Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Aplicação Prática de Criptografia de Curvas Elípticas (ECC)

Fernando Lima Madeira

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Orientadora
Prof.a Dr.a Edna Dias Canedo

Brasília
2021

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Aplicação Prática de Criptografia de Curvas Elípticas (ECC)

Fernando Lima Madeira

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Prof.a Dr.a Edna Dias Canedo (Orientadora)
CIC/UnB

Prof. Dr. João José Costa Gondim
UnB

Prof. Dr. Marcos Fagundes Caetano
UnB

Prof. Dr. Wilson Henrique Veneziano
Coordenador do Curso de Computação — Licenciatura

Brasília, 18 de Outubro de 2021

Dedicatória

Dedico este trabalho a todas as pessoas mais afetadas pela pandemia de Covid-19. Tanto aquelas que perderam entes queridos, como as que perderam a sua fonte de sustento, todos os que precisaram mudar completamente a forma de viver começando do zero.

Dedico também à Ciência e seus profissionais dedicados que buscam meios comprovados cientificamente de salvar vidas mesmo em tempos de muito julgamento inadequado e pouco recurso. Que a Ciência e a produção científica de alta qualidade seja fonte inesgotável de melhoria da qualidade de vida das pessoas, principalmente as mais necessitadas.

Agradecimentos

Agradeço à toda a minha família, base de tudo pra mim, principalmente aos mais próximos, que me acompanham há muito tempo. Fonte das maiores alegrias e responsáveis diretos pelas minhas conquistas. A gente ri, a gente chora, mas a gente tá junto até o fim. Não vou listar nomes, mas sem vocês eu nunca chegaria até aqui.

Agradeço à UnB, ao IE e ao CIC pelas maravilhosas experiências que tive em todo esse processo, que não penso em parar. É o fim de um ciclo, em breve outro se inicia...

Agradeço à professora Edna Canedo, por me acolher junto com o tema que escolhi, pela orientação e direcionamento do trabalho. Agradeço também aos professores João Gondim e Marcos Caetano pelos comentários e boas sugestões de ajustes e correções para a versão final desse trabalho.

Agradeço a todos os professores, colegas e profissionais com quem pude ter contato e troca de experiências nas instituições de ensino que passei. As lembranças do convívio ficarão marcadas na minha memória.

Agradeço aos meus colegas de trabalho no Ministério da Economia e na Caixa Econômica Federal pelos ensinamentos do dia-a-dia, pelo companheirismo e pela paciência.

Agradeço aos amigos de longa data por momentos de lazer e pelas boas palavras em momentos chave da vida.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

A informação tem se tornado cada vez mais valiosa para a sociedade. A segurança envolvida deve evoluir-se ao longo do tempo sob pena de não proteger mais e deixar vulnerável os agentes envolvidos. A criptografia é uma ferramenta essencial para buscar tal segurança. Como existem muitas técnicas pretende-se mostrar a aplicação da técnica de Criptografia de Curvas Elípticas (ECC) em um exercício de cifragem de dados, um exemplo de situação prática em segurança da informação. A metodologia usada é o levantamento associado a um estudo de caso. A técnica da criptografia RSA, considerada como mais popular e conhecida, é aplicada e comparada com a aplicação de ECC. A linguagem R foi o ambiente usado para as implementações. Conceitos matemáticos de aritmética e álgebra foram essenciais para atingir os objetivos. Os resultados mostram que a grande vantagem da ECC sobre a RSA é a maior complexidade para gerar chaves e para cifrar. A análise assintótica dos algoritmos foi usada para mostrar esse resultado. Conclui-se que a ECC tem complexidade maior e, desse modo, fornece maior segurança que RSA tomando como base uma chave do mesmo tamanho.

Palavras-chave: Segurança, Criptografia, Chave Pública, Curvas Elípticas, Complexidade

Abstract

Information has become increasingly valuable to society. The security involved must evolve over time, otherwise it will no longer protect and leave the agents involved vulnerable. Encryption is an essential tool to pursue such security. As there are many techniques, it is intended to show the application of the Elliptic Curve Cryptography (ECC) technique in a data encryption exercise, an example of a practical situation in information security. The methodology used is the survey associated with a case study. The RSA encryption technique, considered to be the most popular and known, is applied and compared with the application of ECC. The R language was the environment used for the implementations. Mathematical concepts of arithmetic and algebra were essential to achieve the goals. The results show that the great advantage of ECC over RSA is the greater complexity to generate keys and to encrypt. Asymptotic analysis of the algorithms was used to show this result. It is concluded that ECC has greater complexity and thus provides greater security than RSA based on a key of the same size.

Keywords: Security, Cryptography, Public Key, Elliptic Curves, Complexity

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AES Advanced Encryption Standard.

API Application Programming Interface.

ASCII Código Padrão Americano para o Intercâmbio de Informação.

DES Data Encryption Standard.

ECC Criptografia de Curvas Elípticas.

ECDSA Elliptic Curve Digital Signature Algorithm.

IoT Internet of Things.

PLD Problema do Logaritmo Discreto.

RSA Algoritmo assimétrico proposto por Rivest, Shamir e Adleman em 1978.

Capítulo 1

Introdução

Quando se fala em proposta de segurança, supõe-se alguma ferramenta, metodologia e/ou artifício para redução de riscos. Em segurança de dados, estamos interessados em proteção de dados que tenham algum valor para alguma parte ou entidade como em [1]. Caso tais dados caiam em mãos erradas ou fiquem indisponíveis por algum período de tempo, os prejuízos podem ser bem elevados. Quanto mais essa área evolui, mais as modalidades de ataque também crescem e se disseminam. Uma modalidade bem conhecida é o ataque cibernético por meio de *ransomware*, que derruba plataformas e causa um problema chamado negação de serviço, quando os usuários desses dados ficam sem acesso a eles, [2]. Para se ter uma ideia, em 2020, o Brasil recebeu cerca de 8,4 bilhões de tentativas de ataques cibernéticos segundo o laboratório FortiGuard Labs [3].

Essa grande quantidade de tentativas de ataque foi aumentada pelo fato de muitas pessoas terem iniciado o *work from home office* em razão da pandemia, sem ter a infraestrutura e cuidados mínimos nos dispositivos utilizados para esse fim dentro de casa segundo reportagem publicada [4]. Um outro fator para justificar o aumento dos ataques foi o crescimento do tráfego global de dados na internet. Segundo [4], esse tráfego subiu 20% alcançando nível mais alto da história.

Um outro problema gerado pelos ataques são os vazamentos de dados pessoais de usuários de sistemas e serviços. Como exemplo, podemos mencionar o caso do vazamento de informações pessoais de 223 milhões de pessoas e empresas do Brasil, descoberto por autoridades em 2021. Esse foi considerado o maior vazamento de dados de brasileiros na história [5].

Segundo Kuhn [6], pode-se listar ainda uma série de outras modalidades de ataques tais como terrorismo, espionagem, sabotagem, disponibilização de conteúdo externo e incoerente com a política da organização, fraude, spam, etc. Com os casos citados e vários outros exemplos, mostra-se a necessidade de proteger os dados e sistemas contra os diversos tipos de ataques. Os principais aspectos a serem observados em segurança de

dados são o sigilo (ou confidencialidade), a integridade e a disponibilidade das informações, desde o armazenamento até a transmissão, passando pelo processamento, [6].

Mesmo com uma boa tecnologia e infraestrutura, o problema da vulnerabilidade no acesso a dados só pode ser combatido por meio de uma política de segurança que tenha impacto pelo menos na cultura de seus usuários. Ou seja, se não houver educação para que dados sensíveis não fiquem expostos, não adianta ter o melhor sistema de segurança do mundo.

Existe uma vasta quantidade de métodos sobre proteção de dados. Cifragem [7], assinatura digital [8], controle de acesso [9], controle de tráfego [10], certificação [11], entre outros, são alguns dos métodos mais tradicionais. A criptografia é usada em praticamente todos eles e possui uma vasta literatura. Dependendo do dispositivo, do serviço, do dado, do contexto, do meio de transmissão, um ou outro método é mais indicado.

Stallings [12] define uma mensagem original como *texto claro*; uma mensagem codificada como *texto cifrado* ou *criptograma* ([13]); e o processo de converter texto claro em texto cifrado como *cifragem* ou *criptografia*. Isso é feito para que a informação contida nessas cifras seja transmitida por um meio não seguro sujeito a ataques ativos e passivos. Ou seja, quando o texto cifrado chegar ao seu destino, ele passa por um processo inverso, de decriptografia ou decifragem, para que ele retorne à condição de texto claro e a transmissão da mensagem se complete.

Existem dois tipos de criptografia: a simétrica e a assimétrica. A criptografia simétrica é uma forma de criptossistema em que a cifragem e a decifragem são realizadas usando a mesma chave, [12]. A criptografia simétrica também é conhecida como criptografia convencional [12] ou como criptografia de chave privada [14]. A chave é uma informação secreta usada como entrada para o algoritmo de criptografia tanto para cifrar como para decifrar [12]. A criptografia assimétrica, por outro lado, é uma forma de criptossistema em que a cifragem e a decifragem são realizadas usando um par de chaves diferentes: uma chave pública e uma chave privada segundo [12]. A criptografia assimétrica também é conhecida como criptografia de chave pública [12]. Nesse caso, uma das duas chaves é usada para cifrar enquanto a outra é usada para decifrar.

Stallings [12] define assinatura digital como um mecanismo de autenticação que permite ao criador de uma mensagem anexar um código que atue como uma assinatura. A assinatura digital é uma aplicação de criptografia bem comum e usada para que se consiga comprovar origem, autoria e integridade em documentos transmitidos pela rede. Essa técnica permite que terceiros possam verificar a origem e a integridade da mensagem assinada por meio de uma chave de conhecimento público [14].

Para que um protocolo de criptografia assimétrica seja efetivo, deve ser inviável a partir da chave pública, obter a chave privada de um algoritmo, [15]. Isso significa que não existe

algoritmo com complexidade de tempo polinomial que consiga encontrar a chave privada a partir da chave pública usando uma estratégia de força bruta, [14]. E esse problema normalmente está associado ao *Problema do Logaritmo Discreto*, que é quando queremos obter o expoente de uma base em uma potência calculada dentro de um grupo munido de uma operação binária, como a multiplicação, [16].

A Criptografia de Curvas Elípticas (ECC) é uma forma de aplicação de criptografia assimétrica que torna o problema do logaritmo discreto mais difícil [16]. Desse modo, o tamanho da chave pode ser reduzido para manter o mesmo nível de segurança, sendo essa a maior vantagem da ECC perante outros tipos de criptografia assimétrica, [16].

1.1 Problema de pesquisa

A criptografia é um mecanismo adicional para garantir a segurança, uma vez que políticas, protocolos e culturas de segurança são elementos importantes em segurança de dados. E o que vai definir o melhor método de criptografia a ser usado envolve questões como complexidade de tempo e de espaço, assim como estratégia e custos diversos.

Lara [16] aponta que a criptografia de curvas elípticas consegue a mesma segurança que um algoritmo assimétrico RSA (proposto por Rivest, Shamir e Adleman em 1978, [14]), mas com uma redução considerável no tamanho da chave. Para ilustrar, por exemplo, para obter a segurança equivalente a um algoritmo RSA com chave de tamanho 1024 bits, a ECC precisa de uma chave com apenas 160 bits. E a tendência é que a redução seja proporcionalmente maior para chaves maiores.

O problema de pesquisa é dado pela busca pela implementação dos códigos de ambas as técnicas de forma a comparar as complexidades. Podemos também mencionar os seguintes problemas complementares que precisam ser investigados:

QP.1: Como comparar modelos criptográficos?

QP.2: Que vantagens, além da redução do tamanho da chave, podemos obter ao usar a criptografia de curvas elípticas no lugar de outras técnicas de criptografia assimétrica?

QP.3: Por que esse método não é tão difundido no Brasil?

QP.4: Quais entraves para a maior utilização de ECC no Brasil?

1.2 Objetivo

O objetivo geral deste trabalho é implementar as técnicas de criptografia assimétrica RSA e ECC, assim como evidenciar a diferença de complexidade entre elas.

1.3 Resultados Esperados

Esse trabalho pretende revisar os principais métodos de criptografia, discutir detalhes de segurança de dados, investigar o tema de Criptografia de Curvas Elípticas (ECC), explorar as possíveis aplicações, listar vantagens e desvantagens de cada método, e implementar uma aplicação prática para atender uma necessidade real das pessoas. Como resultado, espera-se obter as diferenças de complexidade entre as técnicas ECC e RSA.

1.4 Metodologia de Pesquisa

De acordo com Yin [17] um estudo de caso é uma investigação empírica que analisa um fenômeno contemporâneo dentro de seu contexto da vida real, especialmente quando os limites entre o fenômeno e o contexto não estão claramente definidos. De acordo com o tipo de questões de pesquisa apresentadas nesse trabalho, vamos utilizar uma estratégia de **levantamento** associada a um **estudo de caso**. Em virtude de adotar uma estratégia exploratória, um levantamento será usado para listar diversos modelos criptográficos, assim como suas vantagens e usos. O estudo de caso será direcionado para fazer as comparações e responder os porquês associados a falta de uso (ou pouco uso) de curvas elípticas.

