utilizaremos a tabela 6 para isso.

Perceba que os valores deverão começar com $A = 10$, para haver clareza sobre os blocos que representam as letras. Devido ao fato de que será feito uma concatenação dos blocos em seguida, caso resolvêssemos iniciar por $A = 1, B = 2, ..., Z = 26$, ao encontrarmos a mensagem $M = 12$, não saberíamos dizer se ela corresponde aos valores de $A = 1$ e $B = 2$ ou apenas a um único bloco $L = 12$.

Tabela 6 – Tabela de Conversão

A	B	C	D	E	F	G	H	I	J	K	L	M
10	11	12	13	14	15	16	17	18	19	20	21	22
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
23	24	25	26	27	28	29	30	31	32	33	34	35

Assim, convertendo a mensagem letra a letra, obtemos:

$$M = 22$$

$$A = 10$$

$$T = 29$$

$$E = 14$$

$$M = 22$$

$$A = 10$$

$$T = 29$$

$$I = 18$$

$$C = 12$$

$$A = 10$$

Portanto, a mensagem a ser encriptada será a concatenação $M = 22102914221029181210$.

Utilizando, agora, $p = 17$ e $q = 19$, teremos $n = p * q = 17 * 19 = 323$, e, ainda, $\phi(n) = (p-1) * (q-1) = 16 * 18 = 288$.

Então, iniciamos o processo, quebrando a mensagem em blocos, de modo que, em cada bloco, permaneça um valor menor do que $n = 323$. Assim,

M_1	M_2	M_3	M_4	M_5	M_6	M_7
221	0291	42	210	291	81	210

Fonte: Elaborado pela autora.

Criptografamos cada um dos blocos pela fórmula $C_i \equiv (M_i)^e \ mod \ n$. Neste caso, utilizaremos $e = 5$.

$$C_1 = 221^5 \ mod \ 323 \equiv 255 \ mod \ 323$$

$$C_2 = 291^5 \ mod \ 323 \equiv 100 \ mod \ 323$$

$$C_3 = 42^5 \ mod \ 323 \equiv 264 \ mod \ 323$$

$$C_4 = 210^5 \ mod \ 323 \equiv 58 \ mod \ 323$$

$$C_5 = 291^5 \ mod \ 323 \equiv 100 \ mod \ 323$$

$$C_6 = 81^5 \ mod \ 323 \equiv 47 \ mod \ 323$$

$$C_7 = 210^5 \ mod \ 323 \equiv 58 \ mod \ 323$$

De modo que a mensagem encriptada será 255.100.264.58.100.47.58.

A descriptografia será feita a partir da chave privada d, que podemos calcular em função dos outros valores que possuímos.

Como e e d precisam ser, necessariamente, inversos multiplicativos módulo $\phi(n)$, encontramos que $d = 173$, pois

$$5 * 173 = 865 = 3 * 288 + 1,$$

ou seja,

$$ed \equiv 1 \ mod \ 288.$$

Para cada bloco criptografado, fazemos $M_i \equiv (C_i)^d \ mod \ n$.

$$M_1 = 255^{173}\ mod\ 323 \equiv 221\ mod\ 323$$

$$M_2 = 100^{173}\ mod\ 323 \equiv 291\ mod\ 323$$

$$M_3 = 264^{173}\ mod\ 323 \equiv 42\ mod\ 323$$

$$M_4 = 58^{173}\ mod\ 323 \equiv 210\ mod\ 323$$

$$M_5 = 100^{173}\ mod\ 323 \equiv 291\ mod\ 323$$

$$M_6 = 47^{173}\ mod\ 323 \equiv 81\ mod\ 323$$

$$M_7 = 58^{173}\ mod\ 323 \equiv 210\ mod\ 323$$

Voltamos, assim, à mensagem original $M = 221.291.42.210.291.81.210$. A partir daqui, é necessário apenas reorganizar a mensagem de dois em dois algarismos e conferir na tabela para verificar que retornamos à mensagem **MATEMATICA**.

3.2 Fundamentação para seu funcionamento

Mostraremos que

$$D(E(M)) = M = E(D(M))$$

.

Neste caso, seja a um bloco da mensagem, mostraremos que

$$(a^e)^d \equiv (a^d)^e \equiv a\ mod\ n.$$

Uma das propriedades da potenciação transformará estes expoentes em um produto, que será comutativo. Portanto, basta mostrar

$$(a^e)^d \equiv a\ mod\ n.$$

Para isso, utilizaremos a propriedade

$$a \equiv b\ mod\ pq \Leftrightarrow \begin{cases} a \equiv b\ mod\ p \\ a \equiv b\ mod\ q \end{cases}$$

Como sabemos que

$$ed \equiv 1\ mod\ \phi(n)$$

e

$$\phi(n) = (p-1)(q-1),$$

podemos escrever

$$ed = k(p-1)(q-1) + 1, \text{para algum k natural.}$$

Provaremos, então que

$$a^{k(p-1)(q-1)+1} \equiv a \; mod \; pq,$$

que será equivalente a provar que

1. $a^{k(p-1)(q-1)+1} \equiv a \; mod \; p$
2. $a^{k(p-1)(q-1)+1} \equiv a \; mod \; q$

Demonstração (1).

Se $p|a$, então $0 \equiv a \equiv a^{k(p-1)(q-1)+1} \; mod \; p$.

Se p não divide a, pelo Pequeno Teorema de Fermat, temos

$$a^{p-1} \equiv 1 \; mod \; p$$

então,

$$\left[a^{p-1}\right]^{k(q-1)} \equiv 1^{k(q-1)} \equiv 1 \; mod \; p$$

Podemos multiplicar a equivalência por a, obtendo

$$a^{k(p-1)(q-1)+1} \equiv a \; mod \; p$$

□

A demonstração (2) será análoga.

4 Álgebra

O seguinte capítulo foi baseado em Domingues e Iezzi (2003), e trata-se da fundamentação algébrica necessária para os próximos resultados.

4.1 Álgebra

4.1.1 Grupos

Definição 4.1.1. Seja $\star$ uma operação definida em um conjunto G. Dizemos que o par $(G, \star)$ é um grupo se, e somente se

- O conjunto G é fechado sob a operação, isto é, $\forall g, h \in G, g \star h \in G$
- A operação $\star$ é associativa, isto é, $\forall g, h, k \in G, (g \star h) \star k = g \star (h \star k)$
- Existe um elemento identidade $e \in G$ para $\star$, isto é, $\exists e \in G, \forall g \in G, e \star g = g \star e = g$
- Para todo elemento $g \in G$ existe um elemento inverso $h \in G$ tal que $g \star h = h \star g = e$

Definição 4.1.2. Seja $(G, \star)$ um grupo. Dizemos que G é grupo abeliano se $\star$ for uma operação comutativa em G, isto é, se $\forall g, h \in G, g \star h = h \star g$.

Proposição 4.1.1. Se $(G, \star)$ é um grupo, então

- o elemento neutro é único;
- o elemento inverso é único.

Exemplo 4.1.1. $(\mathbb{Q}^*, *)$: O conjunto dos números racionais com a operação de multiplicação usual é um grupo abeliano, pois vale o axioma da comutatividade, com o elemento neutro 1, e o inverso de um elemento $a \in \mathbb{Q}^*$ é a^{-1} em $\mathbb{Q}^*$. Como a operação nesse grupo é a multiplicação, o chamamos de grupo multiplicativo. Da mesma forma, $(\mathbb{Q}, +)$ é chamado de grupo aditivo.

