UNIVERSIDADE PAULISTA

DANIEL CLAROS PRADA – N583185

GUILHERME AUGUSTO – F235289

MATHEUS MOREIRA DIAS – F2265B8

VINICIUS ALVES SERAFIM – N6238E2

VITOR DE SOUZA PORTELLA – N6231D3

CONCEITOS E APLICAÇÕES DA CRIPTOGRAFIA:

A proteção de dados através de algoritmos

SÃO PAULO - SP

2020

DANIEL CLAROS PRADA – N583185
GUILHERME AUGUSTO – F235289
MATHEUS MOREIRA DIAS – F2265B8
VINICIUS ALVES SERAFIM – N6238E2
VITOR DE SOUZA PORTELLA – N6231D3

CONCEITOS E APLICAÇÕES DA CRIPTOGRAFIA:

A proteção de dados através de algoritmos

Trabalho de Atividades Práticas Supervisionadas (APS) do 2º Semestre do curso de Ciência da Computação da Universidade Paulista para obtenção total de pontos na nota da disciplina da APS

SÃO PAULO - SP
2020

RESUMO

A informação sempre foi um recurso essencial para a sobrevivência e desenvolvimento humano. Através da passagem de conhecimento, de geração em geração a humanidade avança; Através da comunicação, é feita a diplomacia, o debate, as negociações, o ensino e as mais variadas trocas de dados. Com o passar do tempo, esses entendimentos foram cada vez mais valorizados, surgindo a necessidade de privar seu acesso de estranhos ou inimigos. Então, foram criadas técnicas de proteção da informação que não dependiam de força bruta, mas de inteligência, como no caso da esteganografia, que tinha como objetivo esconder a mensagem de terceiros, e a criptografia, que tinha como objetivo ocultar o significado da mensagem.

Atualmente, com a crescente digitalização do ambiente urbano, a criptografia se torna cada vez mais conectada ao cotidiano do homem, participando desde transações bancárias até mensagens trocadas casualmente entre pessoas. Portanto, entender sua importância e capacidade é crucial para saber os limites da sua privacidade e da proteção de dados no mundo contemporâneo.

ABSTRACT

Information has always been an essential resource to the human survival and development. Through spread of knowledge, the humanity evolves from generation to generation. Through communication are made diplomacies, debates, negotiations, teaching and other kinds of data sharing. Over time, those understandings got more and more valuable, arising the necessity to deprive its access from strangers or enemies. Then, technics for information protection that didn't depend on brute force, but on intelligence, were created, which is the case of the steganography, that aims to hide the message from others, and the cryptography, that focus on the occlusion of the meaning behind the message.

Nowadays, with the ascending digitalization of the urban environment, cryptography becomes increasingly more connected to the human life, participating from banking transactions to casual message exchange between people. Therefore, understand its importance and capacity is crucial to know the limits of your privacy and data protection on the contemporary world.

SUMÁRIO

1 OBJETIVO

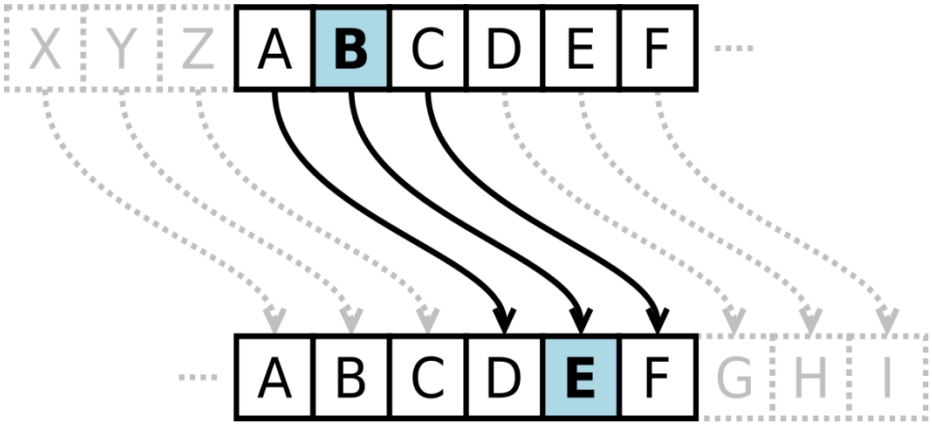
O objetivo desse documento é explanar os mínimos detalhes relativos à criptografia, desde sua origem e evolução ao longo dos anos, detalhando cada ponto de destaque em sua linha do tempo, até chegar ao estado atual, cobrindo seus principais aspectos no mundo contemporâneo, suas diversas ramificações e suas principais aplicações no mercado e na vida do cidadão comum. Além disso, apresentar aspectos referentes a técnica de criptografia RSA, contando com sua definição, algoritmo, características únicas, comparação com outras técnicas, entre outras particularidades.

2 INTRODUÇÃO

O termo criptografia não é nada mais do que a prática de codificar e decodificar algum tipo de dados.

A criptografia existe há milhares de anos, onde teve a sua origem com os hebreus a 600 a.C por meio da escrita. Com o passar do tempo, surgiu a necessidade de esconder ou dificultar que qualquer pessoa possa ler a mensagem. Isso era feito por meio de cifras de substituição monoalfabéticas (onde um símbolo do alfabeto é substituído por outro símbolo no alfabeto cifrado).

Figura 1 – Cifra de César



Fonte: Google imagens

Com o passar dos anos o estudo da criptografia foi-se aperfeiçando até que durante a primeira guerra mundial, onde Alexander´s Weekly desenvolveu métodos de criptografia que até então ajudariam os criptoanalistas britânicos na quebra dos códigos e cifras alemães.

A partir dai, a criptografia pode-se dividir em duas partes, criptografia clássica e moderna, onde a clássica foi usada por povos antigos até chegar a segunda guerra mundial já a moderna começa a ser aplicada na computação até hoje em dia.

A criptografia moderna nos ajuda em diversas coisas, onde nos podemos proteger os nossos dados sem que aconteçam interceptações por meio externos, assim deixando as nossas senhas, contas bancárias, informações pessoais mais seguras.

Não só os nossos dados podem ser protegidos por esse “Cadeado” mas como empresas que querem mais seguranças para si, aplicativos de mensagens onde as conversas são frequentemente criptografadas para que não ocorra nenhum tipo de comprometimento nesses dados.