A abordagem de mais de uma estratégia, como o caso da mista de levantamento com estudo de caso é permitida e até citada por Yin [17]. O referido autor também orienta o estudo de caso como estratégia de pesquisa que compreende um método que abrange tudo – com a lógica de planejamento incorporando abordagens específicas à coleta e à análise de dados.

Sobre a unidade de análise, Yin [17] ensina que o **caso** pode ser um evento, entidade, ou programa, que são menos definidos que um indivíduo. E nesse trabalho propõe-se o estudo de caso Criptografia de Curvas Elípticas em comparação com casos de outras metodologias criptográficas. A Figura 1.1 apresenta o fluxo de trabalho por meio das metodologias de pesquisa levantamento de dados e estudo de caso. Os seguintes passos serão seguidos em sequência na realização desse trabalho:

1. Projeto: etapa que envolve o planejamento do escopo do trabalho, a definição do problema de pesquisa e das questões de pesquisa assim como a escolha das meto-

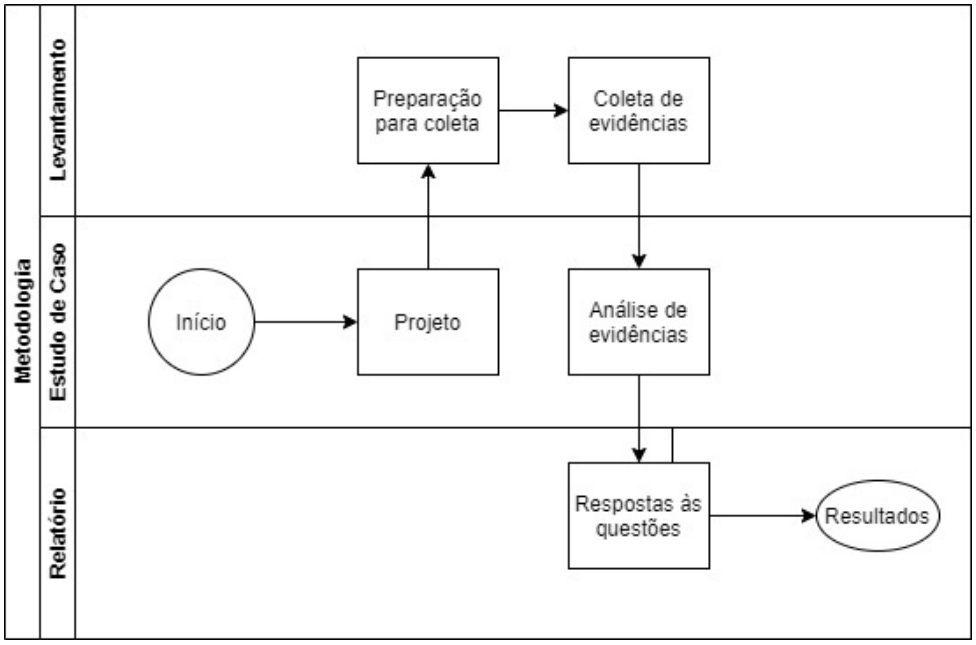

Figura 1.1: Diagrama com o fluxo de trabalho associado a metodologia escolhida

dologias a serem aplicadas a partir dessas questões. Etapa de definição do caso e revisão da teoria, levantamento das proposições e hipóteses. Para estudos de caso, Yin [17] também orienta que o desenvolvimento da teoria como parte da fase de projeto é essencial, caso o propósito decorrente do estudo de caso seja determinar ou testar uma teoria.

2. Preparação para coleta: etapa de levantamento, testes e ajustes nos instrumentos de coleta de dados para aplicação da teoria e produção de evidências. Pelo proposto por Yin [17], precisa-se ter as habilidades necessárias, fazer o treinamento para um estudo de caso específico, desenvolver um protocolo para a investigação e conduzir um estudo piloto.

3. Coleta de evidências: etapa de efetiva coleta de dados a partir das fontes disponíveis. Yin [17] recomenda três princípios para a realização dessa etapa: a utilização de várias fontes de evidências, a criação de um banco de dados para o estudo de caso, e a manutenção de um encadeamento de evidências.

4. Análise de evidências: etapa que consiste em examinar, categorizar, classificar, e até reavaliar as evidências a partir das proposições e hipóteses iniciais do estudo. Baseando-se em proposições teóricas para guiar a análise, dentre as técnicas disponíveis, a escolhida é a adequação ao padrão.

5. Composição do relatório: etapa de reunião de todas as informações usadas e geradas pelo estudo. O formato do relatório do estudo de caso a ser adotado nesse estudo

será em estruturas comparativas e estruturas de construção da teoria. Segundo Yin [17], a estrutura comparativa repete o mesmo estudo de caso duas ou mais vezes, comparando as descrições ou explanações alternativas do mesmo caso. Ele argumenta que o propósito da repetição é mostrar até que ponto os fatos adaptam-se a cada modelo, e as repetições ilustram a técnica de adequação ao padrão em atividade. A intenção de usar também a estrutura de construção de teoria é fazer a sequência de capítulos e seções terem uma lógica de construção para produzir as conclusões e respostas às questões de pesquisa.

Vamos considerar os seguintes passos como fases metodológicas desse trabalho:

I: Investigar na literatura a utilização da criptografia de curvas elípticas.

II: Implementar em linguagem de programação alguns casos de modelos de criptografia.

III: Calcular a complexidade computacional da RSA e da ECC.

1.5 Estrutura do Trabalho

Este trabalho está organizado da seguinte maneira: O Capítulo 2 discorre sobre o referencial teórico e os principais textos usados como referência para esse trabalho. Fundamentos de criptografia e suas classificações e detalhes fazem parte. São explicados desde conceitos matemáticos elementares para essa área, passando pela teoria básica de segurança, até os modelos de criptografia mais gerais. No trecho mais específico, são detalhados os elementos da criptografia de curvas elípticas. São citados desde textos tradicionais sobre segurança computacional, passando por artigos científicos específicos do tema principal de Criptografia de Curvas Elípticas e notícias sobre ataques virtuais criminosos.

No Capítulo 3 apresenta-se o desenvolvimento do trabalho em si a partir da metodologia e teoria tratados nos capítulos anteriores. Toda a implementação em ambiente de programação está exposta e analisada. A discussão sobre a comparabilidade de modelos de criptografia, assim como as aplicações de ECC também fazem parte desse capítulo. Os resultados obtidos com as comparações entre os modelos estão listados por meio da implementação de um caso de cifragem de uma mensagem codificada em um número. Ainda são mostradas possíveis respostas para as questões de pesquisa e ao objetivo do trabalho, além de apresentar suas limitações.

O Capítulo 4 apresenta as conclusões do trabalho. São feitas algumas reflexões e indicações para trabalhos futuros.

Em virtude de se usar muita matemática e termos matemáticos, um capítulo complementar foi feito para ilustrar esse conteúdo, como Apêndice A. Nele são apresentados

conceitos e propriedades da Aritmética e da Álgebra importantes para o desenvolvimento desse trabalho.

Outro capítulo foi feito como Apêndice B para explicitar a aplicação das funções criadas nos códigos-fontes e produzir os resultados dos exemplos citados no capítulo 3.

Capítulo 2

Teoria da Criptografia

A criptografia existe há muito tempo como uma ferramenta capaz de viabilizar uma comunicação secreta menos vulnerável, que podemos chamar de *mais segura*. Segundo [14], historicamente os maiores usuários de criptografia foram organizações militares e governos. Hoje, a criptografia está em todo lugar na vida de uma pessoa comum. Desde um acesso ao *e-mail* até uma compra numa loja de conveniência.

A palavra **criptografia** vem do latim, em que *cripto* significa oculto/escondido e *grafia* significa escrita. Dessa forma, criptografia pode ser entendida como o conjunto de métodos e técnicas para cifrar e codificar informações legíveis por meio de um algoritmo, sendo viável, mediante processo inverso, recuperar as informações originais, [16]. Nesse trabalho, vamos usar a definição de criptografia de Katz [14].

Definição 1 [Criptografia] Estudo de técnicas matemáticas para segurança de dados, sistemas e qualquer plataforma digital contra ataques criminosos.

Segundo [12], podemos classificar os possíveis ataques criminosos em dois grupos: ataques passivos e ataques ativos. Os ataques passivos são usados para monitorar ou capturar dados e informações confidenciais de terceiros (vazamento). Já os ataques ativos são aqueles que modificam de alguma forma os dados ou criam novos dados falsos (fraude).

De acordo com [6], grande parte dos estudos de segurança de dados se concentram em três categorias:

- Confidencialidade ou sigilo: garantir que a informação seja acessada apenas por aqueles com acesso autorizado.
- Integridade: resguardar que a informação seja completa e não seja modificada.
- Disponibilidade: garantir que a informação possa ser acessada sempre que for solicitada.

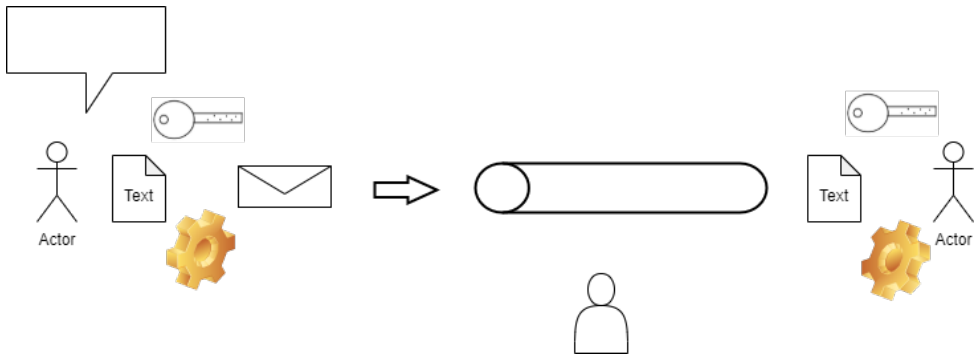

Figura 2.1: Elementos de uma situação comunicativa

Podemos resumir uma situação comunicativa padrão, assim como [1] faz, como um *teatro da segurança*. Nesse teatro temos:

1. Atores: Agentes principais A e B, que chamamos como personagens **Alice** e **Bob**, e possíveis outros agentes

2. Cenários: Neutralidade tecnológica de hardware e software, e evolução tecnológica desejável e benéfica

3. Papéis: Emissor, receptor, interlocutor, autoridade, etc

4. Cenas: Entre os planos psíquico (o que se quer fazer) e físico (o que realmente é feito)

5. Enredos: Riscos, mecanismos de proteção, etc.

Ao longo de uma situação comunicativa, os interlocutores Alice e Bob podem desempenhar ambos os papéis de emissor e receptor. Eles usam um canal transmissivo, por onde eles fazem a sua comunicação (trocam sinais que codificam dados). Se o canal transmissivo for um meio virtual ou digital, podemos dizer que é um canal em banda, [1]. Uma pequena ilustração resumida com alguns elementos de uma situação comunicativa pode ser vista na figura 2.1.

A função da Segurança da Informação nesse cenário seria **proteger dados** segundo os interesses dos agentes principais e do dono do canal, evitando ou permitindo certas ações. As ações que seriam evitadas são as tentativas de ataques passivos, como vazamentos, e de ataques ativos, como fraudes.

A criptografia é uma das ferramentas usadas para obter segurança da informação. A proteção requerida, no entanto, não é para os dados em si, mas sim para valores que os dados podem significar.

Segundo [12], em uma situação comunicativa, chamamos de **texto claro** a mensagem ou os dados originais que o emissor deseja que o receptor receba. Já o **algoritmo de**

Figura 2.2: Processo de cifragem

Figura 2.3: Processo de decifragem

criptografia realiza diversas transformações no texto claro, esse processo é chamado **cifragem** e ele está resumido na figura 2.2. O resultado disso é o **texto crifrado**, uma mensagem com caracteres embaralhados. Uma **chave secreta** é também usada como entrada no algoritmo de criptografia. A chave é um valor independente do texto claro e do algoritmo. Para cada valor de chave, o algoritmo produz uma saída de texto cifrado diferente. Por fim, o **algoritmo de decifragem** ou decriptografia é usado para transformar o texto cifrado em texto claro e esse processo de decifragem está ilustrado na figura 2.3.

As técnicas usadas para decifrar uma mensagem sem qualquer conhecimento dos detalhes de criptografia estão na área da criptoanálise, [12]. Informações como natureza do algoritmo, características gerais do texto claro ou pares de amostras de texto claro com texto cifrado podem ser exploradas por um criptoanalista, embora sempre exista a técnica de ataque por *força bruta*, na qual o atacante experimenta cada chave possível em um trecho do texto cifrado até obter uma tradução inteligível para o texto claro [12].

Rezende [13] faz um resumo das classificações de ataques listadas por Ford [18]. As *ameaças* são vazamento, fraude, bloqueio e uso indevido. Os ataques primários podem ser por penetração ou por implantação [18]. Dentre os ataques por penetração, as modalidades mais conhecidas são personificação, desvio de controle e violação de autoridade. Já os mais conhecidos ataques por implantação são gancho ou *backdoor*, infecção (vírus) e embuste. Os ataques subjacentes podem estar por trás de qualquer ataque primário ou ameaça [18] são listados por: escuta passiva, análise de tráfego, descuido, grampo, varredura, escuta ativa, refutação, sobrecarga intencional, furto de sessão, *replay* e espelhamento.