Definição 4.1.3. A ordem de um grupo é o número de elementos do conjunto G. Denotamos por $|G|$. G pode ser um grupo de ordem infinita, caso G seja um grupo com um número infinito de elementos.

4.1.2 Anéis

Definição 4.1.4. Um conjunto A com as operações de adição $(+)$ e multiplicação $(*)$ é um anel se:

1. $(A, +)$ é um grupo abeliano, com o elemento neutro $0 \in A$ e, para todo $a \in A$, o elemento inverso $-a \in A$.
2. $(A, *)$ é um semigrupo, ou seja, em A valem:
   - Fechamento: $\forall a, b \in A, a * b \in A$.
   - Associatividade: $\forall a, b, c \in A, a * (b * c) = (a * b) * c$.
3. A operação $*$ é distributiva em relação a $+$, ou seja, tomando quaisquer $a, b, c \in A$,

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

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

Definição 4.1.5. Se além das propriedades citadas anteriormente, A também satisfazer as seguintes:

- Existir elemento neutro na multiplicação, ou seja $\exists 1 \in A; \forall a \in A, a * 1 = 1 * a = a$.
- A multiplicação ser uma operação comutativa, ou seja, $\forall a, b \in A, a * b = b * a$

então, A é um anel comutativo com identidade.

Definição 4.1.6. Para todo inteiro $n > 1$, definimos como anel das classes de resto módulo m, o conjunto

$$\mathbb{Z}_n = \{\overline{0}, \overline{1}, ..., \overline{n-1}\}$$

em relação às operações

$$\overline{a} + \overline{b} = \overline{a + b}, \overline{a} * \overline{b} = \overline{ab}.$$

O zero neste anel será a classe $\overline{0}$, enquanto que o oposto de um elemento $\overline{a} \in \mathbb{Z}_n$ é a classe $\overline{m - a}$.

Definição 4.1.7. Seja A um anel comutativo com unidade. Se, para este anel, vale a lei do anulamento do produto, ou seja, se uma igualdade do tipo

$$ab = 0_A$$

em que $a, b \in A$ só for possível para $a = 0_A$ ou $b = 0_A$, então se diz que A é um anel de integridade ou domínio.

Em um anel comutativo A em que não é verificada a lei do cancelamento, ou seja, há pelo menos um par de elementos $a, b \neq 0; ab = 0_A$, diz-se que a e b são divisores próprios do zero do anel.

Proposição 4.1.2. Um anel de classes de restos $\mathbb{Z}_n$ é anel de integridade se, e somente se, n é um número primo.

Demonstração. ($\Rightarrow$) Suponha n composto, então podemos encontrar inteiros a, b tal que $0 < a, b < m$ e $m = ab$. Portanto, $\overline{a}, \overline{b} \in \mathbb{Z}_n, \overline{a}, \overline{b} \neq 0$ e $\overline{ab} = \overline{m} = \overline{0}$, que corresponde ao zero do anel, o que contraria a hipótese.

($\Leftarrow$) Sabemos que $\mathbb{Z}_n$ é um anel comutativo com unidade, qualquer que seja $m > 1$. Suponhamos que $\overline{a} * \overline{b} = \overline{ab} = \overline{0}$ para algum par de elementos $\overline{a}, \overline{b} \in \mathbb{Z}_n$. Portanto, $ab = nq, q \in \mathbb{Z}$, logo, $n|ab$. Mas, como n é primo, por hipótese, então $n|a$ ou $n|b$. Mas essas relações, em termos de classe de equivalência, se traduzem por $\overline{a} = \overline{0}$ ou $\overline{b} = \overline{0}$. Ou seja, se n é primo, $\mathbb{Z}_n$ não possui divisores próprios do zero. (DOMINGUES; IEZZI, 2003) □

4.1.3 Corpos

Definição 4.1.8. Seja A um anel comutativo com identidade. Dizemos que $a \in A^*$ é inversível se existe $a^{-1} \in A$, tal que

$$a * a^{-1} = a^{-1} * a = 1$$

Se para todo $a \in A$ existir a^{-1}, dizemos que A é um corpo.

Teorema 4.1.1. Temos que $a \in \mathbb{Z}_n$ é inversível para a multiplicação se, e somente se, $mdc(a, n) = 1$.

Demonstração. Suponha que n e $1 < a < n$ são inteiros que possuem um fator primo em comum $1 < p < n$. Podemos escrever $n = p * c$ e $a = p * e$. Como $1 < p < n$, então $c = \frac{n}{p}$ também satisfaz $1 < c < n$. Por sua vez, como $1 < a < n$ por hipótese, temos que nem c, nem a são congruentes a zero módulo n. No entanto,

$$c * a \equiv c * p * e \; mod \; n.$$

Porém, $n = c * p$, então $c * p \equiv n \equiv 0 \; mod \; n$, de onde sai que

$$c * a \equiv c * p * e \equiv 0 \; mod \; n. \tag{4.1}$$

Supondo que a realmente possua um inverso a' módulo n, teríamos que $a * a' \equiv 1 \; mod \; n$. Multiplicando ambos os membros por c, obtemos $c * (a * a') \equiv c \; mod \; n$, ou seja,

$$(c * a) * a' \equiv c \; mod \; n. \tag{4.2}$$

Porém, por 4.1, $c * a \equiv 0 \ mod \ n$, de modo que $(c * a) * a' \equiv 0 * a' \equiv 0 \ mod \ n$. Comparando com a equação 4.2, obtemos que $c \equiv 0 \ mod \ n$, ou seja, n divide c. Isto é um absurdo, visto que, por hipótese, $1 < c < n$. Portanto, a não possui inverso módulo n. □

Teorema 4.1.2. O anel $\mathbb{Z}_n$ é um corpo se, e somente se, n é primo.

Demonstração. Como já foi demonstrado, se n é primo, então, $\mathbb{Z}_n$ é um anel de integridade. Como $\mathbb{Z}_n$ é finito, asseguramos que $\mathbb{Z}_n$ é um corpo. □

4.2 Geometria Algébrica

Para este trabalho, utilizaremos o conceito de curvas elípticas da geometria algébrica, que possuem aplicações na criptografia quando definidas sobre corpos finitos. (**??**)

Definição 4.2.1. Seja K um corpo ($\mathbb{R}, \mathbb{Q}, \mathbb{C}$ ou $\mathbb{F}_q$, um corpo finito de $q = p^r$ elementos, onde p é primo e $r \in \mathbb{Z}^+$), com característica diferente de 2 e 3, e seja $X^3 + aX + b$ (onde $a, b \in K$), um polinômio cúbico sem raízes múltiplas. Uma curva elíptica sobre K é o conjunto de pontos $(x, y) \in K^2$ que satisfazem:

$$y^2 = x^3 + ax + b \tag{4.3}$$

junto de um elemento chamado ponto no infinito, que pode ser denotado por O.

Observação. Seja F(x,y) = 0 uma equação implícita que define uma curva elíptica e que possui y e x como variáveis desta função em 4.3, isto é,

$$F(x, y) = y^2 - x^3 - ax - b,$$

então, um ponto (x, y) na curva é chamado "não singular"se o gradiente ∇F é não nulo neste ponto.

Pode-se mostrar que a condição para que os polinômios cúbicos à direita de 4.3 não tenham raízes múltiplas é equivalente a dizer que todos os pontos na curva são não singulares.

4.2.1 Curvas Elípticas sobre os Números Reais

Mostraremos que o conjunto de pontos de uma curva elíptica junto do ponto no infinito O forma um grupo abeliano, bastando definir o ponto no infinito e a soma de dois pontos deste grupo sobre um corpo.(FLOSE, 2011)

Por definição, o ponto no infinito é a identidade do grupo.