Com isso sabemos o quão importante foi todo esse desenvolvimento das técnicas criptográficas para que ele nos proporcione diversas vantagens e segurança ao nossos dados.

3 CRIPTOGRAFIA

Criptografia é a técnica de codificação e decodificação de dados. Todos os dados são transformados em algoritmos para codificá-los para que não tenham mais o formato original, tornando assim impossível de ser entendido. Esses dados só poderão voltar ao seu formato original com a chave de descriptografia específica para esses dados. Existem várias técnicas de criptografia e elas são muito importantes para a segurança dos dados criptografados, pois ela irá proteger a informação de todo tipo de invasão.

O contato do ser humano com a criptografia é constante, estando presente em todos os lugares. Por exemplo, quando inserimos informações confidenciais em sites (RG, CPF, Senha de banco). Para ter a máxima segurança e não vazar informações, a criptografia se faz crucial, caso contrário, seria fácil para hackers mal intencionados conseguirem o que querem.

Imagem 1 – Criptografia

Fonte: Google imagens

3.1 História da criptografia

A criptografia surgiu há milhares de anos com os hebreus (600 A.C), que criaram a cifra de substituição monoalfabética, onde um símbolo é trocado por outra letra do alfabeto. Um exemplo desse tipo de cifra é a cifra Atbash, que consiste na substituição da primeira letra do alfabeto pela última, da segunda pela penúltima e assim por diante.

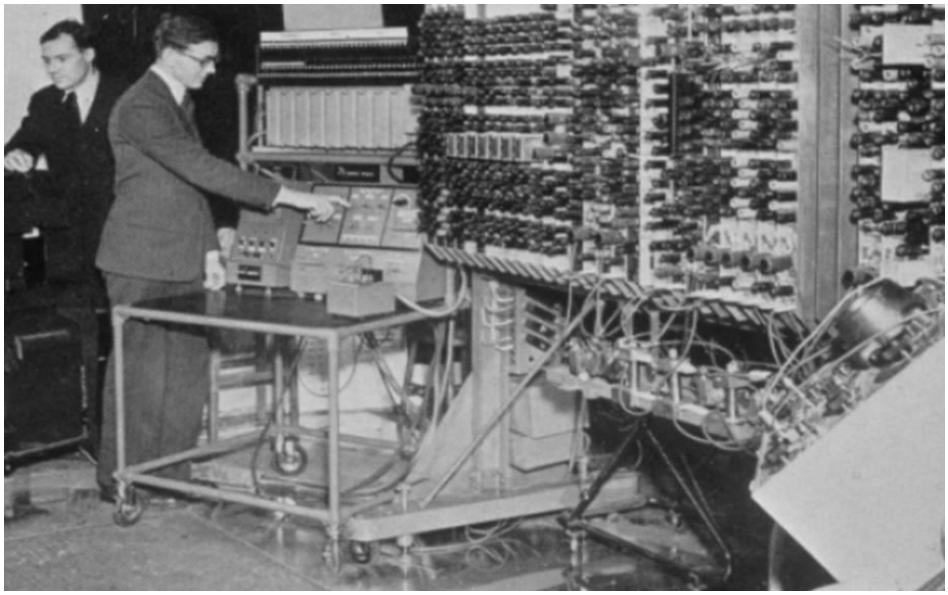
Por volta de 800 D.C, quando estavam iniciando os estudos da criptoanálise, um matemático chamado Ibrahim Al-Kadi, conseguiu quebrar esse tipo de cifra com uma técnica de autoria própria.

Até o começo da primeira guerra mundial, a criptografia não havia se desenvolvido tanto, até que o jornal Alexander's Weekly Messenger escreveu alguns métodos de criptografias, no que foi muito útil servindo como base para os criptoanalistas britânicos na quebra de códigos dos alemães durante a guerra.

Além de ser um ótimo instrumento para quebra de códigos, a sua maior contribuição foi para decodificar o telegrama Zimmermann, enviado para o embaixador da Alemanha no México, Heinrich Von Eckardt, ordenando que se aproximasse do governo mexicano para fazer uma aliança contra os Estados Unidos, em troca ele concederia terras norte-americanas caso o governo aceitasse o acordo. O telegrama foi interceptado pelos britânicos e enviado para os Estados Unidos, que logo se apressaram e entraram de vez na guerra.

Durante a Segunda Guerra Mundial, os alemães criaram uma máquina para criptografar e descriptografar as mensagens, denominada enigma. Logo após a guerra estourar, um grupo de criptógrafos britânicos, matemáticos e mestres em xadrez, entre eles se destacam nomes como Newman e Alan Turing, conseguiram através de muito esforço quebrar as cifras da Enigma, e conseguindo decifrar todas as mensagens dos alemães, salvando, assim, milhares de vidas e colocando eles na dianteira contra os inimigos.

Imagem 2 – Criptoanalistas britânicos na Segunda Guerra Mundial



Fonte: Google imagens

Durante a Guerra Fria entre os Estados Unidos e a União Soviética, foram criadas diversas técnicas de criptografia para esconder mensagens e informações confidenciais, dentre elas a criptografia Simétrica e a Criptografia Assimétrica.

Atualmente a criptografia é muito usada na internet, principalmente para a proteção de informações confidenciais, como transações bancárias, em mensagens de e-mails, entre outros. A criptografia quântica também vem ganhando espaço na área de criptografia, com crescente surgimento de novos estudos desse tipo de criptografia. Ela se destaca por não correr um risco muito alto de ser interceptada por terceiros ao necessitar de uma comunicação prévia para o envio de chaves. Fundamentalmente, essa técnica não se baseia em funções, mas sim nas leis da física.