Assim como diz [14], a criptografia clássica esteve concentrada em desenho e uso de códigos, também chamados de cifras, que permitem às duas partes uma comunicação secreta na presença de um interceptador que pode monitorar toda a comunicação entre

eles. Existem dois grandes grupos de modelos de criptografia:

- Criptografia Simétrica ou de Chave Privada
- Criptografia Assimétrica ou de Chave Pública

2.1 Criptografia Simétrica

Segundo [14] e [13], um esquema de criptografia simétrica a partir de um espaço de mensagens $\mathcal{M}$, um espaço de chaves $\mathcal{K}$, um espaço de criptogramas $\mathcal{C}$ e três algoritmos é dado por:

1. Algoritmo de geração de chave Gen é um algoritmo probabilístico que fornece uma chave k de acordo com alguma distribuição dentro de $\mathcal{K}$.
2. Algoritmo de cifragem Enc pega como entradas uma chave $k \in \mathcal{K}$ e uma mensagem em texto claro $m \in \mathcal{M}$, e fornece como saída o texto cifrado $c \in \mathcal{C}$ associado. A notação é dada por: $c = Enc_k(m)$. Enc é uma função parametrizável por k e inversível em m.
3. Algoritmo de decifragem Dec pega como entradas a chave k e o texto cifrado c, e fornece a mensagem em texto claro m. A notação é dada por: $m = Dec_k(c)$

Um requisito que deve ser satisfeito para que todo o processo seja válido é que: para qualquer chave $k \in \mathcal{K}$ gerada por Gen e qualquer mensagem $m \in \mathcal{M}$, seja verdade que:

$$Dec_k(Enc_k(m)) = m \tag{2.1}$$

Podemos dividir as técnicas de criptografia simétrica em técnicas clássicas e técnicas modernas [12]. Enquanto as técnicas clássicas se baseiam geralmente em cifras de fluxo, quando codifica um bit ou um byte de cada vez; as cifras modernas são baseadas em cifras de bloco, em que um bloco de texto claro é tratado como um todo e usado para produzir um bloco de texto cifrado com o mesmo tamanho.

No grupo das técnicas clássicas, temos:

- Técnicas de substituição: quando letras de texto claro são substituídas por outras letras ou por números ou símbolos. São exemplos a Cifra de César, cifras monoalfabéticas, cifras polialfabéticas e One-Time Pad.
- Técnicas de transposição: quando o mapeamento das letras é obtido pela realização de algum tipo de permutação nas letras do texto claro.

- Máquinas de rotor: associação de múltiplos cilindros em que cada cilindro define uma cifra de substituição polialfabética. O principal exemplo desse tipo de máquina foi a *Enigma* usada pela Alemanha na Segunda Guerra Mundial [12].
- Esteganografia: técnica de esconder a mensagem de texto claro. São exemplos: arranjo de palavras e letras dentro de um texto aparentemente inofensivo soletra a mensagem real; marcação de caractere visível de acordo com um ângulo com uma fonte de luz; tinta visível quando uma química é aplicada ao papel, etc.

A cifra de César é conhecida por ser uma das primeiras e mais simples [12], a chave é facilmente descoberta pelo fato do espaço de chaves $\mathcal{K}$ ser pequeno, com apenas 26 chaves possíveis, por isso é apenas ilustrativa. Apenas aumentar o tamanho do espaço de chaves não garante a segurança requerida em ataques de força bruta, pois outras técnicas de ataque podem ser usadas na criptoanálise, tais como análise estatística de frequência relativa de letras do alfabeto [12]. Já no conjunto de técnicas modernas, podemos destacar conforme [12]:

- *Data Encryption Standard - DES*: cifra de Feistel com blocos de 64 bits do texto claro e chave com 56 bits é usada em 16 rodadas de difusão e confusão.
- *Advanced Encryption Standard - AES*: versão mais sofisticada do DES com blocos de 128 bits do texto claro e chave com 128 a 256 bits, em que as operações são realizadas no grupo finito $GF(2^8)$.

Segundo [12], Feistel [19] propôs uma cifra de bloco usando uma cifra produto de duas ou mais cifras simples em sequência, alternando substituições e permutações. E essa técnica foi baseada para aplicar os termos *difusão* e *confusão* definidos por Shannon [20] como técnicas para impedir criptoanálise baseada em análise estatística. Dessa forma, na difusão, a estrutura estatística do texto claro é dissipada em estatísticas de longa duração do texto cifrado. E, na confusão, a relação entre o valor da chave e as estatísticas de frequência do texto ficam mais complexas e difíceis de usar para dedução de chave. Essas técnicas podem ser avaliadas medindo o *efeito avalanche*, que mostra o que ocorre com o texto cifrado quando se altera um bit do texto claro ou da chave.

Ainda segundo [12], o polinômio usado para a aritmética de grupo finito $GF(2^8)$ em AES é dado por $m(x) = x^8 + x^4 + x^3 + x + 1$. Além disso, o AES não usa cifra de Feistel por não dividir o bloco em duas metades.

Para que a criptografia simétrica possa funcionar, as duas partes precisam compartilhar a mesma chave, e ela precisa estar protegida contra acesso por outras partes [12]. Desse modo, mesmo que a geração de chaves seja feita por um método aleatório, a distribuição de chaves é um problema pois recomenda-se que se use um canal confiável para esse

compartilhamento. Outra forma de compartilhamento é o uso de uma terceira parte confiável, que pode inclusive ser um *centro de distribuição de chaves* [12].

2.2 Criptografia Assimétrica

Fazendo um paralelo com os detalhes da criptografia simétrica, [6] e [13] enumeram os elementos que devem ter em um esquema de criptografia assimétrica a partir de um espaço de mensagens $\mathcal{M}$, um espaço de chaves $\mathcal{K}$ e um espaço de criptogramas $\mathcal{C}$:

1. Algoritmo de geração de chave Gen é um algoritmo probabilístico que fornece um par de chaves $(pk, sk) \in \mathcal{K}$ (chave pública e chave secreta) de acordo com alguma distribuição.

2. Algoritmo de cifragem Enc pega como entradas a chave pública pk e uma mensagem em texto claro $m \in \mathcal{M}$, e fornece como saída o texto cifrado $c \in \mathcal{C}$ associado. A notação é dada por: $c = Enc_{pk}(m)$

3. Algoritmo de decifragem Dec pega como entradas a chave privada sk e o texto cifrado c, e fornece a mensagem em texto claro m. A notação é dada por: $m = Dec_{sk}(c)$

Um requisito que deve ser satisfeito para que todo o processo seja válido é que: para qualquer par de chaves (pk, sk) gerada por Gen e qualquer mensagem $m \in \mathcal{M}$, seja verdade que:

$$Dec_{sk}(Enc_{pk}(m)) = m \tag{2.2}$$

A criptografia assimétrica surgiu a partir de uma tentativa de resolver o problema de distribuição de chaves, comum em criptografia simétrica [12]. A proposta de Diffie e Hellman em 1976 [21] ocorre da seguinte maneira, segundo [15]:

Suponha que Alice e Bob queiram estabelecer uma chave secreta compartilhada. Alice seleciona aleatoriamente um número inteiro secreto a tal que $a \in \mathbb{Z}_p$ e envia para Bob $P_A = g^a \pmod p$, sendo p um número primo e g um gerador do grupo cíclico $\mathbb{Z}_p^*$. Analogamente, Bob escolhe um inteiro secreto $b \in \mathbb{Z}_p$ e envia à Alice $P_B = g^b \pmod p$. Agora ambos usam sua chave secreta para obter uma chave secreta compartilhada $k = (P_A)^b = (P_B)^a = g^{ab} \pmod p$.

Logo, a segurança desse protocolo Diffie Hellman está baseada na dificuldade computacional de calcular a ou b dados g e p [15]. Esse problema é conhecido como *Problema do Logaritmo Discreto - PLD*, conforme definição 6 do Apêndice A. Quando esses parâmetros e variáveis são relativamente grandes, esse problema se torna inviável [15].

2.2.1 Criptografia RSA

Em 1978, Rivest, Shamir e Adleman propuseram um algoritmo criptográfico de chave pública baseado na dificuldade para fatorar inteiros grandes [12] [14]. Desde então, a técnica foi largamente utilizada como modelo de criptografia assimétrica e chamada de *algoritmo RSA*. Vamos ilustrar o funcionamento do RSA pela sua importância para esse tema.

No RSA, o texto claro é criptografado em blocos, com cada bloco tendo um valor binário menor que algum número n [12]. Assim, para algum bloco de texto claro M e bloco de texto cifrado C:

$$C = M^e \pmod{n}$$

$$M = C^d \pmod{n} = (M^e)^d \pmod{n} = M^{ed} \pmod{n}$$

Tanto o emissor quanto o receptor precisam conhecer o valor de n. O emissor conhece o valor de e, e somente o receptor conhece o valor de d. Dessa forma a chave pública é dada por $pk = (e, n)$ e a chave privada é dada por $sk = (d, n)$. Assim, para que o algoritmo seja satisfatório, precisa cumprir os requisitos:

1. Ser possível encontrar valores de e, d, n tais que $M^{ed} \pmod{n} = M$ para todo $M < n$.

2. Ser relativamente fácil calcular $M^e \pmod{n}$ e $C^d \pmod{n}$ para todos os valores de $M < n$.

3. Ser inviável determinar d dados e e n.

Sobre o primeiro requisito, ele é garantido se e e d forem inversos multiplicativos módulo $\phi(n)$. O teorema de Euler dado pela equação A.2 é usado na demonstração desse fato. E os componentes para o RSA são dados por:

- p, q, dois números primos escolhidos e sigilosos
- $n = pq$, calculado e divulgado
- e, tal que $\text{mdc}(\phi(n), e) = 1$; $1 < e < \phi(n)$, escolhido e divulgado
- $d \equiv e^{-1} \pmod{\phi(n)}$ calculado e sigiloso

O seguinte exemplo de RSA foi retirado e adaptado de [13]:

Suponha que Alice precise receber uma mensagem secreta de Bob. Para isso ela escolhe dois número primos, $p = 3$ e $q = 11$. Calcula-se $n = pq = 3 * 11 = 33$ e $\phi(33) = \phi(3)\phi(11) = 2 * 10 = 20$. Escolhe-se um inteiro e relativamente primo com $\phi(33) = 20$ e calcula-se o seu inverso em $\mathbb{Z}_{20}$. Por exemplo, $e = 17$, pelo algoritmo de Euclides estendido resolve-se $17d + 20y = 1$ e encontra-se $d = 13$. Assim, a chave pública de Alice é dada por $(e, n) = (17, 33)$ a qual Bob terá conhecimento; e a chave privada de Alice é dada por $(d, n) = (13, 33)$, que ela vai guardar em segredo. Seja um bloco de cinco bits da mensagem de Bob em texto claro $m = (10100)_2 = 20$. Ele então vai cifrar esse bloco com a chave pública de Alice usando $c = Enc_{pk}(m) = m^e \pmod{n} = 20^{17} \pmod{33} = 26 = (11010)_2$ e transmitir esse criptograma c para Alice. Alice então vai receber o criptograma e decifrar usando sua chave privada com $m = Dec_{sk}(c) = c^d \pmod{n} = 26^{13} \pmod{33} = 20 = (10100)_2$.

2.2.2 Criptografia de Curvas Elípticas

Vamos falar um pouco sobre as Curvas Elípticas para, em seguida, contextualizar a criptografia envolvida. Usando a definição de [15], temos:

Definição 2 [Curva Elíptica] Uma curva elíptica sobre um corpo $\mathbb{F}$, de característica maior que 3, é o lugar geométrico dos pontos $(x, y) \in \mathbb{F} \times \mathbb{F} = \mathbb{F}^2$ que satisfazem a equação

$$y^2 + Axy + By = x^3 + Cx^2 + Dx + E \tag{2.3}$$

mais um ponto, chamado de ponto no infinito, que será denotado por ∞. Podemos simplificar a equação 2.3 deixando na forma

$$y^2 = x^3 + ax + b \tag{2.4}$$

com $a, b \in \mathbb{F}$. Esta curva deve ser uma curva não-singular, ou seja, não possuir raízes múltiplas. Para tanto, precisamos ter

$$\Delta = 4a^3 + 27b^2 \neq 0 \tag{2.5}$$

Se associarmos uma determinada soma, podemos dizer que os pontos da curva elíptica satisfazem uma estrutura algébrica de grupo. Vamos definir essa soma. Para isso, considere que Ω é uma curva elíptica sobre um corpo $\mathbb{F}$ e que $P \in \Omega$:

$$P + \infty = P = \infty + P \tag{2.6}$$

Chamando $P = (x, y)$ então definimos $-P = (x, -y)$, e assim:

$$P + (-P) = \infty = (-P) + P \tag{2.7}$$

Alguns autores fazem um apelo geométrico para definir a soma de dois pontos de uma curva elíptica definida sobre o corpo dos reais $\mathbb{R}$. Mas como esse corpo tem característica zero, então ele não obedece a definição 2, e por isso vamos evitar tal apelo geométrico. Considere que o corpo é dado por $\mathbb{F} = \{0, 1, 2, ..., p-1\}$, sendo p um primo qualquer. Desse modo, para garantir o fechamento das operações aritméticas, elas serão calculadas em função do resto da divisão por p, ou seja, pela aritmética modular.