Definição 4.2.2. Seja E uma curva elíptica $F(x, y) = y^2 - x^3 - ax - b$ com $K = \mathbb{R}$, e sejam P e Q dois pontos em E. Define-se o oposto de P e a soma $P + Q$ de acordo com as seguintes regras:

1. Se P é um ponto no infinito, define-se $-P$ como O e $P + Q$ como Q, ou seja, O serve como identidade aditiva do grupo de pontos.

Para as próximas regras, supomos que nem P e nem Q são pontos no infinito.

2. O oposto $-P$ é dado por $-(x, y) = (x, -y)$, ou seja, a mesma coordenada x e o oposto da coordenada y de P. É possível perceber que P e $-P$ pertencerão simultaneamente à curva 4.3.

3. Se P e Q têm diferentes coordenadas em x, a reta $l = \overline{PQ}$ intercepta a curva em mais um ponto R. Define-se, então, $P + Q$ como $-R$, isto é, a imagem simétrica em relação ao eixo x deste terceiro ponto.

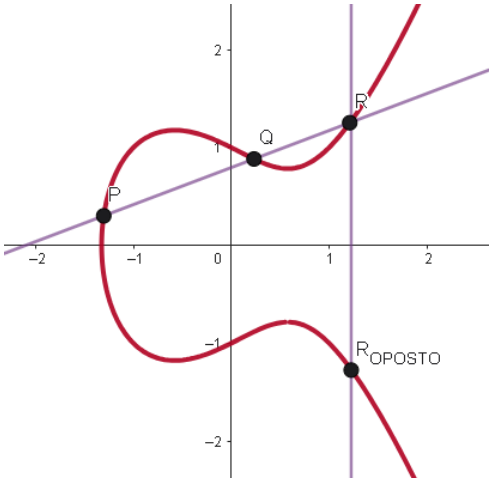

Figura 3 – $P + Q = -R$

Fonte: Elaborado pela autora.

4. Para o caso anterior, caso a reta seja tangente à curva em P (ou Q), tomamos $R = P$ (ou $R = Q$). Deste modo, teremos $P + Q = -P$ (ou $P + Q = -Q$).

5. Se $Q = -P$, então, $P + Q$ é definido como o ponto no infinito O.

6. Se $P = Q$, então, seja l a reta tangente à curva em P e seja R o único outro ponto de interseção de l com a curva. Definimos $P + Q = -R$. Se P é um ponto de inflexão, ou seja, l encontra E em um único ponto, R será igual a O.

Deste modo, percebemos que esta operação de soma é fechada em relação ao conjunto dos pontos de E unidos ao ponto O. Além disso, mostramos que existe o elemento neutro O e o elemento simétrico $-P, \forall P \in E$.

A demonstração da associatividade pode ser encontrada em Meireles (2020).

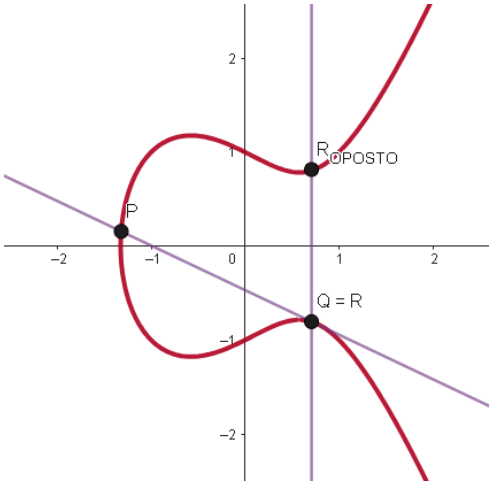

Figura 4 – Soma para o caso em que a reta é tangente a Q

Fonte: Elaborado pela autora.

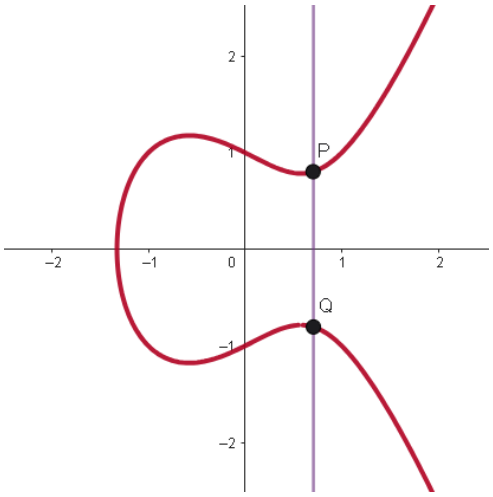

Figura 5 – Soma para o caso $P + Q = P - P = O$

Fonte: Elaborado pela autora.

4.2.2 Curvas Elípticas sobre Corpos Finitos

Para a Criptografia de Curvas Elípticas, utilizaremos os conceitos vistos anteriormente, porém aplicados em corpos finitos.

Seja $\mathbb{F}_p$ um corpo finito, com p primo. Tomando uma curva elíptica E sobre este corpo, caso esta possua uma quantidade finita de pares (x, y) com $x, y \in \mathbb{F}_p$, o corpo $E(\mathbb{F}_p)$ é finito.

Exemplo 4.2.1. Seja E a curva $y^2 = x^2 + x + 1$ sobre $\mathbb{F}_5$. Contaremos os pontos em E.

Inicialmente, faremos uma lista com todos os possíveis valores para x, considerando que estamos no corpo $\mathbb{F}_5$. Em seguida, calculamos $x^3+x+1 \ mod \ 5$ e suas raízes quadradas y dentro de $\mathbb{F}_5$. Adicionaremos, ainda, o elemento neutro (o ponto no infinito). Isso nos dará os pontos em E.

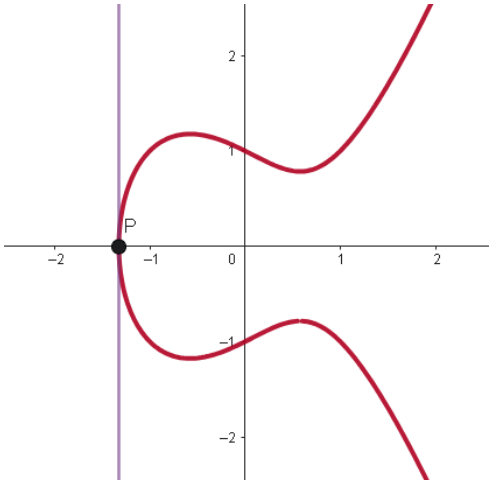

Figura 6 – Caso em que P é ponto de inflexão

Fonte: Elaborado pela autora.

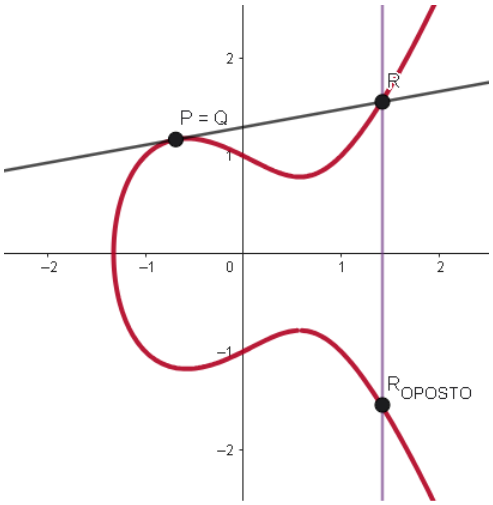

Figura 7 – Soma para o caso $P + Q = 2P = -R$

Fonte: Elaborado pela autora.

Logo, como pode ser visualizado na tabela 7, $E(\mathbb{F}_5)$ é um corpo finito de ordem 9.