3.2 Criptografia simétrica

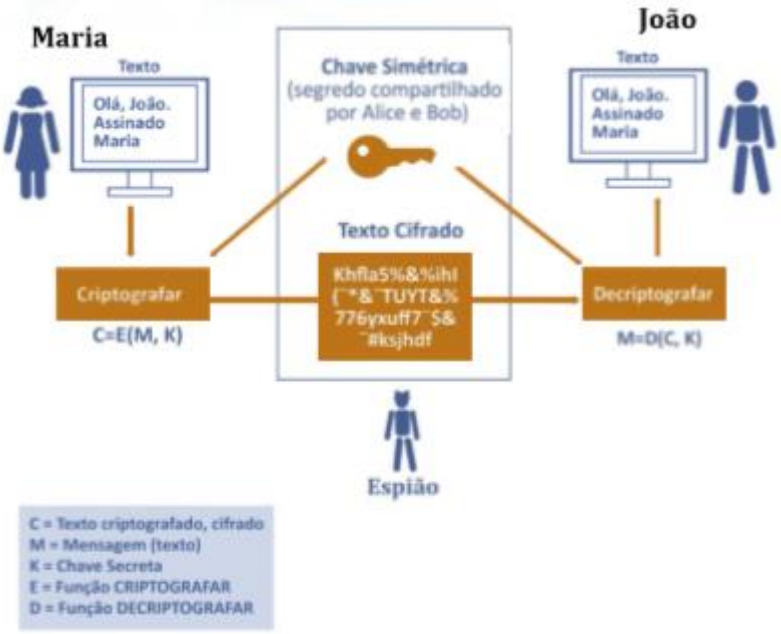
Na criptografia simétrica, o ciframento da informação é baseado em 2 componentes, um algoritmo e uma chave. Apenas uma chave é usada e passada para o remetente e o destinatário. A chave é sua própria sequência de bits e define como o algoritmo criptografa as informações.

A criptografia tem bom desempenho e pode manter a comunicação contínua entre várias pessoas ao mesmo tempo, caso aconteça alguma coisa com a chave basta substitui-la por uma nova chave e manter o algoritmo original.

A segurança varia de acordo com o tamanho da chave que o sistema usa. Por exemplo, um algoritmo baseado no DES tem 56 bits, e pode criar 72 quadrilhões de chaves diferentes, pode até parecer muito mas com a capacidade de processamento que temos hoje em dia esse sistema é considerado inseguro. Mas tem também sistemas mais confiáveis como o RC2, que usa o protocolo S/MIME, onde possui uma chave que pode variar entre 8 e 1024 bits, assim dificultando muito mais quem tenta usar força bruta para descriptografar a informação (Força bruta é uma forma que os invasores utilizam, onde são utilizadas inúmeras combinações de caracteres na tentativa de uma delas ser a chave.)

Por outro lado, este modelo de criptografia tem algumas desvantagens que estão relacionadas à geração e compartilhamento de chaves. No primeiro caso, uma chave muito simples pode ser facilmente quebrada usando um algoritmo de força bruta. Na segunda situação, deve-se prestar atenção à maneira como as chaves são compartilhadas com as informações interessadas, para evitar que sejam adquiridas pelo invasor.

Figura 2 – Funcionamento da criptografia simétrica



Fonte: Google imagens

3.3 Criptografia assimétrica

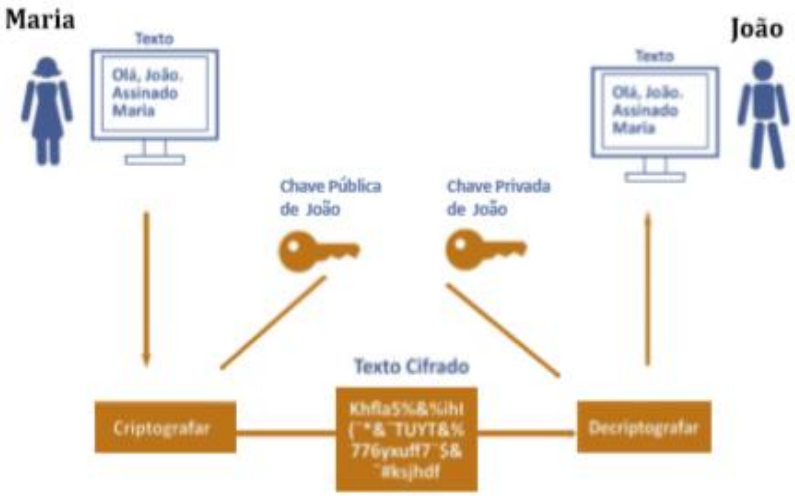
Na criptografia assimétrica usamos dois tipos de chaves de segurança, uma chave pública e a outra privada, para criptografar as informações e para decifrá-las.

Basicamente, a chave privada serve para descriptografar a informação e a chave pública para criptografar. Assim então, qualquer um pode passar uma informação criptografada, ele precisa apenas da chave pública de seu destinatário, que usa a chave privada para decifrar a informação.

Uma das principais técnicas usadas na criptografia assimétrica é a RSA. Ele é baseado na multiplicação de números primos em grande escala para gerar uma chave pública. O tempo preciso para quebrar uma chave RSA pode ser consideravelmente grande.

Em 1999 o instituto nacional de pesquisa da Holanda fez um trabalho com cientistas de 6 países diferentes e usando 300 computadores, em 7 meses eles conseguiram quebrar uma chave RSA de 512 bits.

Figura 3 – Exemplificação da criptografia assimétrica



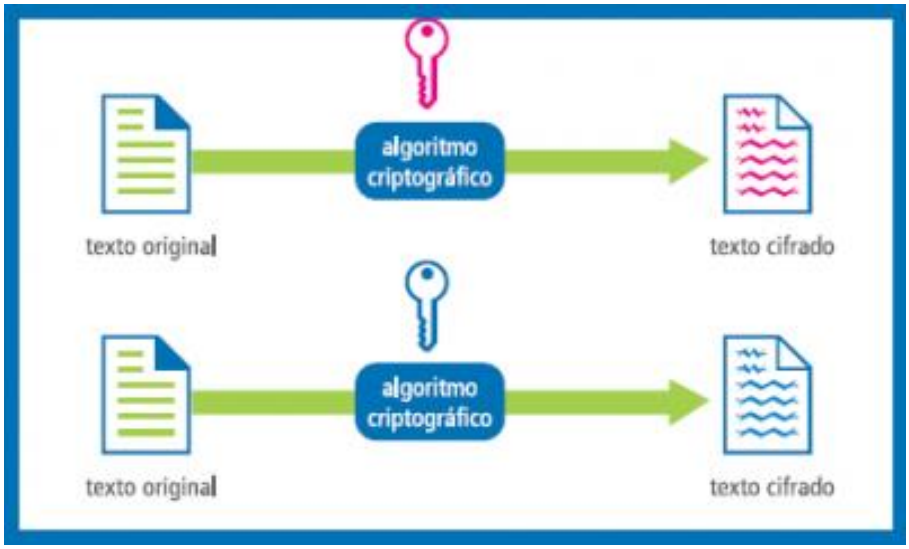
Fonte: Google imagens

4 TÉCNICAS CRIPTOGRÁFICAS MAIS UTILIZADAS

Como nós vimos anteriormente, podemos observar que a medida do avanço tecnológico, as diferentes maneiras de criptografia foram se aprimoraram e consequentemente começaram a aparecer outros tipos e técnicas de criptografias que seriam usadas até o mundo moderno, sendo assim abordaremos dois tipos de chaves que são usadas para fazer o processo de criptografia de uma mensagem: chaves simétricas e assimétricas.