Assim, para somar $P = (x_P, y_P)$ com $Q = (x_Q, y_Q)$, considerando que $Q \neq -P$ e que $P + Q = R = (x_R, y_R)$, faz-se:

$$x_R = \lambda^2 - x_P - x_Q \pmod{p} \tag{2.8}$$

$$y_R = \lambda(x_P - x_R) - y_P \pmod{p} \tag{2.9}$$

onde

$$\lambda = \begin{cases} \dfrac{y_Q - y_P}{x_Q - x_P} \pmod{p}, & \text{se } x_P \neq x_Q \\ \dfrac{3x_P^2 + a}{2y_P} \pmod{p}, & \text{se } x_P = x_Q \text{ e } y_P \neq 0 \end{cases} \tag{2.10}$$

Para multiplicar um ponto P por um número natural n, basta somá-lo n vezes. Além disso, vamos denotar por $E_p(a, b)$ o conjunto de todos os pares de inteiros (x, y) que satisfazem a equação 2.4, juntamente com o ponto infinito ∞ [12].

Definição 3 [Ordem] A quantidade de pontos que satisfaz uma curva elíptica Ω (e o ponto no infinito) é chamado ordem e denotada por $\#\Omega$.

Katz [14] enuncia o Teorema de Hasse que traz um intervalo de variação para a ordem da curva elíptica.

Teorema 1 (Limite de Hasse)**.** *Seja p um número primo e E uma curva elíptica sobre $\mathbb{Z}_p$, ou seja $\Omega = E(\mathbb{Z}_p)$. Então $p + 1 - 2\sqrt{p} \leq \#\Omega \leq p + 1 + 2\sqrt{p}$.*

Com isso, conseguimos terminar a definição do grupo finito elíptico $\Omega = (E_p(a, b), +)$. Segundo [14], a condição de não singularidade 2.5 serve para que a equação $x^3 + ax + b \equiv 0 (\text{mod } p)$ não tenha raízes repetidas.

A operação de adição de pontos em curvas elípticas é equivalente à multiplicação modular usada no RSA, e a adição múltipla (multiplicação de um ponto por um inteiro

positivo) é o equivalente da exponenciação modular, [12]. Para formar um criptossistema usando curvas elípticas, precisamos encontrar um *problema difícil* como fatorar o produto de dois primos ou calcular o logaritmo discreto. Considere a equação $Q = kP$, onde $Q, P \in E_p(a, b)$ e $k \in \mathbb{N}$ tal que $k < p$. É relativamente fácil calcular Q a partir de k e P, mas é computacionalmente difícil determinar k a partir de Q e P [12]. Esse é o chamado problema do logaritmo discreto para curvas elípticas.

Um sistema de criptografia de curva elíptica é um tipo específico de criptografia assimétrica que requer um grupo elíptico $E_q(a, b)$ e um ponto G como parâmetros [12]. Cada usuário X seleciona uma chave privada n_X e gera uma chave pública $P_X = n_X \times G$. Para cifrar e enviar uma mensagem P_m a B (Bob), A (Alice) escolhe um inteiro positivo aleatório k e produz o texto cifrado C_m consistindo no par de pontos $C_m = \{kG, P_m + kP_B\}$. Observe que Alice usou a chave pública de Bob, P_B. Para decifrar o texto cifrado, Bob multiplica o primeiro ponto do par pela sua chave secreta n_B e subtrai o resultado do segundo ponto:

$$P_m + kP_B - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m \tag{2.11}$$

Note que Alice mascarou a mensagem P_m somando kP_B a ela. Ninguém além de Alice conhece o valor de k, de modo que, embora P_B seja uma chave pública, ninguém pode remover a máscara kP_B. Porém Alice também inclui uma *dica*, que é suficiente para remover a máscara se alguém souber a chave privada n_B. Para um atacante recuperar a mensagem, ele teria que calcular k a partir de G e kG, o que é considerado difícil e o que garante a segurança do ECC.

2.3 Trabalhos Correlatos

O tema de Criptografia de Curvas Elípticas foi proposto inicialmente por Miller e Koblitz, respectivamente em 1985 e 1987, segundo [22]. O trabalho original de Miller é o artigo [23]. Ao longo do tempo, muitos estudos foram feitos sobre ECC. A busca por publicações sobre o tema para esse trabalho envolveu artigos publicados em 2021, com algumas exceções.

Dentre os artigos analisados, o primeiro foi dos pesquisadores Fábio Borges e Pedro Lara [15] publicado em 2007. Ele trata exemplos de aplicação da criptografia de curvas elípticas. Ainda sobre a construção do modelo de Curvas Elípticas, Pedro Lara [16] aprofundou um pouco mais o tema em 2008 e propôs uma API (*Application Programming Interface*) de uma ECC.

Já uma aplicação mais avançada de Curvas Elípticas, foi encontrada no artigo dos pesquisadores Majumder et al. [22] de 2021. Os autores apresentam uma proposta de

esquema de estabelecimento de chave de sessão entre dispositivos que usam IoT e seus respectivos servidores por meio de ECC. Essa técnica é considerada segura pelos autores uma vez que conseguiu se defender de ataques conhecidos em ambiente de teste.

Chen et al. [7] (de 2021) apresentaram um método padrão de criptografia híbrida para auditoria de integridade de dados sem a necessidade de uma autoridade confiável. A criptografia híbrida desse trabalho usa o método AES (*Advanced Encryption Standard*) conhecido método de criptografia simétrica, com o método de ECC como criptografia assimétrica.

Ainda em 2021, Singh et al [24] apresentaram uma aplicação de autenticação de usuários de dispositivos IoT da área de saúde com o objetivo de bloquear ataques a dados de pacientes. A proposta sugeriu um esquema de ECC para preservação da privacidade dos usuários (autenticação) no lugar da técnica de criptografia de Li et al [25], de 2018, a qual é muito atacada.

Aggarwal e Kumar [8] em 2021 descreveram a aplicação de um algoritmo de assinatura digital por meio de curvas elípticas. Os autores chamaram a técnica de *Elliptic Curve Digital Signature Algorithm* (ECDSA).

Abbas et al [26] apresentaram uma forma de criptografia de imagens usando ECC tanto para geração de números aleatórios quanto para a cifragem, tudo isso rodando em paralelo. Eles conseguiram ganhos de segurança, *performance* e eficiência em seu trabalho publicado em 2021.

Este trabalho pretende explorar as potencialidades da criptografia de curvas elípticas e fazer a sua implementação comparando com outra técnica de criptografia assimétrica.

2.4 Síntese do Capítulo

Este capítulo descreveu os conceitos de criptografia, aspectos de segurança de dados, componentes de uma situação comunicativa, função da segurança da informação, texto claro, criptograma, chave secreta, chave privada, chave pública, algoritmo de criptografia, algoritmo de decifragem, criptoanálise, ataque de força bruta, criptografia simétrica, criptografia assimétrica e criptografia de curvas elípticas. Foram fornecidos exemplos dessas teorias para ilustrar o seu funcionamento.

No Capítulo 3 será apresentado o desenvolvimento do estudo de caso prático por meio de implementação e comparações das técnicas. Os resultados, as comparações e as limitações também serão apresentados.

Capítulo 3

Desenvolvimento

Neste Capítulo são apresentadas as implementações dos algoritmos em linguagem de programação. A escolha da linguagem, suas características, e os detalhes dos algoritmos implementados com os dados de entrada, saída além da análise de complexidade são objeto desse capítulo.

3.1 Linguagem R

A Linguagem R surgiu no início da década de 90 como uma versão livre da linguagem de programação estatística S ou S-plus. Segundo o seu site [27], R está disponível como um Software Livre sob os termos da Licença Pública Geral GNU da *Free Software Foundation* na forma de código-fonte. Ele compila e roda em uma ampla variedade de plataformas UNIX, Windows e Mac. A imagem de sua logo está na figura 3.1.

O ambiente R possui facilidade com manipulação de dados e cálculos matemáticos. R permite aos seus usuários trabalhar com funcionalidades adicionais por meio de definição de funções [27]. A linguagem também pode ser facilmente estendida via pacotes disponibilizados em bibliotecas na internet por meio da sua rede de usuários colaboradores.

O R foi escolhido como linguagem de programação para esse trabalho devido a maior experiência que tenho com ela, em comparação com outras linguagens. Essa escolha visou direcionar mais tempo ao conteúdo do trabalho em si, do que a ferramenta de aplicação.

3.2 Cifras Simétricas

Uma primeira parte do estudo é implementar duas cifras simétricas. O objetivo é mostrar o funcionamento dessas técnicas no ambiente R. Considera-se então uma mensagem que se quer cifrar e uma chave que será usada como parâmetro em ambos os algoritmos.

Figura 3.1: Logo da linguagem R.

3.2.1 Cifra de César

Resumidamente, segundo [12], a Cifra de César consiste em substituir cada letra do alfabeto pela letra que fica três posições adiante no alfabeto.

Em R, existe uma biblioteca chamada *Caesar*, que faz o básico mas não atende bem por não usar a tabela ASCII estendida, ou seja, não codifica determinados caracteres tais como vogais acentuadas, por exemplo. Desse modo optou-se por implementar o algoritmo.

```
1 library(DescTools)
2
3 cesar_completo1<-function(texto1, i=3){
4   AscToChar(CharToAsc(texto1)+i)
5 }
6
7 decifra_cesar_completo1<-function(cifra1, i=3){
8   AscToChar(CharToAsc(cifra1)-i)
9 }
```

Listing 3.1: Código fonte em R da Cifra de César

Aplicando a função *cesar_completo*1 à mensagem

Amar ao próximo como a ti mesmo.

o resultado é a seguinte cadeia de caracteres:

Dpdu#dr#suö{lpr#frpr#d#wl#phvpr1

E, ao aplicar a funçao inversa $decifra\_cesar\_completo1$ à mensagem cifrada, retorna-se a mensagem original. O deslocamento observado na cifra de César pode ser de qualquer quantidade, não precisa ser obrigatoriamente de três unidades.

3.2.2 Cifra de Vigenère

A Cifra de Vigenère é um tipo de cifra polialfabética que é uma generalização da cifra de César. Stallings [12] diz que cada letra da mensagem passa por uma regra de substituição monoalfabética por um deslocamento da cifra de César. Esse deslocamento é determinado por uma letra-chave, que é a letra do texto cifrado que substitui a letra do texto claro **a**. Assim, a letra-chave da cifra de César padrão com deslocamento de três unidades é representada pela letra **d**. Dessa forma, para cifrar uma mensagem, precisa-se de um conjunto de letras-chave, chamada de **chave**, que, em geral, é menor que o tamanho da mensagem. E deve-se então repetir a chave até o fim da mensagem. A cifra *One-Time Pad* é uma variação da Vigenère quando o tamanho da chave é o mesmo tamanho da mensagem, [6].

A cifra de César tem finalidade apenas ilustrativa devido a sua alta vulnerabilidade por conta da facilidade de descoberta da mensagem original por força bruta. A cifra de Vigenère também tem uma certa vulnerabilidade e está sujeita a ser descoberta com uma criptoanálise não muito complexa. Algumas alternativas foram elencadas para elevar a segurança da cifra de Vigenère, uma delas foi considerar a chave do tamanho do texto claro. Outra técnica é tratar os dados como binários e fazer uma operação *XOR* entre os bits da chave com os bits do texto. E essa foi a técnica implementada:

```
1  library(R.utils)
2
3  zera8_ <- function(bitasc){
4    while(nchar(bitasc)<8){
5      bitasc<-paste0("0",bitasc)
6    }
7    return(bitasc)
8  }
9
10 concatbits<-function(bits){
11   novo<-""
12   caracteres<-length(bits)
13   for(j in 1:caracteres){
```

```
    novo<-paste0(novo,bits[j])
  }
  return(novo)
}

letrachave2<-function(codascl,codasck){
  binl<- intToBin(codascl)
  bink<- intToBin(codasck)
  binl_<-zera8_(binl)
  bink_<-zera8_(bink)
  binql <- strsplit(binl_, split = "")[[1]]
  binqk <- strsplit(bink_, split = "")[[1]]
  bl <- as.integer(binql)
  bk <- as.integer(binqk)
  cf<- xor(bl,bk)
  cfb<-as.integer(cf)
  cfb_<-concatbits(cfb)
  c1<-as.integer(cfb_)
  c_<-BinToDec(c1)
  return(c_)
}

cifragemvig2<-function(msgn,chaven){
  cifra<-NA
  tmsg<-length(msgn)
  tchave<-length(chaven)
  for(l in 1:tmsg){
    letramsg <- msgn[l]
    pchave <- if(l %% tchave == 0){tchave}else{l %% tchave}
    letrakey <- chaven[pchave]
    letracifra <- letrachave2(letramsg,letrakey)
    cifra<-c(cifra,letracifra)
  }
  return(cifra[2:(tmsg+1)])
}
```

Listing 3.2: Código fonte em R da Cifra de Vigenère

A função $cifragemvig2$ depende de dois argumentos, que são a mensagem em texto claro e a chave, ambas já em formato ASCII. Para cada letra da mensagem, usa a letra correspondente na chave para fazer o XOR por meio da função $letrachave2$. Essa última função usa as funções complementares $zera8\_$ e $concatbits$ para deixar o binário de cada caracter em oito bits e para concatenar os resultados do XOR bit a bit em um novo

binário, respectivamente. Considere a seguinte mensagem:

A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original.