Como outro exemplo, computaremos $(3,1)+(2,4)$ em E. A inclinação da reta que passará por estes pontos é dada por

$$\frac{4-1}{2-3} \equiv 2 \; mod \; 5,$$

e, então, a reta será

$$y = 2(x-3)+1 = 2x-5 \equiv 2x \; mod \; 5.$$

Substituindo em $y^2 = x^3 + x + 1$, temos $x^3 - 4x^2 + x + 1 = 0$.

Pela propriedade do coeficiente do x^2, sabemos que as raízes somam 4. No entanto, já conhecemos as raízes 3 e 2, então, sabemos que a raíz restante é $x = 4$.

Tabela 7 – Pontos em $E(\mathbb{F}_5)$ (MEIRELES, 2020)

x	$x^3 + x + 1 \; mod \; 5$	y	Pontos
0	1	± 1	(0,1),(0,4)
1	3	-	-
2	1	± 1	(2,1),(2,4)
3	1	± 1	(3,1),(3,4)
4	4	± 2	(4,2),(4,3)
∞		∞	∞

Como $y = 2x$, temos que $y \equiv 3$. Refletindo através do eixo x, temos que

$$(3, 1) + (2, 4) = (4, -3) = (4, 2).$$

5 Criptografia de Curvas Elípticas

5.1 Método de funcionamento da ECC

5.1.1 Multiplicação de pontos

Sabendo que a curva elíptica E é um grupo aditivo, as operações que utilizaremos para a nossa "função de mão única"não serão exponenciais, como na Criptografia RSA, mas sim, multiplicativas.

Neste caso, a analogia de elevar para a k-ésima potência em $\mathbb{F}_p$ é o mesmo que multiplicar um ponto $P \in E$ por um inteiro k.

Esta multiplicação, no entanto, não é feita somando o ponto a si mesmo um número k de vezes, mas sim, modificando a operação para que apenas seja necessário realizar "dobragens"do ponto, bem como algumas adições.

Exemplo 5.1.1. Para encontrar $100P$, não é necessário realizar 100 operações, pois é possível escrever

$$100P = 2(50P)$$
$$100P = 2(2P + 48P)$$
$$100P = 2(2P + 2(24P))$$
$$100P = 2(2P + 2(2(12P)))$$
$$100P = 2(2P + 2(2(2(6P))))$$
$$100P = 2(2P + 2(2(2(2(3P)))))$$
$$100P = 2(2P + 2(2(2(2(2P + P)))))$$
$$100P = 2(2(P + 2(2(2(P + 2P)))))$$

Portanto, apenas será necessário realizar 6 multiplicações e 2 adições de pontos para encontrarmos o valor de $100P$.

5.1.2 Escolhendo os parâmetros

Os parâmetros escolhidos para a implementação da ECC são:

- A equação (ou seja, os valores para a e b);

- O valor p do corpo, que deverá ser um número primo;
- O ponto base para os cálculos;
- A ordem do ponto base (que será uma função dos outros parâmetros, e é recomendável que seja preferencialmente grande).

Lembramos que, a chave pública, neste caso, será a multiplicação entre o ponto base e a chave privada.

Exemplo 5.1.2 (Bitcoin)**.** Uma das mais conhecidas aplicações atuais da ECC é o Bitcoin, uma criptomoeda que utiliza a criptografia para controlar o registro de transações da unidade monetária, por meio da atribuição de assinaturas (ULRICH, 2014). Deste modo, torna-se uma moeda independente, que não necessita de bancos ou outras instituições para seu funcionamento.

O Bitcoin utiliza números grandes para garantir a segurança de suas transações, bem como todas as aplicações reais da Criptografia de Curvas Elípticas.

A seleção de parâmetros para o Bitcoin é conhecida como *secp256k1*, e os parâmetros utilizados são:

- Equação: $y^2 = x^3 + 7$
- $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$ = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F
- Ponto base: 04 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8
- Ordem: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141

Perceba como, mesmo todos os valores utilizados para a criptografia que segura o Bitcoin sendo de público acesso, a criptomoeda consegue manter sua segurança pela complexidade em conseguir "quebrar"o código com força bruta.

Na ECDSA, a chave privada é escolhida aleatoriamente entre o número 1 e a ordem. No contexto de corpos finitos, existem algumas equações que podem ser utilizadas para representar a soma de dois pontos e a duplicação de um ponto.

Para a soma $R = P + Q$, com $P = (P_x, P_y)$ e $Q = (Q_x, Q_y)$, $R = (R_x, R_y)$ pode ser definida como

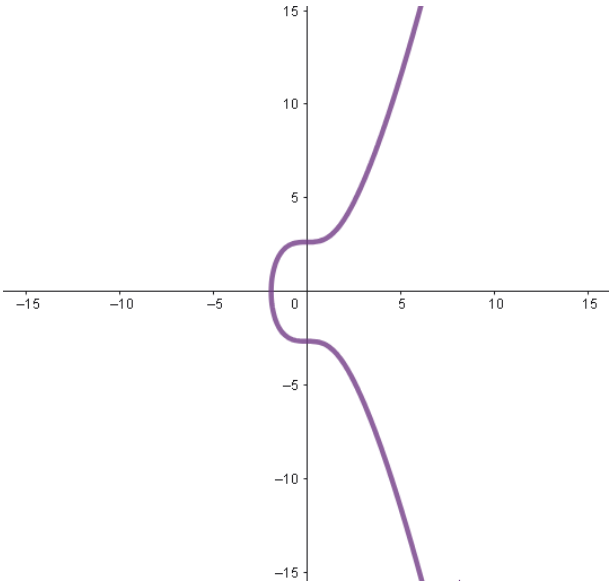

Figura 8 – Curva utilizada para a execução do Bitcoin

Fonte: Elaborado pela autora.

$$\begin{cases} c = \frac{Q_y - P_y}{Q_x - P_x} \\ R_x = c^2 - P_x - Q_x \\ R_y = c(P_x - R_x) - P_y \end{cases}$$

Além disso, para a duplicação do ponto P para encontrar R, temos

$$\begin{cases} c = \frac{3P_x^2 + a}{2P_y} \\ R_x = c^2 - 2P_x \\ R_y = c(P_x - R_x) - P_y \end{cases}$$

Exemplo 5.1.3. Queremos encontrar a chave pública que corresponde aos seguintes parâmetros:

- Equação: $y^2 = x^3 + 7$, ou seja, $a = 0, b = 7$
- $p = 67$
- Ponto base: $(2, 22)$
- Ordem: 79
- Chave privada: 2

Como temos uma chave privada de valor 2, tudo o que precisaremos fazer é duplicar o valor de $(2, 22)$, para encontrar o ponto correspondente.

Primeiro, encontramos o valor de c:

$$c = \frac{3 * 2^2 + 0}{2 * 22} \; mod \; 67$$
$$c = \frac{3 * 4}{44} \; mod \; 67$$
$$c = \frac{12}{44} \; mod \; 67$$

Como, no corpo $\mathbb{F}_{\not\triangleleft\not\triangleright}$, temos que $44^{-1} = 32$, obtemos:

$$c = 12 * 32 \; mod \; 67$$
$$c = 384 \; mod \; 67$$
$$c = 49$$

Com o valor de c, agora, encontraremos as coordenadas de R:

$$R_x = (49^2 - 2 * 2) \; mod \; 67$$
$$R_x = (2401 - 4) \; mod \; 67$$
$$R_x = 2397 \; mod \; 67$$
$$R_x = 52$$

$$R_y = (49 * (2 - 52) - 22) \; mod \; 67$$
$$R_y = (49 * (-50) - 22) \; mod \; 67$$
$$R_y = (-2450 - 22) \; mod \; 67$$
$$R_y = -2472 mod 67$$
$$R_y = 7$$

Portanto, o valor correspondente à chave pública será o ponto $(52, 7)$.