4.1 Chaves Simétricas

Figura 4 – Funcionamento da criptografia simétrica



Fonte: https://www.diegomacedo.com.br/chaves-simetricas-assimetricas/

É um criptossistema que utiliza uma única chave para criptografar e descriptografar dados. Por isso, essa técnica é mais eficaz para criptografia de algoritmos simples, pois na criptografia assimétrica são realizadas diversas operações e cálculos, o que pode atrapalhar no desempenho e na velocidade. Com isso, entende-se que por se tratar de algoritmos simples, a criptografia simétrica será muito mais rápida que a assimétrica e geralmente será a mais adequada para a criptografia de grande quantidade de dados. Segue alguns exemplos:

4.1.1 DES (Data Encryption Standard)

Foi desenvolvido pela IBM em 1977.É um dos primeiros tipos de criptografia, ou seja, básico. A chave é composta por uma cifra de blocos de 64 bits (8 caracteres). O DES faz combinações, substituições e permutações entre o texto e a chave. Por ser uma estrutura simples de 64 bits o método foi sendo quebrado por tentativa e erro, sendo assim, um método não muito seguro para criptografia de dados.

4.1.2 RC (Ron’s Code ou Rivest Cipher)

Foi criado na RSA Data Security, por Ron Rivest, ela utiliza chaves de 0 a 1024 bits e por se tratar de um método seguro é usado geralmente em e-mails.

4.1.3 Blowfish

Foi desenvolvido por Bruce Schneier em 1993, o diferencial dela é que não é patenteada e está à disposição de qualquer usuário também é uma cifra de blocos com o uso de chaves de 32 a 448 bits. Desde então foram feitos vários estudos analisando o sistema de criptografia Blowfish, porém Serge Vaurdenay estudou pontos fracos nesse método e concluiu que por mais que possam ser detectadas não poderão ser quebradas.

4.1.4 IDEA (International Data Encryption Algorithm)

Desenvolvido por Xueija Lai e James Massey, foi criado em 1991 no Swiss Federal Institute of technology. Por ser uma cifra de bloco e usar uma chave para encriptar e decriptar considera-se também uma chave simétrica, ela é uma chave de 128 bits que encripta dados em blocos de 64 bits, por se tratar de uma chave de 128 bits ela pode ser considerada mais segura que o DES, pois demoraria muito tempo e tentativas para conseguir decriptar um dado sem a chave.

4.2 Chaves assimétricas:

Diferente das chaves simétricas as assimétricas usam duas chaves diferentes para realizar o processo, uma ira encriptar e outra decriptar, que também podem ser chamadas de chave pública e chave privada.

A chave pública é usada para cifrar o texto, algoritmo ou dado, onde a mesma pode ser enviada pela internet sem problema nenhum e o dado não ficara vulnerável pois será usada uma outra chave para decifrar esse dado que é chamada de chave privada, onde está chave precisará ficar secreta.

Por se tratar de uma técnica criptográfica mais complexa, pode-se concluir que ela é mais demorada nos casos em que o tamanho do dado for maior, mas ao mesmo tempo essa chave é uma das formas de criptografia mais seguras e usadas no tempo moderno. Exemplos:

Figura 5 – Funcionamento da criptografia assimétrica

Fonte: https://medium.com

4.2.1 RSA (Rivest-Shamir-Adleman)

Pode ser considerado um dos primeiros algoritmos de uso de chave assimétrica e um dos mais usados nos dias de hoje, foi criado por Ronald Rivest e Adi shamir ambos eram cientistas da computação no MIT com a ajuda do matemático Leonard Adleman, assim patenteando o método RSA no ano de 1977.

Esse método é construído a partir da teoria dos números onde para gerar chaves públicas e privadas se faz o uso da multiplicação de dois números primos onde o resultado dará a chave pública.

4.2.2 El Gamal

Em 1985 o investigador egípcio Taher Elgamal mais conhecido como o pai de SSL, propõe um algoritmo de cifra e de assinatura digital chamando-o assim de El Gamal, por mais que o algoritmo é seguro o algoritmo RSA continua sendo o mais usados atualmente.

É um algoritmo que usa o problema de logaritmo de algoritmo discreto. A geração de suas chaves segue o padrão das chaves públicas, ou seja, cada entidade gera um par de chaves e acertam a forma de como vão distribuir as chaves públicas.

Uma desvantagem desse método de criptografia é o texto a ser criptografado é duplicado e gerando assim, tempos de computação mais longos para garantir a segurança do canal de comunicação.

Imagem 3 – Retrato do investigador Taher Elgamal

Fonte: https://en.wikipedia.org

4.2.3 DSA (Digital Signature Algorithm):

Foi criado em 1991 por um funcionário da NSA chamado Dr. David W. Kravitz. É baseada na assinatura do El Gamal, mas não duplica o texto, e caracterizado por ser mais compacto e econômico. Não é nada mais do que um algoritmo que usa assinatura digital faz também o uso de algoritmos de hash assim como é um algoritmo de chave pública.

Um dos pontos negativos desse método é que não pode ser usada para criptografia de dados já que é usada apenas para gerar assinaturas digitais.

5 SOBRE O RSA

A origem da técnica criptográfica RSA surgiu a partir de um conceito proposto por Diffie e Hellman para criação de um método de criptografia com chave pública. Apesar das diversas técnicas desenvolvidas por especialistas ao redor do mundo, o RSA foi o primeiro a atingir êxito, ao ser criado em 1978, e se mantém até hoje como uma das técnicas criptográficas mais utilizadas.