A sequência ASCII da mensagem acima é dada por

$$\begin{aligned} &65, 32, 109, 101, 110, 116, 101, 32, 113, 117, 101, 32, 115, \\ &101, 32, 97, 98, 114, 101, 32, 97, 32, 117, 109, 97, 32, 110, 111, \\ &118, 97, 32, 105, 100, 101, 105, 97, 32, 106, 97, 109, 97, 105, \\ &115, 32, 118, 111, 108, 116, 97, 114, 225, 32, 97, 111, 32, 115, \\ &101, 117, 32, 116, 97, 109, 97, 110, 104, 111, 32, 111, \\ &114, 105, 103, 105, 110, 97, 108, 46 \end{aligned} \tag{3.1}$$

E considere também a seguinte chave: **tccFernando**, que tem como sequência ASCII: $116, 99, 99, 70, 101, 114, 110, 97, 110, 100, 111$. Ao aplicar a função $cifragemvig2$ às sequências ASCII da mensagem e da chave, obtemos a seguinte sequência ASCII da cifra gerada:

$$\begin{aligned} &53, 67, 14, 35, 11, 6, 11, 65, 31, 17, \\ &10, 84, 16, 6, 102, 4, 16, 28, 4, 78, 5, 79, 1, 14, 2, 102, 11, 29, 24, \\ &0, 78, 13, 11, 17, 102, 102, 15, 19, 3, 0, 7, 23, 79, 2, 12, 15, 50, \\ &4, 0, 143, 65, 15, 11, 79, 7, 6, 22, 102, 17, 19, 3, 0, 0, 12, 0, 84, \\ &12, 17, 472, 27, 0, 0, 2, 74 \end{aligned} \tag{3.2}$$

O retorno da sequência ASCII da cifra gerada para os caracteres correspondentes é prejudicada por haver caracteres não imprimíveis, por isso não foi feita.

3.3 Cifras Assimétricas

3.3.1 RSA

A clássica cifra RSA está baseada no antigo problema de fatoração numérica [14]. Esse problema é resumido da seguinte maneira: dado um número inteiro composto N, pede-se encontrar inteiros $p, q > 1$ tais que $pq = N$.

E o problema do RSA é dado por: dados o inteiro N, uma mensagem m e um inteiro e relativamente primo com $\phi(N)$, definimos $Enc_e(m) = m^e \pmod N \equiv y \pmod N$ como a cifragem de m a partir desses parâmetros; o grande problema do RSA é o cálculo de

$y^{e^{-1}} \pmod{N}$ sem saber a fatoração de N [14]. Assim, a cifra RSA pode ser obtida com as seguintes funções em R:

```
library(dplyr)
library(FRACTION)

inv_mod1<-function(x,m){
  resto<-0
  j<-1
  while(resto!=1){
    resto <- (x*j) %% m
    j<-j+1
  }
  j<-j-1
  return(j)
}

rel_primos<-function(m){
  l<-c()
  for(j in 1:m){
    if(gcd(m,j)==1){
      l<-c(l,j)
    }
  }
  return(l)
}

exp_mod4<-function(base,expdec,modulo){
  expbin<-as.integer(strsplit(zera8_(intToBin(expdec)),split = "")[[1]])
  f<-1
  for(j in 1:(length(expbin)-1)){
    if(expbin[j]==1){
      f<-(f*base) %% modulo
    }
    f<-(f^2) %% modulo
  }
  if(last(expbin)==1){
    f<-(f*base) %% modulo
  }
  return(f)
}
```

Listing 3.3: Código fonte em R da Cifra RSA

A função principal é a *exp_mod4* que recebe como argumentos o inteiro referente a um trecho da mensagem como base, a chave pública como expoente e o número resultado

do produto de dois números primos como módulo. A cifra dessa mensagem ou trecho de mensagem sai do resultado dessa potência. Porém o algoritmo foi feito de forma eficiente para que não gere valores intermediários muito grandes e que seja executado de forma rápida, conforme as dicas dadas por [12] e [13]. Essas dicas buscam decompor o expoente em seus dígitos binários para efetuar menos multiplicações.

A função auxiliar *rel_primos* lista os números positivos relativamente primos com um determinado inteiro positivo, que sejam menores que ele. O objetivo dela é fornecer opções de chave pública que seja um número relativamente primo com o tociente de n, conforme premissa. Já a *inv_mod*1 busca encontrar o inverso multiplicativo de um número dentro do corpo $\mathbb{Z}_n$.

Como exemplo, considere que a cifragem é feita por caracter e que por isso o valor binário de cada item a ser cifrado é limitado pelos 256 itens da tabela ASCII, ou seja 2^8. Assim, o número n deve estar entre 2^8 e 2^9, entre 256 e 512, respectivamente. Suponha, então que os dois primos escolhidos são $p = 13$ e $q = 29$, cujo produto é $n = 377$. Logo $\phi(n) = 336$. Usando a função *rel_primos*, conseguimos listar os números relativamente primos com 336, e escolhemos um deles, $e = 199$ como chave pública. Em seguida, usando a função *inv_mod*1, descobrimos o valor do inverso de e dentro de $\mathbb{Z}_{336}$, que chamamos de $d = 103$. Desse modo, a chave privada é dada por $d = 103$.

Fixados os parâmetros, faz-se um teste com um caracter, por exemplo, **f**, que é representado na tabela ASCII pelo número 102. Aplicando a função *exp_mod*4 aos argumentos dados $(msgasc = 102, e = 199, n = 377)$, encontramos o valor de cifra 301. E usando novamente a função *exp_mod*4 a cifra encontrada, e a chave privada no lugar da pública $(msgasc = 301, e = 103, n = 377)$ retornamos ao valor do texto claro 102. Tais passos estão explicitados no código B.1 no Apêndice B.

O algoritmo de geração de chaves *Gen* deve ser implementado para buscar minimizar a chance de um atacante descobrir a chave usada por meio de alguma técnica de criptoanálise a partir de textos cifrados com chave repetida. Caso usemos a mesma chave para o exemplo acima, mas com um texto no lugar da mensagem, ou seja, cifrando caracter a caracter com a mesma chave; uma análise de frequência simples dos caracteres gerados pode ser capaz de obter a chave usada. Outra técnica usada é a resolução de sistemas lineares por meio de cifragem de textos planos escolhidos para descobrir os parâmetros usados (inclusive as chaves), [6].

Dessa forma, seguindo [14], optou-se por implementar um algoritmo de geração de chaves para chaves seguindo um determinado padrão dentro de um espaço de chaves $\mathcal{K}$. Esse padrão vai depender da quantidade de bits da mensagem que se quer cifrar com a chave.

```
1 binario_decimal <- function(bits){
```

```
  tbits<-length(bits)
  j<-tbits-1
  decimal<-0
  for(b in bits){
    decimal<-decimal + b*2^j
    j<-j-1
  }
  return(decimal)
}

concatbits1<-function(bits){
  novo<-1
  caracteres<-length(bits)
  for(j in 1:caracteres){
    novo<-paste0(novo,bits[j])
  }
  return(novo)
}

gerachave3 <- function(tbits){
  bits_aleat<-round(runif(tbits-1))
  chavebin<-concatbits1(bits_aleat)
  chavealeat<-binario_decimal(bits_aleat)
  return(c(chavebin,chavealeat))
}

gera_parametros_RSA2<- function(tbits){
  n_aleat<-gerachave3(tbits)
  primos<-Primes(sqrt(as.numeric(n_aleat[2])))
  p<-last(primos)
  q<-tail(primos)[1]
  toc<-(p-1)*(q-1)
  n<-p*q
  e<-tail(rel_primos(toc))[1]
  d<-inv_mod1(e,toc)
  return(cbind(p,q,n,toc,e,d))
}
```

Listing 3.4: Código fonte em R do algoritmo de geração de chaves para a Cifra RSA

Assim, a função *gera_parametros_RSA2* tem como argumento apenas a quantidade de bits do número do módulo N a ser gerado. E tem como saídas, além de N, seus dois fatores primos p e q, a sua função tociente, o valor da chave pública e e o valor da chave privada d. São evitados números primos pequenos nesse algoritmo como forma de fortalecer a segurança da cifra. Um primeiro número aleatório com a quantidade indicada

de bits é gerado por meio da função *gerachave*3, que gera binários com quantidade definida de dígitos, com as funções auxiliares *concatbits*1 e *binario_decimal*.

3.3.2 ECC

A cifra usando criptografia de curvas elípticas precisa de diversas funções auxiliares, que serão detalhadas. A partir dessas funções, se consegue gerar as curvas elípticas com parâmetros iniciais.

```
library(dplyr)
library(reshape2)

lambda_dobro <- function(x,y,a,m){
  if(y!=0){
    num<-(3*x^2+a) %% m
    den<-(2*y) %% m
    l<-(num*inv_mod1(den,m))%%m
    return(l)
  }else{
    return("erro divisao por zero")
  }
}

lambda_soma <- function(x1,y1,x2,y2,m){
  num<-(y2-y1) %% m
  den<-(x2-x1) %% m
  l<-(num*inv_mod1(den,m))%%m
  return(l)
}

pt_soma <- function(x1,y1,x2,y2,a,m){
  if(x1==x2 & y1==y2){
    l<-lambda_dobro(x1,y1,a,m)
  }else{
    l<-lambda_soma(x1,y1,x2,y2,m)
  }
  xr <-(l^2 -x1 -x2) %% m
  yr <-(l*(x1-xr) -y1) %% m
  return(c(xr,yr))
}

pt<-function(primo,a,b){
  condicaoexistencia <- ((4*a^3 + 27*b^2) %% primo) != 0
  if(condicaoexistencia){
    seqx <- 0:(primo-1)
```

```
    seqy2 <- (seqx^3 + a*seqx + b) %% primo
    quadrados <- seqx^2 %% primo
    names(quadrados)<-seqx
    posicoes <- seqy2 %in% quadrados
    seqx_<-seqx[which(posicoes)]
    seqy2_<-seqy2[which(posicoes)]
    l2<-list()
    l1<-list()
    raiz1<-list()
    raiz2<-list()
    for(k in 1:length(quadrados)){
      if(quadrados[k] %in% l2){
        l1 <- c(l1, quadrados[k])
        raiz1 <- c(raiz1,names(quadrados[k]))
      }else{
        l2 <- c(l2, quadrados[k])
        raiz2 <- c(raiz2,names(quadrados[k]))
      }
    }
    l1_<-unlist(l1)
    raiz1_<-unlist(raiz1)
    rz1<-data.frame(l1_,raiz1_)
    rz1$raiz1_<-as.numeric(rz1$raiz1_)
    rz1<-rz1[order(rz1$l1_),]
    l2_<-unlist(l2)
    raiz2_<-unlist(raiz2)
    rz2<-data.frame(l2_,raiz2_)
    rz2$raiz2_<-as.numeric(rz2$raiz2_)
    rz2<-rz2[order(rz2$l2_),]
    rz2_<-rz2
    row.names(rz2_)<-NULL
    rz1_<-rz1
    row.names(rz1_)<-NULL
    rz1_<-rbind(c(0,0),rz1_)
    rz<-cbind.data.frame(rz2_,rz1_)
    rz<-rz[,-3]
    names(rz)<-c("x","rx1","rx2")
    df<-data.frame(seqx_,seqy2_)
    seqy<- df %>% left_join(rz, by = c("seqy2_"="x"))
    seq_<-seqy[,-2]
    pts<-melt(seq_, id.vars = "seqx_")[-2]
    pt<-pts[order(pts$seqx_),]
    row.names(pt)<-NULL
    names(pt)<-c("x","y")
    pt1<-distinct(pt)
```

```
    return(pt1)
  }else{
    return("âParmetros áinvlidos")
  }
}

qtd_ptos<-function(pt){
  return(nrow(pt))
}

multiplo <- function(k,x,y,a,m,qtd_ptos){
  if(k==1){
    return(c(x,y))
  }else{
    if(k>1 & k<qtd_ptos+1){
      xr <- x
      yr <- y
      for(i in 1:(k-1)){
        pr <- pt_soma(x,y,xr,yr,a,m)
        xr <- pr[1]
        yr <- pr[2]
        if(xr==x & i<k-1){
          return("multiplo inexistente")
          break
        }
      }
      return(c(xr,yr))
    }else{
      return("multiplo inexistente")
    }
  }
}
```

Listing 3.5: Código fonte em R da Cifra de Curvas Elípticas

A função *pt* tem como argumentos os parâmetros da curva elíptica conforme a equação 2.4 e o número primo associado. A condição de existência 2.5 é testada logo no início do algoritmo. A partir dos possíveis valores de abcissas dos pontos da curva, são gerados quadrados das ordenadas correspondentes seguindo a equação 2.4. Os passos seguintes são para calcular as raízes desses quadrados (os resíduos quadrados) e listar os pontos da curva com ambas as coordenadas. A saída da função é a lista com os pontos da curva elíptica associada aos argumentos dados na entrada, com exceção do ponto no infinito.