5.1.3 Assinatura

O processo de assinatura também é parte fundamental dos criptossistemas mencionados. É a partir dele que, além de um destinatário da mensagem conseguir interpretá-la, também conseguirá comprovar quem foi o remetente original da mesma.

Tomando G como o ponto base, n como a sua ordem, z a mensagem a ser assinada e d a chave privada, a escolha para o par (r, s) que corresponde à assinatura é dada da seguinte forma:

1. Escolha um inteiro k entre 1 e $n-1$
2. Calcule o ponto $(x, y) = k * G$
3. Encontre $r = x \ mod \ n$. Se $r = 0$, retorne ao passo 1.
4. Encontre $s = k^{-1}(z + r * d) \ mod \ n$. Se $s = 0$, retorne ao passo 1.
5. A assinatura será o par (r, s).

Com a mensagem devidamente assinada, precisaremos de um método para comprovar se a assinatura é realmente válida, pois esta poderia ter sido forjada por algum invasor do meio.

Para verificarmos, utilizamos os seguintes passos:

1. Verifique que r e s estão entre 1 e $n-1$
2. Calcule $w = s^{-1} \ mod \ n$
3. Calcule $u = z * w \ mod \ n$
4. Calcule $v = r * w \ mod \ n$
5. Calcule $(x, y) = uG + vP_A$
6. Verifique que $r = x \ mod \ n$. A assinatura será invalida se não for.

Exemplo 5.1.4. O dado que assinaremos será $z = 17$. Nosso ponto base será $G = (2, 22)$ com a chave privada $d = 2$ e a ordem $n = 79$ novamente.

Primeiramente, encontraremos o par (r, s) que assina a mensagem.

1. Escolheremos o número aleatório $k = 3$
2. Calcularemos o ponto $(x, y) = k * G$, ou seja, $(x, y) = 3 * (2, 22)$.

   Mas $(x, y) = 3G = 2G+G$ e já calculamos $2G = (52, 7)$, logo $(x, y) = (52, 7)+(2, 22)$.

   E, com a fórmula anterior, encontraremos

   $$(x, y) = (62, 63)$$

3. Calcularemos $r = x \ mod \ n$, ou seja, $r = 62 \ mod \ 79$.

   Assim, encontramos $r = 62$. Como $r \neq 0$, continuaremos.

4. Calcularemos $s = \frac{z+r*d}{k}\ mod\ n$.

   Teremos que $s = \frac{17+62*2}{3}\ mod\ 79$

$$s = \frac{17+124}{3}\ mod\ 79$$
$$s = \frac{141}{3}\ mod\ 79$$
$$s = 47\ mod\ 79$$
$$s = 47$$

5. Assim, a assinatura será o par $(62, 47)$.

Encontrado o par (r, s), verificaremos a sua veracidade.

1. Tanto $r = 62$ quanto $s = 47$ são menores do que 78, então está verificado.
2. Calcularemos $w = s^{-1}\ mod\ n$

$$w = 47^{-1}\ mod\ 79 = 37$$

3. Calcularemos $u = z * w\ mod\ n$

$$u = 17 * 37\ mod\ 79 = 629\ mod\ 79 = 76$$

4. Calcularemos $v = r * w\ mod\ n$

$$v = 62 * 37\ mod\ 79 = 2294\ mod\ 79 = 3$$

5. Calcularemos o ponto $(x, y) = uG + vQ$

$$uG = 76G = (62, 4)$$
$$vQ = 3Q = Q + 2Q = (11, 20)$$

   Então, $(x, y) = (62, 63)$

6. Por fim, verificamos que $r = x\ mod\ n$.

   Isto é confirmado, pois, $62 = 62\ mod\ 79$.

   Logo, a assinatura é verificada.

5.2 Fundamentação para seu funcionamento

5.2.1 O uso de um ponto secreto S

Primeiro, explicaremos brevemente como ocorre o funcionamento do algoritmo da ECC em si. Para isso, utilizaremos a comutatividade da multiplicação escalar.

Supomos que Alice possua uma chave privada a e uma chave pública $P_A = a \times G$ (sendo G o ponto gerador público) e Bob possua, da mesma forma, as chaves b e P_B. Caso Bob queira enviar uma mensagem para Alice, ele deverá utilizar sua chave privada b e a chave pública de Alice P_A, computando um ponto secreto S, desta forma:

$$S = b \times P_A$$

Repare que, o que Bob está fazendo, consiste, no fundo, em realizar as seguintes operações:

$$S = b \times (a \times G)$$

Alice, por sua vez, computará um ponto secreto S', que será obtido pela multiplicação da sua chave privada com a chave pública de Bob.

$$S' = a \times P_B$$

Perceba, também, que esta operação pode ser escrita da seguinte forma:

$$S' = a \times (b \times G)$$

É possível notar, portanto, que pela associatividade e comutatividade das operações, teremos que $S = S'$.

$$S = b \times (a \times G) = (b \times a) \times G = (a \times b) \times G = S'$$

Deste modo, tanto Alice como Bob possuem a mesma chave secreta S, que não é possível um usuário do meio inseguro conseguir interceptar.

Este problema pode, ainda, ser pensado da seguinte maneira: dados G, aG e bG, pela complexidade dos objetos, torna-se de grande dificuldade descobrir abG.

Perceba que, com isto, provamos que Alice e Bob possuem acesso à mesma chave secreta, mas nada dissemos sobre encriptação e decriptação de mensagens enviadas por eles. Isso se deve pelo fato de que a ECC sozinha não provê um método de encriptação,

necessitando de algum tipo de esquema híbrido, em que, no final, a criptografia é realizada de modo simétrico. Isso é possível utilizando alguma informação obtida de S, como, por exemplo, utilizar sua coordenada x como chave. Um exemplo amplamente utilizado é a ECDH (Elliptic Curve Diffie-Hellman).

5.2.2 Assinatura e Verificação

Por sua vez, para a explicação da assinatura e sua verificação, estaremos supondo que Alice esteja assinando.

Lembramos inicialmente que, ao final da verificação, constatamos que

$$P = uG + vP_A,$$

onde P_A é a chave pública de Alice.

No entanto, como $P_A = d_A G$, logo,

$$P = uG + vd_A G.$$

Dessa forma, podemos isolar G, de modo que

$$P = (u + vd_A)G.$$

Mas, como $u = zw$ e $v = rw$, ignorando, por ora, as parcelas de $mod\ n$, temos que

$$P = (wz + wrd_A)G.$$

E, assim, isolamos w, para que tenhamos

$$P = w(z + rd_A)G.$$

Mas, sabemos ainda que $k = w(z + rd_A)\ mod\ n$. Dessa forma, retornamos ao valor original de

$$P = kG.$$

Assim, confirmamos que a assinatura está verificada.

6 Implementação

A implementação da Criptografia de Curvas Elípticas foi feita utilizando a linguagem de programação Python (Python Software Foundation, 2025). Para uma utilização mais simples, o código foi gerado no software online Google Colab (Google Research, 2017).

A biblioteca Python utilizada foi a biblioteca *ecies*. A sigla ECIES refere-se ao Elliptic Curve Integrated Encryption Scheme (ou seja, Esquema de Criptografia Integrada de Curva Elíptica), que corresponde a um dos sistemas híbridos de criptografia mencionados anteriormente. Esta biblioteca ajuda a gerar, com grande facilidade, chaves públicas e privadas baseadas em uma curva elíptica específica.