Seu nome é composto pela primeira letra do sobrenome dos criadores: Rivest, Shamir e Adleman. Além de terem criado o RSA, eles trabalharam na criação de outras técnicas menos populares, como no caso dos algoritmos RC.

5.1 Conceitos e estruturação do RSA

O algoritmo por trás dessa técnica envolve fórmulas com expressões exponenciais, teorias de números primos e outros conceitos matemáticos. Entre eles estão:

- **Função totiente de Euler:** É definida como a quantidade de números menores que n e relativamente primos a n (números que não são divisíveis por n nem múltiplos de outro número divisível por n), sendo n um número qualquer.

  Ex. $\phi(14) = 6$, pois os números relativamente primos são: 1, 3, 5, 9, 11, 13

  Caso n seja um número primo, a função pode ser calculada pela fórmula:

  $$\phi(n) = n - 1$$

- **Teorema de Fermat:** Esse teorema diz que se um número p é primo e n é um número inteiro positivo não divisível por p, então:

  $$N^{p-1} \equiv 1 (\text{mod } p)$$

  Esse é considerado um teorema importante para o estudo da Teoria dos Números, que envolve grande parte do conhecimento matemático utilizado na produção do algoritmo RSA.

- **Teorema de Euler:** O Teorema de Euler propõe que sendo a e n relativamente primos:

$$a^{\phi(n)} = 1 (\text{mod } n)$$

Assim como o Teorema de Fermat, o Teorema de Euler serve como base para inúmeras fórmulas matemáticas que são utilizadas ou criadas até hoje.

5.1.1 O algoritmo RSA

O algoritmo RSA começa com alguns requisitos para sua viabilidade:

1. Existem valores **e**, **d** e **n** tais que M^{ed} mod n = M para todo M < n
2. Ser relativamente fácil calcular M^{e} mod n e C^{d} mod n para todo M < n
3. Ser inviável o cálculo de **d** sabendo-se **e** e **n**.

Após a viabilidade ser atestada, é necessária a criação de alguns valores específicos para os futuros cálculos de encriptação e decriptação:

1. dois números primos **p** e **q**
2. um valor público **n**, sendo **n** = **p** * **q**
3. **e**, sendo ϕ(**n**) e **"e"** números primos entre si e 1 < **e** < ϕ(**n**)
4. **d** ≡ $\mathbf{e}^{-1}$ (mod ϕ(n))

Após isso, são criadas as chaves do algoritmo: uma chave pública contendo {e, n} e uma chave privada contendo {d, n}. Para o funcionamento do algoritmo, é necessário primeiramente que o receptor da mensagem publique sua chave pública de forma que o emissor consiga obtê-la, sem necessidade de esconder de terceiros. Após isso, o emissor encripta sua mensagem de texto claro em blocos cifrado, e cada bloco M gerado será transformado num bloco de mensagem criptografado C, seguindo a seguinte fórmula de encriptação:

$$C = M^{e} \text{ mod } n$$

A decriptação da mensagem C para texto claro novamente será realizada pelo receptor através da fórmula:

$$M = C^d \bmod n$$

5.2 Benefícios em relação às técnicas anteriores

Quando pensamos em chaves assimétricas uma das primeiras coisas que vem na nossa cabeça é o método de encriptação RSA, além de ser o mais popular e usado, é também um dos primeiros algoritmos assimétricos a ser criado. Com isso, a maioria de chaves assimétricas criadas e usadas hoje em dia se baseou no método RSA e por isso foi-se aprimorando com o passar dos anos.

Dessa forma, nós iremos comparar o método RSA, que ainda é muito usado, com as técnicas baseadas em chaves simétricas usando a tabela a seguir:

	Criptografia simétrica	**Criptografia assimétrica**
Funcionamento	O mesmo algoritmo é usado para criptografar e decriptografar a mensagem.	O mesmo algoritmo é usado para criptografar e decriptografar a mensagem, porém usando duas chaves.
Requer	Que destino e origem saibam o algoritmo e a chave.	A origem e o destino devem saber uma (somente uma) chave do par de chaves. Todos podem ter a chave pública, porém só 1 deve saber a chave privada.
Segurança	- A chave deve ser mantida em segredo - Mesmo sabendo o algoritmo e tendo exemplos dos textos criptografados deve impossibilitar a determinação da chave.	- Apenas 1 das duas chaves deve ser mantida em segredo. - É impossível decifrar uma mensagem mesmo tendo acesso ao algoritmo, à chave pública e a exemplos dos textos cifrados. - Tendo a chave pública deve ser impossível chegar na chave privada.
Utilidade	- Privacidade	- Identificação -Assinatura Digital -Privacidade -Troca de Chaves - Entre outras.
Velocidade de Processamento	- Muito Rápida	- Lenta

5.3 Aplicações do RSA

Esse algoritmo de criptografia já foi utilizado em virtualmente todo tipo de sistema que precisa de encriptação, sendo extremamente popular até hoje. Ele é utilizado em sistemas de criptografia de transações bancárias, sistemas operacionais da Microsoft e Apple, placas de rede Ethernet, e todos os principais protocolos de comunicação segura na Internet, como S/MIME, S/WAN e SSL, além de ser vastamente utilizado por órgãos governamentais e universidades. Ademais, o RSA também é bastante utilizado na transferência de senhas RC4 geradas a partir de sua chave pública, graças à sua velocidade comparada a outras técnicas.

Outras aplicações comuns do RSA são nos sistemas de assinatura digital, o que fez com que o RSA se destacasse entre outras técnicas, pois ela garante que é uma cópia legitima do documento original, assim assegurando também a autoria.

Para conseguirmos usar a assinatura digital é preciso utilizar a função Message Digest (MD – Resumo da mensagem ou mensagem direta). Com esse recurso será possível processar o arquivo, gerando assim um pequeno pedaço de dados, geralmente chamado de hash. Para uma melhor compreensão, Message Digest é uma função matemática que pode refinar as informações do documento em um único pedaço de dados de tamanho fixo.