O cálculo da ordem da curva elíptica (sem contar o ponto no infinito) conforme definição 3 é dada pela função *qtd_pontos*. E a função *multiplo* calcula o múltiplo de um

determinado ponto base de uma curva elíptica a partir das coordenadas do ponto, de parâmetros da curva e de sua ordem.

Por fim, a soma de pontos dentro de uma curva elíptica é dada pela função *pt_soma* que toma como argumentos as coordenadas dos dois pontos a serem somados, assim como o parâmetro a e o número primo associado à curva. Essa função usa outras funções relativas ao cálculo do λ conforme equação 2.10, que, por sua vez precisam da função *inv_mod*1 para efetuar a inversão de números ao calcular divisões por meio de multiplicações.

A título de exemplo, suponha que, dados um grupo elíptico $E_q(a, b) = E_{751}(-1, 188)$ e um ponto base $G = (0, 376)$ de um sistema de criptografia de curva elíptica, exemplo retirado de [12] e [28]. Alice quer enviar uma mensagem para Bob. A mensagem está codificada em $P_m = (562, 201)$. Alice gera o valor aleatório $k = 386$ como sua chave privada. A chave pública de Bob está no ponto $P_B = (201, 5)$. A seguir estão os passos a serem seguidos por Alice para gerar a cifra no algoritmo 3.5 em R:

1. Gera a chave pública dela multiplicando sua chave privada pelo ponto base, usando a função *multiplo* com os argumentos dados e obtendo $P_A = (676, 558)$.

2. Calcula o produto de sua chave privada pela chave pública de Bob, também usando a função *multiplo* obtendo $k * P_B = (239, 377)$.

3. Somando o ponto da mensagem ao ponto calculado no item anterior obtendo $P_m + k * P_B = (385, 328)$

Esses passos estão explicitados no código B.3 no Apêndice B. O resultado é o par de pares ordenados $\{(676, 558), (385, 328)\}$, em que o primeiro par é a chave pública de Alice e o segundo é a mensagem cifrada.

É importante também ter um algoritmo de geração de parâmetros e chaves *Gen* para o ECC. A implementação feita conta com a geração de um número primo aleatório a partir de uma quantidade de bits dada, assim como os parâmetros a, b, ordem da curva e ponto base (gerador). A partir dos parâmetros gerados, pode-se gerar chaves privadas e chaves públicas associadas.

```
gera_parametros_ECC1<- function(tbits){
  primo<-last(Primes(as.numeric(gerachave3(tbits)[2])))
  ordem<-0
  condicaoexistencia<-FALSE
  j<-1
  while(!IsPrime(ordem)|!condicaoexistencia|j<10){
    a<-round(runif(1)*primo)
    b<-round(runif(1)*primo)
    condicaoexistencia <- ((4*a^3 + 27*b^2) %% primo) != 0
    pt1<-pt(primo,a,b)
```

```
11      ordem <- qtd_ptos(pt1)+1
12      pt_base_x <- pt1[1,1]
13      pt_base_y <- pt1[1,2]
14      j<-j+1
15    }
16    if(j==10){
17      if(!IsPrime(ordem)){
18        return("sem sucesso")
19      }else{
20        if(condicaoexistencia){
21          return(cbind(a,b,primo,ordem,pt_base_x,pt_base_y))
22        }else{
23          return("sem sucesso")
24        }
25      }
26    }else{
27      return(cbind(a,b,primo,ordem,pt_base_x,pt_base_y))
28    }
29  }
30
31  gera_chaves_ecc<-function(a,b,primo,ordem,ptx,pty){
32    chpriv<-round(ordem*runif(1))
33    chpubl<-multiplo(chpriv,ptx,pty,a,primo,qtd_ptos(pt(primo,a,b)))
34    return(cbind(chpriv,chpubl[1],chpubl[2]))
35  }
```

Listing 3.6: Código fonte em R do algoritmo de geração de chaves para a Cifra ECC

A função *gera_parametros_ECC*1 foi formulada a partir da sugestão de [14] quanto ao fato da ordem da curva elíptica gerada ser um número primo. Essa ideia decorre de que, nessa situação, qualquer ponto diferente do ponto infinito da curva pode ser usado como ponto base, pois o grupo cíclico gerado a partir desse ponto vai ter a mesma ordem do grupo elíptico finito. O que garante isso é o Teorema 7 (Lagrange) do Apêndice A, conforme [29].

Retomando ao nosso caso, o subgrupo gerado por um ponto não-infinito da curva elíptica tem ordem que divide a ordem da curva elíptica. Se a ordem da curva elíptica for um número primo, então a ordem de qualquer subgrupo gerado por um ponto não-infinito da curva é a mesma ordem da curva, isso quer dizer que todos os elementos podem ser gerados por um ponto qualquer não-infinito escolhido como gerador da curva.

A condição de existência dada pela Equação 2.5 também é verificada na função *gera_parametros_ECC*1, assim como um limite de tentativas para evitar que o programa entre em *looping* infinito. Já a geração de chave privada é feita por meio da função *gera_chaves_ecc* que sorteia um número de 1 até o número da ordem da curva elíptica

gerada, sendo a chave pública também gerada pelo múltiplo do ponto base com a chave privada.

3.4 Comparação e Resultados

A comparação entre os métodos testados busca saber sob que aspectos um método pode ser melhor que outro. Em todo caso, não cabe comparar métodos de criptografia simétrica com métodos de criptografia assimétrica. A essência básica de ambas já se diferencia logo no início.

Retomando a primeira questão de pesquisa, **QP1: Como podemos comparar modelos criptográficos?** Uma forma, citada por [15], considera o tamanho de chave a ser usada mantendo-se contante a segurança fornecida. Para usar esse aspecto, precisa-se saber como manter constante a segurança fornecida em cada método.

Vamos optar por comparar a complexidade dos métodos, dados os seus algoritmos implementados. Para isso, recorreu-se a fundamentos de análise assintótica de algoritmos e complexidade computacional baseados em [30].

Comparando alguns aspectos do RSA com ECC, temos que o processo de geração de chaves do RSA precisa de um valor de entrada de tamanho equivalente para gerar uma chave do mesmo número de bits que a gerada pelo ECC. Ou seja, se usarmos $tbits = 8$ em $gera\_parametros\_RSA2$, é gerado um número aleatório cuja forma binária tem 8 bits. Em seguida é gerada uma lista de primos de 2 até a raiz quadrada do número gerado e pegado dois números primos da parte final dessa lista (cada um desses dois primos tem por volta de 4 bits), de forma que o produto entre eles seja um número cuja forma binária tenha aproximadamente 8 bits também, assim como a função tociente dele. Com isso foram gerados n, p, q e $\phi(n)$. A chave pública e é dada por um dos últimos números relativamente primos com $\phi(n)$. E a chave privada d sai pela inversão da chave pública e dentro do anel $\mathbb{Z}_{\phi(n)}$. Com isso temos que a chave pública também é um número com forma binária de aproximadamente 8 bits, mas a chave privada pode ter menos bits.

Na ECC, se colocarmos o argumento $tbits = 8$ em $gera\_parametros\_ECC1$, var ser gerado um número primo m de mesma magnitude a fim de fixar o corpo base $\mathbb{F} = \mathbb{Z}_m$. Em seguida são testados valores aleatórios para os parâmetros a e b de modo a atender a condição de não singularidade 2.5 e, ao mesmo tempo, ter um número primo como ordem da curva elíptica gerada Ω. Observando o teorema 1, a ordem da curva elíptica gerada tem grandeza próxima do número primo m gerado, ou seja, 8 bits.

Podemos citar a análise assintótica como critério de comparação entre modelos e vantagem do modelo mais complexo em termos de maior segurança, continuando a comparação em resposta a QP1, e também respondendo a **QP2: Que outras vantagens**

Tabela 3.1: Comparação entre os modelos

Características	RSA	ECC
Problema base	Fatoração	Logaritmo Discreto
Geração de chaves	$O(n \log n)$	$O(n^2)$
Execução	$O(n)$	$O(n^2)$

podemos obter ao usar a criptografia de curvas elípticas no lugar de outras técnicas de criptografia assimétrica?. Ou seja, quanto à complexidade das funções usadas, verificou-se que a *gera_parametros_RSA*2 tem tempo de execução da ordem $O(tbits \log tbits)$, fazendo a análise pelo pior caso, sendo que a sua principal operação básica é calculada dentro da função *rel_primos* cuja complexidade é influenciada pela operação de máximo divisor comum entre números para verificação de números relativamente primos. Enquanto que a função *gera_parametros_ECC*1 tem complexidade $O(tbits\sqrt{tbits})$, por conta da complexidade do algoritmo de geração de números primos dado pelo Crivo de Eratóstenes. No entanto, para se obter as chaves da ECC, é necessário executar também a função *gera_chaves_ecc* que assintoticamente leva um tempo de execução de $O(ordem * primo)$. Desse modo, nota-se que a complexidade para geração de chaves é maior para o caso da ECC em comparação com o RSA.

Após as gerações de chaves, podemos comparar a execução das duas técnicas. O RSA usa basicamente a função *exp_mod*4 que tem custo linear em relação ao valor do expoente dado pela chave pública ou privada dentro da exponenciação modular. Já a ECC depende das funções *multiplo* e *pt_soma* para cifrar mensagens, e os custos envolvidos são da ordem do produto entre a chave privada e o número primo usado para a primeira função (*multiplo*) que usa a segunda função (*pt_soma*) internamente para acumular somas até encontrar um múltiplo de um ponto base. Isso mostra que a execução das cifragens tem complexidade maior na ECC que no RSA. A tabela 3.1 resume essas informações quanto às duas técnicas.

Quanto ao grau de uso de cada técnica no Brasil, não foram encontradas fontes para essa informação, o que constitui falta de evidência para algumas afirmações. Desse modo, ficaram sem possibilidade de resposta as duas últimas questões de pesquisa **QP3: Por que esse método não é tão difundido no Brasil?** e **QP4: Quais entraves para a maior utilização de ECC no Brasil?**

Existe um órgão público chamado Instituto Nacional de Tecnologia da Informação (ITI) vinculado à Casa Civil da Presidência da República. O ITI tem por missão manter e executar as políticas da Infraestrutura de Chaves Públicas Brasileira – ICP-Brasil [31]. Ao ITI compete ainda ser a primeira autoridade da cadeia de certificação digital (AC Raiz). No entanto, não foram encontradas informações a respeito das técnicas usadas

como criptografia no sítio do ITI sobre certificação ou outra aplicação qualquer. Sangalli [32] menciona que a própria sofisticação e complexidade da técnica como um empecilho para o seu maior uso.

Somente com a informação do grau de uso das diversas técnicas no Brasil, seria possível validar ou refutar a tese de que a *técnica de ECC é pouco difundida no Brasil* e buscar uma investigação sobre os possíveis fatores, caso a tese tenha sido validada.

3.5 Limitações da Pesquisa

Aqui listamos as limitações do trabalho, as limitações da pesquisa sob a ótica da metodologia usada, dos casos abordados, e das generalizações feitas. Pretende-se com isso explicitar que a pesquisa não é exaustiva e quaisquer generalizações podem ser perigosas se baseadas apenas no desenvolvimento desse trabalho.

A metodologia considerou como o estudo de caso baseado no caso-base *ECC* comparado ao caso *RSA*. Evidentemente, a criptografia assimétrica vai além dessas duas técnicas. E outras técnicas poderiam ter sido usadas no estudo. Desse modo, deve-se tomar cuidado com generalizações. Além disso, a metodologia de pesquisa não seguiu todos os passos do estudo de caso *stricto sensu*, mas alguns deles.

As implementações feitas e apresentadas podem ter um viés de prática de programação não eficiente. Dessa forma, a refatoração dos códigos pode gerar resultados diferentes. Uma possibilidade a ser abordada, por exemplo, é o uso da teoria de Reciprocidade Quadrática, vista em [33], para calcular raízes quadradas dentro dos anéis $\mathbb{Z}_m$ na implementação da *ECC*, que não foi usada. Ainda assim, algumas funções próprias de bibliotecas usadas certamente usam técnicas de programação diferente e isso deve ser considerado. Inclusive seria interessante verificar o comportamento de tais implementações em outra linguagem.

Não foram encontrados na indústria, aplicações existentes de criptografia assimétrica. E também não obtivemos uma lista de vantagens da *ECC* sobre outras técnicas. E a literatura usada quanto ao tema do trabalho apresenta uma limitação da consulta feita, ou seja, outras referências poderiam ter sido consultadas e isso deixaria o trabalho mais rico.

Sobre a estratégia de aplicação da teoria, foi usada apenas a cifragem. Outras formas de aplicação poderiam ter sido tentadas, como a assinatura digital e a certificação.

3.6 Síntese do Capítulo

Nesse capítulo foi apresentado o desenvolvimento do trabalho em termos de implementação e análise. A linguagem escolhida foi colocada assim como as suas características. A criptografia simétrica foi exemplificada por meio de duas técnicas: a Cifra de César e a Cifra de Vigenère. A criptografia assimétrica foi implementada com o clássico modelo RSA, assim como o modelo ECC, que é o principal tema desse trabalho. Todos os códigos desenvolvidos foram explicitados e analisados. Ao final foi feita uma comparação em termos de complexidade dos métodos de criptografia assimétrica abordados. As questões de pesquisa foram discutidas e as limitações do trabalho foram apresentadas.