A curva padrão do ECIES é a mesma curva padrão para o Bitcoin que já mencionamos anteriormente ($y^2 = x^3 + 7$), chamada de "secp256k1".

As funções da biblioteca *ecies* utilizadas para o funcionamento do código são as seguintes:

- PrivateKey: Gera uma chave privada, ou seja, um ponto da curva elíptica selecionada.
- public_key: Com base na chave privada previamente escolhida, gera uma chave pública.
- encrypt: Função que recebe a chave pública e os dados a serem criptografados e devolve, em bytes, os dados encriptados.
- decrypt: Função que recebe a chave privada e os dados a serem decriptografados e devolve, em bytes, os dados decriptados.

Abaixo, é demonstrado um exemplo de chave privada e chave pública (ambas com valores hexadecimais) gerados pela curva secp256k1.

```
Chave privada: 1af3d2ee465c84a980fa3a31b0cec23c184c89409ee85
a9e58d51607afe9f622
Chave publica: 04bd2dc0415704d054cba9e03da05f0327325a2cc3b
006715c4176f6648f4cee9b880fa5e96e56f948fedec3f77eb9d268be1
b79572e345d56c3c557c52dda48ce
```

Além disso, foram utilizadas algumas funções da biblioteca *base64*:

- b64encode: Codifica os dados para base64, retornando em bytes os dados codificados em formato ASCII.
- b64decode: Decodifica os dados que estão em base64, retornando os bytes originais.

O código completo pode ser visualizado no anexo A.

6.1 Base 64

A codificação em base 64 é projetada para representar sequências arbitrárias de grupos de oito caractéres, de modo que não necessariamente seja legível para outros usuários.

São utilizados um subconjunto de 65 caractéres do ASCII, o que permite que cada caractér possua seis bits. O 65º caractér é o "=", conhecido como *padding*. Ele representa uma função especial que ajuda o processo de codificação e decodificação da base64 (JOSEFSSON, 2006).

O processo de codificação funciona ao representar grupos de 24 bits de entrada como cadeias de 4 caracteres codificados. Avançando da esquerda para a direita, um grupo de entrada de 24 bits é formado pela concatenação de três grupos de entrada de 8 bits. Esses 24 bits, então, são então tratados como 4 grupos concatenados de 6 bits, cada um dos quais é traduzido em um único caractere no alfabeto Base 64.

Valor	**Cod.**	**Valor**	**Cod.**	**Valor**	**Cod.**	**Valor**	**Cod.**
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v	(pad)	=
14	O	31	f	48	w		
15	P	32	g	49	x		
16	Q	33	h	50	y		

Tabela 8 – Alfabeto Base 64 conforme RFC 4648 (JOSEFSSON, 2006)

Se menos de 24 bits estiverem disponíveis ao final dos dados, bits com valor zero serão adicionados à direita para formar um número inteiro com 6 bits. Este preenchimento é feito usando o caractér "=".

Como toda entrada em base 64 é um número inteiro múltiplo de oito, apenas os seguintes casos podem ocorrer:

- A última parcela de entrada possui exatamente 24 bits; neste caso, a última unidade de saída será um múltiplo de 4 caracteres, sem necessidade de padding.
- A última parcela possui exatamente 16 bits; nesse caso, a saída final possui 3 caracteres seguidos de um padding.
- A última parcela possui exatamente 8 bits; nesse caso, a saída possui dois caracteres seguidos de dois paddings.

Dados	**Base64**
M	TQ==
MA	TUE=
MAT	TUFU
MATE	TUFURQ==
MATEM	TUFURU0=
MATEMÁ	TUFURU3DgQ==
MATEMÁT	TUFURU3DgVQ=
MATEMÁTI	TUFURU3DgVRJ
MATEMÁTIC	TUFURU3DgVRJQw==
MATEMÁTICA	TUFURU3DgVRJQ0E=

Tabela 9 – Exemplo de letras codificadas em base64

Fonte: Elaborado pela autora.

A utilização da base64 dentro do algoritmo garante que o destinatário da mensagem consiga decriptá-la da maneira correta e com segurança, de modo que não serão corrompidos durante o processo.

6.2 Alguns resultados

A implementação prática do algoritmo baseado em Criptografia de Curvas Elípticas permitiu analisar de forma comparativa seu desempenho em relação ao RSA.

Os testes foram feitos utilizando diferentes tipos de arquivos, como arquivo de texto, imagem em jpeg, arquivo em pdf e outros. O primeiro teste foi feito utilizando o poema "Canção do Exílio", de Gonçalves Dias. O tamanho do poema é de 695 bytes, e ele demorou 0.0036s para ser encriptado e decriptado pela ECC, enquanto o mesmo processo demorou 0.0719s na RSA, ou seja, cerca de 20 vezes a mais.

Listing 6.1 – Poema Canção do Exílio (DIAS, 1957)

```
Minha terra tem palmeiras,
Onde canta o Sabiá;
As aves, que aqui gorjeiam,
Não gorjeiam como lá.

Nosso céu tem mais estrelas,
Nossas várzeas têm mais flores,
Nossos bosques têm mais vida,
Nossa vida mais amores.

Em  cismar, sozinho, à noite,
Mais prazer eu encontro lá;
Minha terra tem palmeiras,
Onde canta o Sabiá.

Minha terra tem primores,
Que tais não encontro eu cá;
Em cismar   sozinho  , à n o i t e
Mais prazer eu encontro lá;
Minha terra tem palmeiras,
Onde canta o Sabiá.

Não permita Deus que eu morra,
Sem que eu volte para lá;
Sem que disfrute os primores
Que não encontro por cá;
Sem quinda aviste as palmeiras,
Onde canta o Sabiá.
```

É claro que, apesar de a razão de um tempo pelo outro ter sido um número substancialmente grande, ainda é possível argumentar que, para usos diários, a RSA não apresentaria diferenças tão significativas, pois encontram-se na casa dos centésimos de segundo.

Por isso, foram realizados outros testes, utilizando arquivos com diversos tamanhos.

Na tabela 10, segue alguns outros dados que foram testados no código, bem como seus respectivos tamanhos e o tempo de demora em cada um dos algoritmos.

Os testes realizados demonstraram que a ECC apresentou um comportamento significativamente mais eficiente, especialmente à medida que o volume de dados aumentava.

Arquivo	**Tamanho**	**ECC**	**RSA**
Canção do Exílio (txt)	695 bytes	0.0036 s	0.0719 s
Código em Python (py)	7.313 bytes	0.0058 s	0.1853 s
Planilha dos alunos da Matemática Aplicada (csv)	112.879 bytes	0.0095 s	0.5547 s
Monalisa (jpeg)	317.486 bytes	0.0113 s	1.515 s
Me at the zoo (mp4, 240p)	791.367 bytes	0.0200 s	3.6413 s

Tabela 10 – Comparação de desempenho entre RSA e ECC para diferentes arquivos

Fonte: Elaborado pela autora.

Enquanto o RSA mostrou tempos de execução progressivamente maiores conforme o tamanho das mensagens crescia, a ECC manteve um desempenho mais estável e responsivo. Essa diferença tornou-se ainda mais evidente em cenários envolvendo operações com muitos bytes, nos quais o RSA se mostrou substancialmente mais lento. Estas razões podem ser observadas na tabela 11.

Algo que pode ser percebido também é que, para o arquivo de texto, sua versão criptografada tornou-se um arquivo com apenas 9 linhas, formado, em sua maioria, por símbolos que o computador não consegue escrever. Enquanto isso, para os demais arquivos, não foi possível observar a versão criptografada. Esses fatores, juntos, comprovam a dificuldade de algum usuário aleatório interceptar mensagens.