5.4 Comparação do RSA com outras técnicas

Quando são comparadas duas técnicas de criptografia diferentes, é possível discutir sob duas perspectivas diferentes. Uma de comparação de técnicas, quando a outra técnica também utiliza uma chave pública e uma privada, e outra de comparação de sistemas em geral, quando se trata de um sistema de chave única.

O RSA, sendo um dos primeiros sistemas de criptografia de chave pública inventados, possui vulnerabilidades das quais foram muito estudadas, e conforme os anos se passaram muitos outros sistemas similares foram desenvolvidos para aperfeiçoar tanto sua velocidade, quanto sua segurança, como por exemplo o uso de criptografias de curvas elípticas, que em comparação com o RSA demanda um número muito menor de bits para ambas as chaves sem comprometer em segurança graças ao problema de logaritmo discreto, e também foram implementados protocolos de segurança que utilizam diversas

etapas para manter a segurança dos dados, como o uso de múltiplos métodos de criptografias sucessivos, tanto para o hash (o processo de “embaralhar” a informação) de arquivos quanto para obter uma assinatura digital, que é utilizada não só para saber a procedência do arquivo, mas também para alguns processos criptográficos.

Dito isso, aqui está uma análise comparativa específica entre técnicas com outros métodos de criptografia e quais são seus usos em determinados contextos:

OTR (Off The Record): que utiliza um algoritmo simétrico com chave de pelo menos 128 bits, a função hash SHA-1 (resume ou aumenta qualquer mensagem para outra de 20 bytes, mas criptografada), e o algoritmo Difiie-Hellman, ele é melhor utilizado para serviços de chat, já que garante a criptografia e uma autorização negável.

SSH (Secure Shell): é um sistema feito para garantir a segurança de informações mandadas para uma network, comumente usado para a segurança de logins, ele utiliza da autenticação criada pela chave pública que ele usa em sua criptografia para garantir a confiabilidade das informações de login.

DSA (Digital Signature Algorithm): Assim como o RSA o DSA é um algoritmo de criptografia assimétrica, sendo uma variante do ElGamal, e a dificuldade em sua decriptografia está nos conceitos matemáticos da exponenciação modular e o problema do logaritmo discreto, já foi muito utilizado em protocolos de segurança, mas foi substituído por outros como o blowfish e o RSA.

RC: utiliza-se de uma chave com valor entre 40 e 2048, gerada pseudorandomicamente (randômica dentro de um certo padrão pré-definido) para efetuar uma permuta na mensagem, mas graças a sua difícil implementação, e problemas de segurança expostos em 2013 seu uso foi descontinuado.

DES: funciona por meio de uma série de permutas dentro de uma string, e tem uma saída com outra string de mesmo tamanho porem, por ser uma estrutura simples de 64 bits o método foi sendo quebrado, mas acredita-se que o método 3DES (uma repetição do método DES, onde se encripta a mensagem, decripta-se e encripta-se suscetivelmente) porem devido ao tempo que esse método leva foi substituído pelo AES.

Blowfish: utiliza-se de um sistema que encripta dados em blocos de 8 bytes, e utiliza uma chave de valor entre 32 e 448 bits e sub-chaves previamente computadas, uma dessas chave de até 448 bits é dividida em outras que juntas somam até 4168 bytes, então na parte do hash do algoritmo ocorre uma alternância dentro dessas sub-subchaves, o Blowfish é muito usado dentro de sistemas Linux, em serviços de e-commerce, em password wallets, etc.

IDEA: é um algoritmo que utiliza uma chave de 128 bits que encripta dados em blocos de 64 bits, é muito utilizado dentro de protocolos de e-mail.

ElGamal: é um método muito parecido com o RSA, mas nas suas operações ele acaba duplicando a mensagem o que o faz bem mais lento que outros algoritmos.

RSA: é muito utilizado para criptografar chaves de outros algoritmos simétricos, isso faz com que ele garanta a segurança das informações, ao mesmo tempo que não seja necessário usar o RSA em toda a mensagem, o que tomaria muito tempo.

5.5 Vulnerabilidades do RSA

Por ser um algoritmo vastamente utilizado, é crucial que essa técnica seja explorada e pesquisada para a manutenção da segurança dos sistemas que a utilizam. O professor de criptografia aplicada e segurança de computadores, Dan Boneh, diz que:

> O sistema de criptografia RSA é comumente usado para providenciar privacidade e assegurar autenticidade dos dados digitais. [...] Em resumo, o RSA é utilizado em áreas onde a segurança dos dados digitais é vital.” (1999, p. 1, tradução nossa)

Partindo dessa observação e da presença do RSA nos inúmeros setores diferentes de tecnologia, entender as vulnerabilidades desse algoritmo se torna crucial, pois ao fazê-lo também se aborda diretamente as vulnerabilidades da segurança de dados de vários sistemas importantes ao redor do mundo.

Entre as vulnerabilidades do RSA, temos:

Fatoração de números inteiros muito grandes: o primeiro método para atacar o sistema. Ele teria como objetivo a chave pública através da fatoração do módulo, dada uma certa fatoração ele pode chegar a decriptografia. Isso é basicamente um ataque de força bruta. Os algoritmos de fatoração de números inteiros vêm melhorando muito durante os anos, essa ainda é uma ameaça muito distante da realidade, caso o sistema RSA for bem executado será muito difícil e trabalhoso decriptografar usando esse método com a tecnologia que temos atualmente.

Pequeno Expoente da chave privada: Nessa situação para decriptografar uma mensagem tenta usar o valor de pequeno no lugar de um aleatório, na intenção de reduzir o tempo de decriptografia ou no tempo para gerar uma assinatura. Assim eles podem alcançar uma melhora de desempenho em torno de 10 fatores para um módulo de 1024 bits, porém Michael Wiener descobriu que isso pode causar na quebra do sistema.

Pequeno expoente da chave pública: A utilização de expoentes públicos pequenos é normalmente usada, o menor expoente é três, porém para o sistema ficar mais seguro é sugerido que use a geração aleatória para se proteger se certos ataques. Ataques desse tipo não levam a uma quebra total do sistema, sendo assim então menos perigosos, diferente do método de pequeno expoente da chave privada.