No capítulo seguinte, vamos apresentar as conclusões do trabalho. Além disso vamos indicar trabalhos futuros de interesse sobre tópicos e temas que deixariam esse trabalho mais completo.

Capítulo 4

Conclusão

A área da criptografia, principalmente a assimétrica, tornou-se um vasto campo de aplicação de Teoria dos Números entre as vertentes da matemática. O avanço da computação com a velocidade cada vez maior de processamento abre muitos caminhos para desenvolvimento de novas cifras com complexidade cada vez maior, com o objetivo de se tornar menos vulnerável a possíveis ataques.

Esse trabalho fez uma breve exposição e aplicação da Criptografia de Curvas Elípticas mostrando desde os fundamentos que a embasam, até o código de implementação em linguagem R. Apesar de mais simples, a criptografia simétrica ainda é bem utilizada para situações simples, em que se exige pouca segurança. Para situações em que se envolve mais interesse de sigilo, por exemplo, usa-se a criptografia assimétrica.

A geração ou *derivação* de chaves é uma etapa importante de qualquer criptografia e, nesse trabalho, foi mostrada a forma de fazer isso aleatoriamente de acordo com o tamanho em bits que se deseja. A comparação feita entre o RSA e a ECC mostrou que a segunda opção tem complexidade maior. Infere-se, portanto, que há uma segurança maior ao se usar a ECC em comparação com o RSA para mesmo tamanho de chave. Isso também pode ser interpretado como a necessidade de um tamanho menor de chave na ECC para obter a mesma segurança do RSA, conforme [15].

Também pode-se fazer a mesma interpretação a respeito da execução de ambos os algoritmos RSA e ECC. Por ter uma maior complexidade, a ECC pode fornecer maior segurança que o RSA. Ou a execução da ECC pode ser feita com tamanho de chave menor e oferecendo a mesma segurança que o RSA.

Não foram encontrados trabalhos relatando diretamente sobre o grau de uso de ECC no Brasil.

4.1 Trabalhos Futuros

São várias as possibilidades para continuação desse trabalho. Reúne-se aqui algumas propostas, tanto explorando as limitações de pesquisa já listadas, como indo além do escopo inicial do tema ECC.

Um exercício que fica para trabalho futuro é a simulação de testes de criptoanálises sob textos cifrados de ambas as técnicas, RSA e ECC. Dessa forma, caso a tese se comprove, a conclusão ficaria mais completa a respeito da maior segurança da ECC sobre o RSA.

O aspecto de assinatura eletrônica, que precisaria de uma função *hash* para converter um texto de qualquer tamanho em uma cadeia de caracteres de tamanho fixo, também é um bom tema para complementar esse trabalho, continuando a comparação entre ECC e RSA.

Consideramos aqui apenas as cifragens de textos com criptografia simétrica e de números com criptografia assimétrica. A implementação da cifragem de textos ou imagens em R das técnicas RSA e ECC também deve ser considerada como um continuação desse trabalho. Inclusive não foram encontrados textos sobre a parte prática de codificação ao longo de uma cadeia de caracteres de modo que a cifra não seja uma mera repetição e não se torne alvo fácil de uma criptoanálise por frequência de letras do alfabeto.

A comparação com outras técnicas de criptografia assimétrica é uma possibilidade de caminho a ser percorrido, inclusive as variantes mistas que envolvem mais de uma técnica. A verificação de qual técnica de criptografia é mais difundida no Brasil também ficou faltando e é um excelente tema para um próximo trabalho.

Apêndice A

Ferramental Matemático

Este capítulo complementar foi feito para elucidar conceitos básicos e resultados importantes usados no desenvolvimento do trabalho. Vamos listar definições, notações, convenções e resultados sem se preocupar com rigorosas demonstrações.

A.1 Aritmética

O conteúdo dessa seção foi retirado em grande parte de Shokranian [33]. Vamos nos limitar aqui apenas aos números inteiros $\mathbb{Z} = \{..., -4, -3, -2, -1, 0, 1, 2, 3, 4, ...\}$ ou aos números naturais $\mathbb{N} = \{1, 2, 3, 4, ...\}$.

Dentro dos números naturais, dizemos que um número b *divide* um número a se, e somente se, existir um único número c tal que $a = bc$. Quando b divide a, dizemos que b é um *divisor* de a. E o número c é chamado *resultado da divisão* e também é um divisor de a, pelo mesmo motivo que b. Quando b divide a, usamos a notação $b \mid a$. Caso b não divida a, usamos a notação $b \not| a$. Quando $b \mid a$, dizemos que a é divisível por b.

O conjunto dos divisores de um número natural a é definido por $\text{Div}(a) = \{x \in \mathbb{N} \text{ tal que } x \mid a\}$. Como todo número é divisível por 1 e por ele mesmo, então $\text{Div}(a)$ nunca é vazio, pois contém pelo menos 1 e a.

Um número natural $p > 1$ é dito *número primo* se e somente se, ele possui apenas 1 e p como divisores naturais. Ou seja, p é número primo se e somente se $\text{Div}(p) = \{1, p\}$.

Para dois números naturais a e b, dizemos que o *máximo divisor comum* (mdc) entre eles é definido como o maior número natural que divide ambos. Em particular, dizemos que a e b são *relativamente primos*, ou primos entre si, quando $mdc(a, b) = 1$.

Dados $a, b \in \mathbb{Z}$ com $b > 0$, então o *Algoritmo de Euclides* nos diz que existe um único inteiro q e um único inteiro r tal que $a = bq + r$, com a condição $0 \leq r < b$. O número r é chamado *resto da divisão de Euclides*.

Vamos denotar por $\mathcal{P}$ o conjunto dos números primos. A seguir temos três importantes teoremas envolvendo números primos.

Teorema 2 (Primalidade e fatoração)**.** *Todo número natural $n > 1$ ou é primo, ou é divisível por um número primo.*

Teorema 3 (Infinidade)**.** *$\mathcal{P}$ é um conjunto infinito.*

Teorema 4 (Teorema Fundamental da Aritmética)**.** *Todo número composto $n > 1$ pode ser representado de uma única forma como produto $n = p_1^{\alpha_1} p_2^{\alpha_2} ... p_k^{\alpha_k}$ com números primos $p_1 < p_2 < ... < p_k$ e $\alpha_1, \alpha_2, ..., \alpha_k$ números naturais.*

Um método bem conhecido e antigo para listar os números primos até determinado ponto é o *crivo de Eratóstenes*. Ele funciona da seguinte maneira: Considere uma lista de números de 2 até n e siga os seguintes passos:

1. Retire o segundo número a partir de 2 e assim por diante:

$$2, 3, \cancel{4}, 5, \cancel{6}, 7, \cancel{8}, 9, \cancel{10}, 11, \cancel{12}, 13, \cancel{14}, 15, \cancel{16}, 17....$$

2. Retire o terceiro número a partir de 3 e assim por diante:

$$2, 3, \cancel{4}, 5, \cancel{6}, 7, \cancel{8}, \cancel{9}, \cancel{10}, 11, \cancel{12}, 13, \cancel{14}, \cancel{15}, \cancel{16}, 17....$$

Os passos devem continuar a cada número primo, como $5, 7, 11..$ (os primeiros não cancelados) até que alcancemos um número primo próximo de $\sqrt{n}$. Quando fizer isso, todos os números compostos (números não-primos) estarão cancelados e na lista existirão apenas números primos menores ou iguais a n.

Suponha que $m > 1$ é um número inteiro fixo até o final dessa seção. Dizemos que dois números inteiros são *congruentes módulo m* se, e somente se, m divide a diferença entre eles. A congruência módulo m entre dois números inteiros a e b é denotada por $a \equiv b \pmod{m}$. Quando a e b não são congruentes, dizemos que $a \not\equiv b \pmod{m}$.

A relação de congruência é uma relação de equivalência pois atende às seguintes propriedades (dados quaisquer $a, b, c \in \mathbb{Z}$):

1. $a \equiv a \pmod{\text{m}}$
2. $a \equiv b \pmod{\text{m}} \Rightarrow b \equiv a \pmod{\text{m}}$
3. $a \equiv b \pmod{\text{m}} \wedge b \equiv c \pmod{\text{m}} \Rightarrow a \equiv c \pmod{\text{m}}$

Dado um número inteiro a, chamamos classe de congruência determinada por a, módulo m, o subconjunto $\overline{a}$ dos números que são congruentes com a, módulo m. Definimos como conjunto quociente ou conjunto das *classes de resto módulo m* o conjunto das classes de congruência, dado por $\{0, 1, 2, ..., m-1\}$. Podemos denotar esse conjunto por $\mathbb{Z}_m$.

Dessa forma, vamos descrever o teorema de Fermat, a definição da função tociente de Euler e o teorema de Euler:

Teorema 5 (Fermat)**.** *Se p é número primo e a é número inteiro positivo não divisível por p, então*

$$a^{p-1} \equiv 1 \ (mod\ p) \tag{A.1}$$

A função tociente de Euler [12], denotada por $\phi(n)$, é definida como a quantidade de inteiros positivos menores que n e relativamente primos com n. Por convenção, temos que $\phi(1) = 1$. Observe que $\phi(6) = 2$ e $\phi(11) = 10$. Essa função tem a propriedade de que, para p primo, vale $\phi(p) = p - 1$. Outra propriedade importante: quando temos dois números primos p e q, com $p \neq q$, então $\phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$. Agora podemos enunciar o teorema de Euler:

Teorema 6 (Euler)**.** *Para cada a e n que sejam relativamente primos:*

$$a^{\phi(n)} \equiv 1 \ (mod\ n) \tag{A.2}$$

A.2 Álgebra

Essa seção foi retirada em grande parte de Domingues [29]. Vamos definir as estruturas algébricas de grupos, anéis e corpos assim como algumas de suas propriedades.

Definição 4 [Grupo] Um sistema constituído de um conjunto não vazio G e uma operação $(x, y) \mapsto x * y$ sobre G é chamado de *grupo* se essa operação se sujeita aos seguintes axiomas:

- $(a * b) * c = a * (b * c)$, quaisquer que sejam $a, b, c \in G$
- Existe um elemento $e \in G$ tal que $a * e = e * a = a$, qualquer que seja $a \in G$
- $\forall a \in G$ existe um elemento $a' \in G$ tal que $a * a' = a' * a = e$

Caso o grupo cumpra o axioma $a * b = b * a$, quaisquer que sejam $a, b \in G$, dizemos que o grupo é abeliano (comutativo).

A operação $*$ sob a qual o grupo é definido pode ser definida de qualquer forma, mas se a operação for uma *adição*, dizemos que o grupo é aditivo, assim como vamos chamar de grupo multiplicativo aquele em que a operação seja uma *multiplicação*.

Um grupo $(G, *)$ em que o conjunto G é finito, chama-se *grupo finito*. Nesse caso, o número de elementos de G é chamado *ordem* do grupo, denotada por $\#G$.

Definição 5 [Subgrupo] Seja $(G, *)$ um grupo. Diz-se que um subconjunto não vazio $H \subset G$ é um subgrupo de G se:

- H é fechado para a operação $*$
- $(H, *)$ também é um grupo

Dado um grupo aditivo G, se $a \in G$ e $m \in \mathbb{Z}$, o m-ésimo múltiplo de a é denotado por $m * a$ e definido por:

- se $m \geq 0$, por recorrência, da seguinte forma:

$$0 * a = e \ \text{ (elemento neutro de } G)$$

$$m * a = (m-1) * a + a \ \text{ (se } m >= 1)$$

- se $m < 0$

$$m * a = -[(-m) * a]$$

Denota-se por $[a]$ o subconjunto de G formado pelos múltiplos inteiros de a, ou seja $[a] = \{m * a \mid m \in \mathbb{Z}\}$. Um resultado importante mostra que o subconjunto $[a]$ é um subgrupo de G.

Um grupo G é chamado *grupo cíclico* se, para algum elemento $a \in G$, se verificar a igualdade $G = [a]$. Nessas condições, o elemento a é chamado *gerador* do grupo G. Pode-se mostrar que todo subgrupo de um grupo cíclico é também cíclico.

Dado um grupo aditivo G e um subgrupo arbitrário H. Considere a relação de equivalência dada por $a \approx b$ se, se somente se, $-a + b \in H$. Pode-se provar que, se $a \in G$, então a classe de equivalência determinada por a é o conjunto $a + H = \{a + h; h \in H\}$. Dizemos que essa classe de equivalência $a + H$ é chamada classe lateral à direira, módulo H, determinada por a. O conjunto quociente de G por essa relação, denotado por G/H, é o conjunto das classes laterais $a + H$, com $a \in G$. Um dos elementos desse conjunto é o próprio H, pois $H = 0 + H$.

Uma proposição que pode ser mostrada é: Seja H um subgrupo de G. Então duas classes laterais quaisquer módulo H são subconjuntos de G com a mesma cardinalidade

(número de elementos). Em particular, todas as classes têm a mesma cardinalidade de H. Se G é um grupo finito, então o conjunto G/H também é finito. Nesse caso, dizemos que $(G : H)$ é o índice de H em G, ou seja, o número de elementos do conjunto quociente G/H.