Arquivo	**Razão entre RSA e ECC**
txt	19.97
py	31.94
csv	58.39
jpeg	134.07
mp4	182.06

Tabela 11 – Razão entre os algoritmos testados para cada arquivo

Fonte: Elaborado pela autora.

Com os dados obtidos, verificou-se que este aumento segue um padrão que se aproxima mais do potencial. Essa tendência pôde ser observada pelo método de regressão linear, cujo resultado pode ser visualizado na figura 9. A curva que apresentou maior similaridade aos dados foi

$$y = 2.223x^{0.312},$$

que obteve coeficiente de determinação $R^2 = 0.9077$. O código utilizado para essa análise pode ser visto no anexo A.

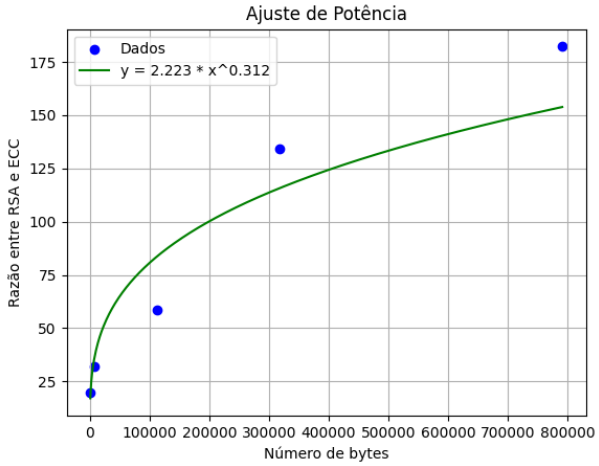

Figura 9 – Curva que aproximou-se melhor dos dados

Fonte: Elaborado pela autora.

7 Conclusão

No presente trabalho, foi possível compreender a grande importância do desenvolvimento da criptografia, em especial, no mundo atual, em que grande parte das interações e transações efetuadas são realizadas por meios inseguros. A crescente digitalização da sociedade tornou indispensável o uso de ferramentas criptográficas capazes de garantir privacidade, integridade e autenticidade das informações trocadas, o que evidencia a relevância do tema estudado.

Estudamos o funcionamento destas criptografias, desde suas fundamentações matemáticas, baseadas na Álgebra e na Aritmética, até exemplos práticos e verificações de seus processos. Esse percurso permitiu estabelecer uma visão clara tanto das propriedades teóricas quanto da aplicabilidade real dos métodos.

Com base nisso, pode-se concluir que a Criptografia de Curvas Elípticas apresenta vantagens significativas em relação à Criptografia RSA, amplamente utilizada até os dias de hoje. Esta vantagem se deve à necessidade de um comprimento menor da chave de criptografia para atingir níveis equivalentes de segurança, além de um tempo computacional inferior na execução de suas operações. Esses fatores tornam a ECC particularmente adequada para dispositivos com recursos limitados.

Percebeu-se, ainda, que não só a RSA atua de maneira mais lenta, como também sofre um aumento proporcionalmente maior dessa lentidão à medida que o tamanho do arquivo a ser criptografado cresce. Esse comportamento deixa evidente a diferença de eficiência entre os dois métodos, reforçando a superioridade da ECC em cenários nos quais o desempenho é fator crucial. Ainda foi verificado que o aumento segue um padrão que se aproxima mais do potencial.

Os resultados obtidos reforçam a relevância e o potencial da Criptografia de Curvas Elípticas no cenário atual, especialmente diante da crescente demanda por segurança e eficiência nos sistemas digitais. Ainda que a RSA continue amplamente utilizada, a tendência observada indica um movimento gradual em direção à adoção de métodos como a ECC, que atendem melhor às necessidades tecnológicas atuais.

No entanto, para o futuro da criptografia mundial, é importante mencionar que os estudos a respeito da computação quântica trazem grande fragilidade em relação aos métodos de criptografia conhecidos. O algoritmo de Shor (SHOR, 1994) é conhecido por conseguir vencer a dificuldade de encontrar fatores primos de números grandes, o que comprometeria a segurança de sistemas criptográficos.

Isso acontece pois, na computação quântica, não temos mais bits, representados

pelos valores de 0 e 1, mas sim qubits, que encontram-se em uma combinação linear entre os estados 0 e 1, com coeficientes complexos (FREITAS, 2010). Em um computador quântico, para fatorar um número n, o tempo necessário seria de apenas $\log n$.

Referências

BARKER, E. *Recommendation for Key Management, Part 1: General*. [S.l.]: National Institute of Standards and Technology, 2020. Citado na página 15.

BRAGA, B. da R. *Análise de Frequência de Línguas*. 2003. Citado 2 vezes nas páginas 8 e 12.

BROWN, D. *Fortaleza Digital*. [S.l.]: St. Martin's Press, 1998. Citado na página 4.

COSTA, C.; FIGUEIREDO, L. M. *Introdução à Criptografia*. Rio de Janeiro - RJ: Centro de Estudos Pessoal, 2006. 106 p. ISBN 85-7648-303-3. Citado na página 11.

COUTINHO, S. C. *Criptografia*. Rio de Janeiro, RJ: Associação Instituto Nacional de Matemática Pura e Aplicada - IMPA, 2016. 217 p. ISBN 978-85-244-0340-8. Citado na página 12.

DIAS, G. Poesias completas. In: _____. [S.l.]: Saraiva, 1957. cap. Canção do Exílio. Citado na página 53.

DIFFIE, W.; HELLMAN, M. E. New directions in cryptography. *IEEE Transactions on Information Theory*, v. 22, n. 6, 1976. Citado na página 14.

DOMINGUES, H. H.; IEZZI, G. *Álgebra Moderna*. São Paulo - SP: Atual Editora, 2003. 368 p. Citado 2 vezes nas páginas 34 e 36.

FLOSE, V. B. S. *Criptografia e Curvas Elípticas*. Dissertação (Mestrado) — Universidade Estadual Paulista "Júlio de Mesquita Filho", 2011. Citado na página 37.

FREITAS, A. X. *Algoritmo de Shor e sua aplicação à fatoração de números inteiros*. Dissertação (Mestrado) — Universidade Federal de Minas Gerais, 2010. Citado na página 57.

GAUSS, C. F. *Disquisitiones arithmeticae*. [S.l.]: Lipsiae : In commiss, 1801. Citado na página 21.

Google Research. 2017. /urlhttps://colab.google/. Acesso em: 01/07/2025. Citado na página 50.

HEFEZ, A. *Elementos de Aritmética*. [S.l.]: Sociedade Brasileira de Matemática, 2006. 169 p. ISBN 978-85-85818-25-5. Citado 4 vezes nas páginas 18, 19, 21 e 25.

HELLMAN, M. E. An overview of public key cryptography. *IEEE COMMUNICATIONS SOCIETY MAGAZINE*, 1978. Citado na página 28.

JOSEFSSON, S. *The Base16, Base32, and Base64 Data Encodings*. RFC Editor, 2006. RFC 4648. (Request for Comments, 4648). Disponível em: <https://www.rfc-editor.org/info/rfc4648>. Citado 2 vezes nas páginas 8 e 51.

KOBLITZ, N. Elliptic curve cryptosystems. *Mathematics of Computation*, v. 48, n. 177, p. 203–209, 1987. Citado na página 14.

MEIRELES, T. A. B. *Curvas Elípticas e Criptografia*. 2020. Citado 3 vezes nas páginas 8, 38 e 41.