Para evitar a geração de módulos diferentes poderia ser considerado o uso do mesmo módulo para todos os usuários emitido, por exemplo, por uma autoridade central confiável. Apesar de parecer eficiente em uma primeira análise, um usuário poderia usar seus próprios expoentes para fatorar o módulo de outros usuários. Devido a este fato, um módulo RSA nunca deve ser utilizado por mais de uma entidade. Este é considerado um ataque elementar pois ilustra o uso errôneo do sistema RSA.

Modulo Comum: para evitar a geração de módulos diferentes, é usado o mesmo módulo para emitir para todos os usuários. Apesar de aparentemente ser eficiente, um usuário poderia usar seus próprios expoentes para fatorar módulos de outros usuários. Devido a isso, um módulo RSA nunca deve ser usado por mais de um usuário. Este tipo de ataque é considerado um ataque elementar, pois mostra o uso errado do sistema RSA

5.6 Melhorias propostas ou implementadas do RSA

Por ser um algoritmo de criptografia considerado lento, muitas bibliotecas criptográficas fazem alguns ajustes na hora de aplicas o RSA no seu sistema. Um desses ajustes é utilizar o algoritmo chinês de resto. Ao utilizar dessa técnica, o custo de recurso para processar os cálculos do algoritmo diminuem, pois em vez de calcular uma exponenciação de alto custo, são feitas 2 exponenciações modulares no lugar.

Para compensar sua baixa velocidade de processamento, diversas companhias fazem uso do RSA misturado com um sistema de criptografia simétrico, conseguindo utilizar a forte proteção do RSA e a velocidade e flexibilidade do sistema simétrico de criptografia.

6 ESTRUTURA DO PROGRAMA

O programa criado para o trabalho de Atividades Práticas Supervisionadas (APS) foi escrito na linguagem de programação Python, que é mais simples comparado à outras linguagens populares no mercado e possui todos os recursos necessários para o desenvolvimento do programa.

Por ser um programa baseado no algoritmo do RSA, ele pode ser separado em 3 partes, começando com a simulação do receptor da mensagem gerando as chaves necessárias, seguido da criptografia e envio da mensagem pelo emissor utilizando a chave pública gerada, e finalizando com a decriptação com a chave privada por parte do receptor.

6.1 Parte 1: Criação das Chaves

Para começar o algoritmo RSA é necessário gerar 2 números primos quaisquer. Quanto maiores os números primos, maior a segurança da encriptação, porém o impacto no desempenho do sistema é realizado exponencialmente. No programa em questão, os números 29 e 31 foram selecionados.

Após isso, são gerados os seguintes valores:

$$n = p * q \quad (1)$$

$$\phi(n) = (p - 1) * (q - 1) \quad (2)$$

Eles são necessários para realizar a fórmula do algoritmo RSA, sendo n diretamente utilizado na chave pública e privada, enquanto ϕ(n) é utilizado na criação do resto das chaves.

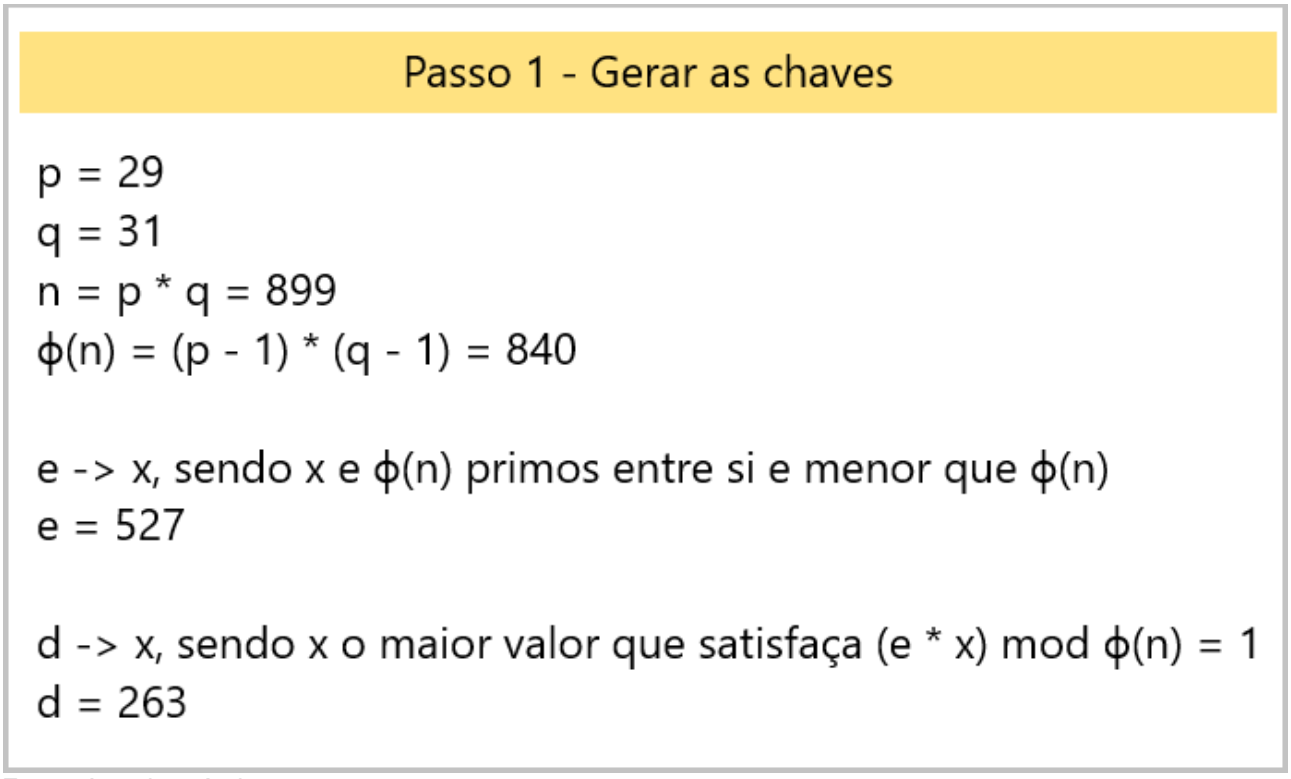
Gerado os valores de p, q, n e ϕ(n), inicia-se o processo de criação das chaves. O valor de **e** (nomeado assim por se tratar do valor utilizado para encriptação) será calculado para obter-se o conjunto da chave pública. O valor de **e** será qualquer valor que quando comparado a ϕ(n) seja um número primo entre si, podendo resultar em uma lista cheia de opções para **e**. No caso do programa, isso é realizado através de loopings do tipo “for”, que avaliarão a disponibilidade de todos os valores dentro do alcance de 1 à ϕ(n) através de cálculos dentro de estruturas de condição “if”.