Teorema 7 (Lagrange)**.** *Seja H um subgrupo de um grupo finito G. Então $\#G = \#H(G : H)$ e, portanto, $\#H|\#G$.*

Para ilustrar, vamos mostrar em um pequeno exemplo. Seja G o grupo aditivo $\mathbb{Z}_6 = \{0, 1, 2, 3, 4, 5\}$. Considere o subgrupo $H = \{0, 3\}$, temos que: $0 + H = \{0, 3\}$, $1 + H = \{1, 4\}$ e $2 + H = \{2, 5\}$. Nota-se que qualquer classe $a + H$ tem cardinalidade 2 e o índice de H em G é 3, que é a quantidade de classes laterais distintas. Assim $\#G = 6 = 2 * 3 = \#H(G : H)$ como diz o teorema de Lagrange.

Podemos enunciar a definição do problema do logaritmo discreto a partir de Chen [7]:

Definição 6 [Problema do Logaritmo Discreto] Seja G um grupo cíclico multiplicativo de ordem p (primo) e g o gerador de G. Suponha que a seja um elemento desconhecido, tal que $a \in \mathbb{Z}_p^*$, dado que se conhece os valores de $g, g^a \in G$ como entrada. O *Problema do Logaritmo Discreto* é descobrir o valor de a.

Definição 7 [Anel] Um sistema constituído de um conjunto não vazio A e um par de operações sobre A, respectivamente uma adição $(x, y) \mapsto x + y$ e uma multiplicação $(x, y) \mapsto xy$, é chamado de *anel* se:

- $(A, +)$ é um grupo abeliano
- $(ab)c = a(bc)$, quaisquer que sejam $a, b, c \in A$
- $a(b + c) = ab + ac$ e $(a + b)c = ac + bc$, quaisquer que sejam $a, b, c \in A$

Caso o anel A obedeça a propriedade comutativa para a multiplicação, isto é, se $ab = ba$ para quaisquer $a, b \in A$, então diz-se que A é anel comutativo. Se A tiver um elemento neutro para a multiplicação, isto é, se existir um elemento $1_A \in A$, $1_A \neq 0_A$, tal que $a * 1_A = 1_A * a = a$ qualquer que seja $a \in A$, então diz-se que A é um anel com unidade.

Definição 8 [Domínio de Integridade] Seja A um anel comutativo com unidade. Se para esse 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 diz-se que A é um *domínio de integridade*.

Vamos usar a notação $U(A)$ para indicar o conjunto dos elementos do anel comutativo com unidade A que tem inverso multiplicativo. $U(A)$ nunca é vazio (pois a unidade tem seu inverso que é ela mesma), mas também nunca inclui o zero.

Definição 9 [Corpo] Seja K um anel comutativo com unidade. Se $U(K) = K^* = K - \{0\}$, então K é chamado de *corpo*.

Por fim, dado A um anel, suponha que, para algum inteiro $n > 0$ e para qualquer $a \in A$, verifica-se a igualdade $na = 0_A$. Então existe um menor inteiro. Então existe um menor inteiro estritamente positivo r tal que $ra = 0_A$, qualquer que seja $a \in A$. Esse inteiro r é chamado *característica* do anel A. Se, ao contrário, o anel A possui pelo menos um elemento a tal que $na \neq 0$, qualquer que seja o inteiro estritamente positivo n, então se diz que a característica do anel é zero.

Apêndice B

Códigos dos exemplos

```
p<-13
q<-29
toc<-(p-1)*(q-1)
n=p*q
rel_primos(toc)
e<-199
d<-inv_mod1(e,toc)

msg<-"f"
msgasc<-CharToAsc(msg)

cifra<-exp_mod4(msgasc,e,n)
decifra<-exp_mod4(cifra,d,n)
```

Listing B.1: Código fonte em R da aplicação da técnica RSA para parâmetros arbitrários

```
parametros<-gera_parametros_RSA2(16)
chave_publica<-parametros[c(3,5)]
chave_privada<-parametros[6]
msg<-"f"
msgasc<-CharToAsc(msg)

cifra<-exp_mod4(msgasc,chave_publica[2],chave_publica[1])
decifra<-exp_mod4(cifra,chave_privada,chave_publica[1])

msgasc == decifra
```

Listing B.2: Código fonte em R da aplicação da técnica RSA para parâmetros gerados aleatoriamente

```
primo<-751
a<- -1
```

```
b<- 188
G <- c(0,376)
chprivA <- 386
chpublB <- c(201,5)
M <- c(562,201)

kG <- multiplo(chprivA,G[1],G[2],a,primo,qtd_ptos(pt(primo,a,b)))
kPB <- multiplo(chprivA,chpublB[1],chpublB[2],a,primo,qtd_ptos(pt(primo,a,b
    )))
SPmkPB <- pt_soma(M[1],M[2],kPB[1],kPB[2],a,primo)
C <- c(kG, SPmkPB)
```

Listing B.3: Código fonte em R da aplicação da técnica ECC para parâmetros arbitrários

```
parametros<-gera_parametros_ECC1(8)

chA<-gera_chaves_ecc(parametros[1],parametros[2],parametros[3],parametros
    [4],parametros[5],parametros[6])
chB<-gera_chaves_ecc(parametros[1],parametros[2],parametros[3],parametros
    [4],parametros[5],parametros[6])
M<-gera_chaves_ecc(parametros[1],parametros[2],parametros[3],parametros[4],
    parametros[5],parametros[6])

kG <- c(chA[2],chA[3])
kPB <- multiplo(chA[1],chB[2],chB[3],parametros[1],parametros[3],parametros
    [4])
SPmkPB <- pt_soma(M[2],M[3],kPB[1],kPB[2],parametros[1],parametros[3])
C <- c(kG, SPmkPB)
```

Listing B.4: Código fonte em R da aplicação da técnica ECC para parâmetros gerados aleatoriamente

Referências

[1] Rezende, Pedro: *Preliminares em Segurança Computacional*, 2019. Instituição: Universidade de Brasília. 1, 9

[2] Oliveira, Jéssica Cristina de: *Ransomware - Laboratório de Ataque do WannaCry*, 2018. Monografia de Curso de Graduação em Engenharia de Software, UnB (Universidade de Brasília), Brasília/DF, Brasil. 1

[3] World, Computer: *Mais de 8,4 bilhões de tentativas de ataques cibernéticos atingiram o Brasil em 2020.* `https://computerworld.com.br/seguranca/mais-de-84-bilhoes-de-tentativas-de-ataques-ciberneticos-atingiram-o\-brasil-em-2020/`, 2021. Accessed: 2021-04-26. 1

[4] Brito, Sabrina: *O home office facilita a invasão de computadores por hackers.* `https://veja.abril.com.br/tecnologia/o-home-office-facilita-a-invasao-de-computadores-por-hackers/`, 2020. Accessed: 2021-05-08. 1

[5] Nacional, Jornal: *PF prende dois hackers suspeitos do maior roubo de dados do Brasil.* `https://g1.globo.com/jornal-nacional/noticia/2021/03/19/pf-prende-dois-hackers-suspeitos-do-maior-roubo-de-dados-do-brasil.ghtml`, 2021. Accessed: 2021-04-25. 1

[6] Kuhn, Markus: *Security I.* Computer Laboratory, 2016. 1, 2, 8, 13, 21, 25

[7] Chen, Yange, Hequn Liu, Baocang Wang, Baljinnyam Sonompil, Yuan Ping e Zhili Zhang: *A threshold hybrid encryption method for integrity audit without trusted center.* J. Cloud Comput., 10(1):3, 2021. `https://doi.org/10.1186/s13677-020-00222-6`. 2, 18, 42

[8] Aggarwal, Shubhani e Neeraj Kumar: *Chapter four - digital signatures.* Adv. Comput., 121:95–107, 2021. `https://doi.org/10.1016/bs.adcom.2020.08.004`. 2, 18

[9] Pal, Shantanu: *Internet of Things and Access Control - Sensing, Monitoring and Controlling Access in IoT-Enabled Healthcare Systems*, volume 37. Springer, 2021, ISBN 978-3-030-64997-5. `https://doi.org/10.1007/978-3-030-64998-2`. 2

[10] Ching, Bryan, Mani Amoozadeh, Chen-Nee Chuah, H. Michael Zhang e Dipak Ghosal: *Enabling performance and security simulation studies of intelligent traffic signal light control with VENTOS-HIL.* Veh. Commun., 24:100230, 2020. `https://doi.org/10.1016/j.vehcom.2020.100230`. 2

[11] Fallucchi, Francesca, Marco Gerardi, Michele Petito e Ernesto William De Luca: *Blockchain framework in digital government for the certification of authenticity, timestamping and data property.* Em *54th Hawaii International Conference on System Sciences, HICSS 2021, Kauai, Hawaii, USA, January 5, 2021*, páginas 1–10. ScholarSpace, 2021. http://hdl.handle.net/10125/70895. 2

[12] Stallings, William: *Criptografia e Segurança de Redes.* São Paulo: Pearson Prentice Hall, 2008. 2, 8, 9, 10, 11, 12, 13, 14, 16, 17, 20, 21, 25, 30, 40

[13] Rezende, Pedro: *Criptografia e Segurança na Informática*, 2019. Institution: Universidade de Brasília. 2, 10, 11, 13, 14, 25

[14] Katz, Jonathan e Yehuda Lindell: *Introduction to Modern Cryptography.* CRC Press, Taylor and Francis Group, FL, USA, 2015. 2, 3, 8, 10, 11, 14, 16, 23, 24, 25, 31

[15] Borges, Fábio e Pedro C. S. Lara: *Curvas elípticas: Aplicação em criptografia assimétrica.* Em *Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, 2007. 2, 13, 15, 17, 32, 36

[16] Lara, Pedro C. S.: *Implementação de uma API para Criptografia Assimétrica Baseada em Curvas Elípticas*, 2008. Trabalho de Conclusão de Curso (Tecnólogo da Informação e Comunicação), ISTCC-P (Instituto Superior de Tecnologia em Ciências da Computação de Petrópolis), Petrópolis/RJ, Brasil. 3, 8, 17

[17] Yin, Robert: *Estudo de Caso: Planejamento e Métodos.* Porto Alegre: Bookman, 2a edição, 2001. 4, 5, 6

[18] Ford, Warwick: *Secure electronic commerce: building the infrastructure for digital signatures and encryption.* New Jersey: Prentice Hall, 2ª edição, 2001. 10

[19] Feistel, H.: *Criptography and computer privacy.* Scientific American, may 1973. 12

[20] Shannon, C.: *Communication theory of secret systems.* Bell Systems Technical Journal, 1(4), 1949. 12

[21] Diffie, W. e M. E. Hellman: *New directions in cryptography.* IEEE Trans. Information Theory, 22(6):644–654, 1976. 13

[22] Majumder, Suman, Sangram Ray, Dipanwita Sadhukhan, Muhammad Khurram Khan e Mou Dasgupta: *ECC-CoAP: Elliptic Curve Cryptography Based Constraint Application Protocol for Internet of Things.* Wirel. Pers. Commun., 116(3):1867–1896, 2021. https://doi.org/10.1007/s11277-020-07769-2. 17

[23] Miller, Victor S.: *Use of elliptic curves in cryptography.* Em *Conference on the theory and application of cryptographic techniques*, páginas 417–426. Berlin: Springer, 1985. 17

[24] Singh, Deepti, Bijendra Kumar, Samayveer Singh e Satish Chand: *A Secure IoT-Based Mutual Authentication for Healthcare Applications in Wireless Sensor Networks Using ECC.* Int. J. Heal. Inf. Syst. Informatics, 16(2):21–48, 2021. https://doi.org/10.4018/IJHISI.20210401.oa2. 18

[25] Li, X., J. Niu, M. Z. A. Bhuiyan, F. Wu, M. Karuppiah e S. Kumari: *A robust ECC-based provable secure authentication protocol with privacy preserving for industrial internet of things*. IEEE Transactions on Industrial Informatics, 14(8):3599–3609, 2018. doi:10.1109/TII.2017.2773666. 18

[26] Abbas, Alaa M., Ayman Alharbi e Saleh Ibrahim: *A novel parallelizable chaotic image encryption scheme based on elliptic curves*. IEEE Access, 9:54978–54991, 2021. https://doi.org/10.1109/ACCESS.2021.3068931. 18

[27] Chambers, John: *R*. https://www.r-project.org/, 2021. Accessed: 2021-08-23. 19

[28] Koblitz, Neal: *A Course in Number Theory and Cryptography*. New York: Springer-Verlag, 2ª edição, 1994. 30

[29] Domingues, H.: *Álgebra Moderna*. São Paulo: Atual, 4a edição, 2003. 31, 40

[30] Cormen, T. H., C. E. Leiserson, R. L. Rivest e C. Stein: *Introduction to Algorithms*. The MIT Press, 3ª edição, 2009. 32

[31] Brasil: *Instituto Nacional de Tecnologia da Informação – ITI*. https://www.gov.br/iti/pt-br/acesso-a-informacao/institucional/o-iti, 2021. Accessed: 2021-10-10. 33

[32] Sangalli, Leandro e Marco Henriques: *Criptossistemas baseados em Curvas Elípticas e seus Desafios*, 2012. Institution: Universidade Estadual de Campinas. 34

[33] Shokranian, Salahoddin: *Uma Introdução À Teoria dos Números*. Rio de Janeiro: Ciência Moderna, 1ª edição, 2008. 34, 38