MILLER, V. Use of elliptic curves in cryptography. *Advances in Cryptology - CRYPTO '85*, p. 417 – 426, 1986. Citado na página 14.

POULTER, W.; KULP, J. The adfgvx cipher. *Lakehead University Cryptography Class*, 2017. Citado na página 13.

Python Software Foundation. *What is Python? Executive Summary*. 2025. <https://www.python.org/doc/essays/blurb/>. Acesso em: 01/07/2025. Citado na página 50.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtainingdigital signatures and public-key cryptosystems. *Communications of the ACM*, v. 2, 1978. Citado 2 vezes nas páginas 14 e 28.

SHOR, P. W. Algorithms for quantum computation: discrete logarithms and factoring. In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1994. (SFCS '94), p. 124–134. Citado na página 56.

SMART, N. P. The enigma machine. In: _____. *Cryptography Made Simple*. Cham: Springer International Publishing, 2016. p. 133–161. ISBN 978-3-319-21936-3. Disponível em: <https://doi.org/10.1007/978-3-319-21936-3_8>. Citado na página 13.

ULRICH, F. *BITCOIN - A MOEDA NA ERA DIGITAL*. [S.l.]: Instituto Ludwig Von Mises Brasil, 2014. ISBN 978-85-8119-076-1. Citado na página 43.

Anexos

ANEXO A – Códigos

Listing A.1 – Código Python da Implementação da ECC (e RSA)

```
#Dando upload no arquivo e salvando ele em uma variável

from google.colab import files
uploaded = files.upload()
arquivoCaminho = list(uploaded.keys())[-1]

######### ECC

#Importar bibliotecas
from ecies import encrypt, decrypt
import base64, os
from ecies.keys import PrivateKey
import time

tempoInicial = time.perf_counter()

#Chaves privada e pública
secp_k = PrivateKey("secp256k1")
#secp256k1 = curva utilizada para o bitcoin (y^2 = x^3 + 7)
privadaHex = secp_k.to_hex()
publicaHex = secp_k.public_key.to_hex()
#chaves pública e privadas convertidas para hexadecimal

#Dividir o caminho entre diretório e nome do arquivo
diretorio, nomeArquivo = os.path.split(arquivoCaminho)
arquivoEnc = os.path.join(diretorio, f'encriptado_{nomeArquivo}')
#cria arquivo para encriptação
arquivoDec = os.path.join(diretorio, f'decriptado_{nomeArquivo}')
#cria arquivo para decriptação

#Converter em base64
dados = 0
with open(arquivoCaminho, "rb") as f:
#rb = read binary (leitura em binário)
    dados = base64.b64encode(f.read())
    #codifica esses bytes em base64
```

```
secpEncriptado = encrypt(publicaHex, dados)
#encripta os dados com a chave pública


#Abre o arquivo encriptado e escreve nele
with open(arquivoEnc,"wb") as ef:
#wb = write binary (escrever em binário)
    ef.write(secpEncriptado)

#Decriptação
secpDecriptado = decrypt(privadaHex, secpEncriptado)
#decriptar o arquivo encriptado, utilizando a chave privada

#Abre o arquivo decriptado e escreve nele
with open(arquivoDec,"wb") as df:
    df.write(base64.b64decode(secpDecriptado))
    #decodifica uma string em base64, voltando ao original

tempoFinal = time.perf_counter()


print("\nTempo necessário:",tempoFinal - tempoInicial,"segundos.")

######### RSA

#Importando bibliotecas
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.backends import default_backend

tempoInicial = time.perf_counter()

#Lendo o arquivo
with open(arquivoCaminho, "rb") as f:
    dados_rsa = f.read()

#Gerando par de chaves RSA
private_key = rsa.generate_private_key(
    public_exponent=65537,
    #expoente público, padrão mundial (primo e com formato binário simples)
    key_size=2048, #padrão moderno
    backend=default_backend()
```

```
)
public_key = private_key.public_key()

# Definindo tamanho máximo de cada partição da mensagem
hashSize = 32  # SHA-256 -> 32 bytes
tamanhoParticao = 2048 // 8 - 2 * hashSize - 2  # ~190 bytes
#2048 bits/8 = 256 bytes, o resto é requisito do algoritmo

# Dividindo a mensagem em partições
particoes = [dados_rsa[i:i+tamanhoParticao] for i in range(0,
                len(dados_rsa), tamanhoParticao)]

# Criptografando cada partição
particoesEncriptadas = []
for particao in particoes:
    particaoEncriptada = public_key.encrypt(
        particao,
        padding.OAEP( #Optimal Asymmetric Encryption Padding
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    particoesEncriptadas.append(particaoEncriptada)

# Junta todas as partições criptografados
mensagemEnc = b"".join(particoesEncriptadas)

# Salva a mensagem criptografada
arquivoEnc = os.path.join(diretorio, f"rsa_encriptado_{nomeArquivo}")
with open(arquivoEnc, "wb") as f:
    f.write(mensagemEnc)

# Descriptografa cada partição
# O tamanho de cada partição criptografada = tamanho da chave
#em bytes
tamanhoPartEnc = 2048 // 8  # 256 bytes

particoesDec = [
    private_key.decrypt(
        mensagemEnc[i:i+tamanhoPartEnc],
        padding.OAEP(
```

```
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    for i in range(0, len(mensagemEnc), tamanhoPartEnc)
]

# Junta partições descriptografadas
mensagemDec = b"".join(particoesDec)

# Salva mensagem descriptografada
arquivoDec = os.path.join(diretorio, f"rsa_decriptado_{nomeArquivo}")
with open(arquivoDec, "wb") as f:
    f.write(mensagemDec)

tempoFinal = time.perf_counter()
print("\nTempo necessário:", tempoFinal - tempoInicial, "segundos.")
```

Listing A.2 – Código Python da Regressão Não-Linear

```
import numpy as np
import matplotlib.pyplot as plt

# Dados
numero_bytes = np.array([695, 7313, 112879, 317486, 791367], dtype=float)
razao_rsa_ecc = np.array([19.97, 31.94, 58.39, 134.07, 182.06], dtype=floa

# Transformação logarítmica para regressão
log_bytes = np.log(numero_bytes)
log_razao = np.log(razao_rsa_ecc)

# Regressão linear
expoente, ln_coeficiente = np.polyfit(log_bytes, log_razao, 1)
coeficiente = np.exp(ln_coeficiente)

print("Modelo de potência: y = coeficiente * x^expoente")
print(f"Coeficiente = {coeficiente:.3f}")
print(f"Expoente = {expoente:.3f}")

# Valores ajustados
razao_ajustada = coeficiente * numero_bytes**expoente
```

```
# Calcular R quadrado
soma_residuos_quadrado = np.sum((razao_rsa_ecc - razao_ajustada)**2)
soma_total_quadrado = np.sum((razao_rsa_ecc - np.mean(razao_rsa_ecc))**2)
r_quadrado = 1 - soma_residuos_quadrado / soma_total_quadrado

print("R   =", r_quadrado)

# Curva ajustada para plotagem
bytes_fit = np.linspace(min(numero_bytes), max(numero_bytes), 500)
razao_fit = coeficiente * bytes_fit**expoente

# Plot
plt.scatter(numero_bytes, razao_rsa_ecc, label="Dados", color='blue')
plt.plot(bytes_fit, razao_fit, color='green', label=f"y = {coeficiente:.3f}
plt.xlabel("Número de bytes")
plt.ylabel("Razão entre RSA e ECC")
plt.legend()
plt.grid(True)
plt.title("Ajuste de Potência")
plt.show()
```