Escolhido o valor de **e**, obtêm-se a chave pública que será utilizada no sistema – a chave pública é representada por **{e, n}**, ambos já calculados.

Após obtido o valor de **e**, será gerado o valor de **d** (nomeado dessa forma para representar o valor utilizado na chave de decriptação). O valor de **d**, assim como o valor de **e**, também poderia ser escolhido entre uma lista de opções, mas por questões de praticidade foi decidido que sempre o maior valor de **d** possível será utilizado. Esse processo é feito através de um único looping que vai de 1 à ϕ(n), é checado toda vez que um número satisfaz a fórmula **(e * x) mod ϕ(n) == 1**, aplicado o valor de x à d. Graças ao looping que verificará do menor ao maior número do limite imposto, toda vez que um valor x satisfazer a estrutura de condição "if" ele será atribuído à **d**, sempre prevalecendo o maior valor utilizável.

Com o valor de **d** já determinado, chega-se ao conjunto que representa a chave privada, que será utilizada para decriptar a mensagem para o receptor – a chave privada é representada por **{d, n}**, ambos já calculados.

Ilustração 1 – Visualização da estrutura do programa – Passo 1



Fonte: Autoria própria

6.2 Parte 2: Criptografia da mensagem

Pelo algoritmo RSA realizar a criptografia através de cálculos matemáticos, a mensagem do emissor deverá ser convertida de alguma forma para um tipo numérico. Nesse programa em questão, a mensagem é tratada em 3 passos no processo da criptografia.

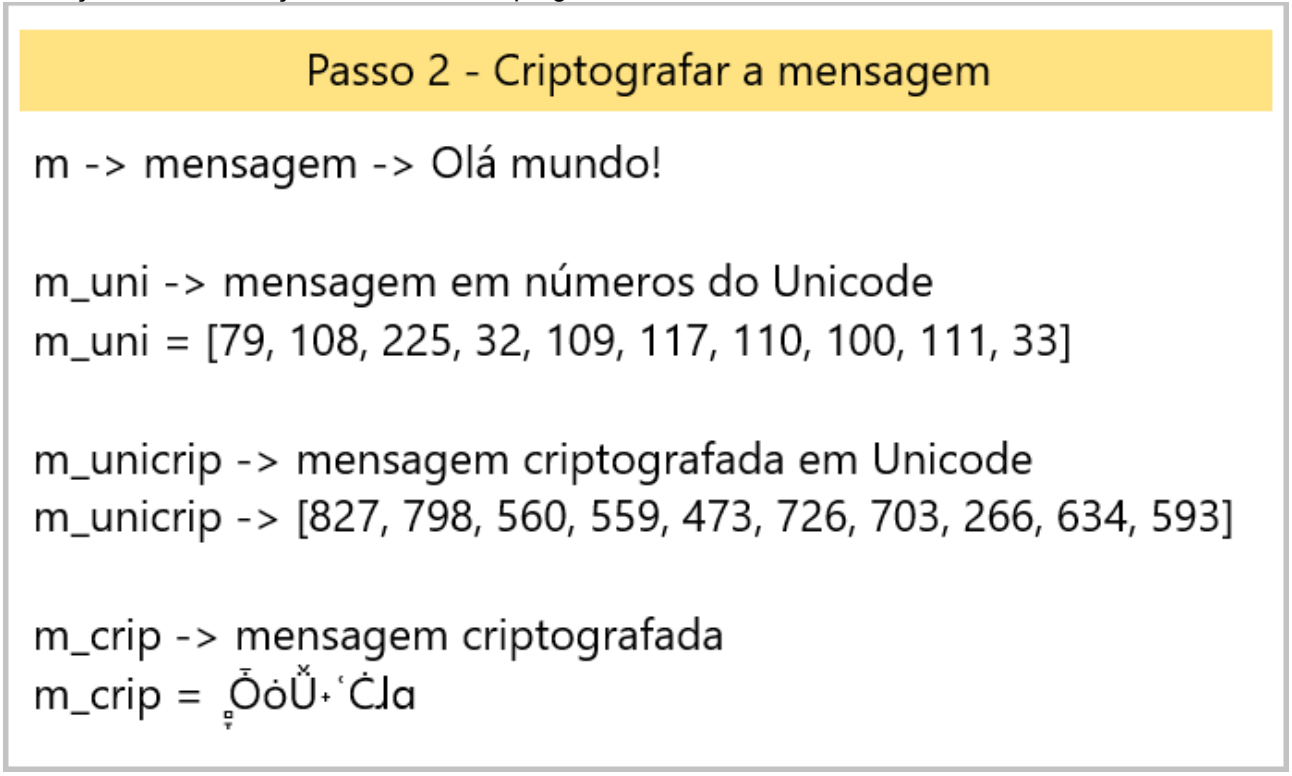
No primeiro passo, é realizada uma pseudo-separação da mensagem em caracteres únicos através de um looping do tipo “for”, que cuidará de cada caractere da mensagem individualmente.

No segundo passo, é obtido o valor numérico inteiro daquele caractere representado no código Unicode, caso seja um objeto Unicode, ou o valor do byte quando se trata de uma string de 8-bit. O recurso do python responsável por esse processo é o método **ord(string** text**)**.

No terceiro passo, o valor do caractere é aplicado à fórmula de criptografia do RSA: $\mathbf{num^e\ mod\ n}$, sendo **num** o valor Unicode ou em byte do caractere, **e** o valor principal da chave pública de encriptação, **mod** a operação “módulo”, que retorna o resto da divisão entre 2 números, e **n** o produto dos dois números primos gerados no começo do processo.

Logo após cada encriptação de caractere, ele é transformado de número para caractere novamente e armazenado numa variável a frente dos tratados anteriormente a ele, tendo como resultado a criação da mensagem criptografada.

Ilustração 2 – Visualização da estrutura do programa – Passo 2



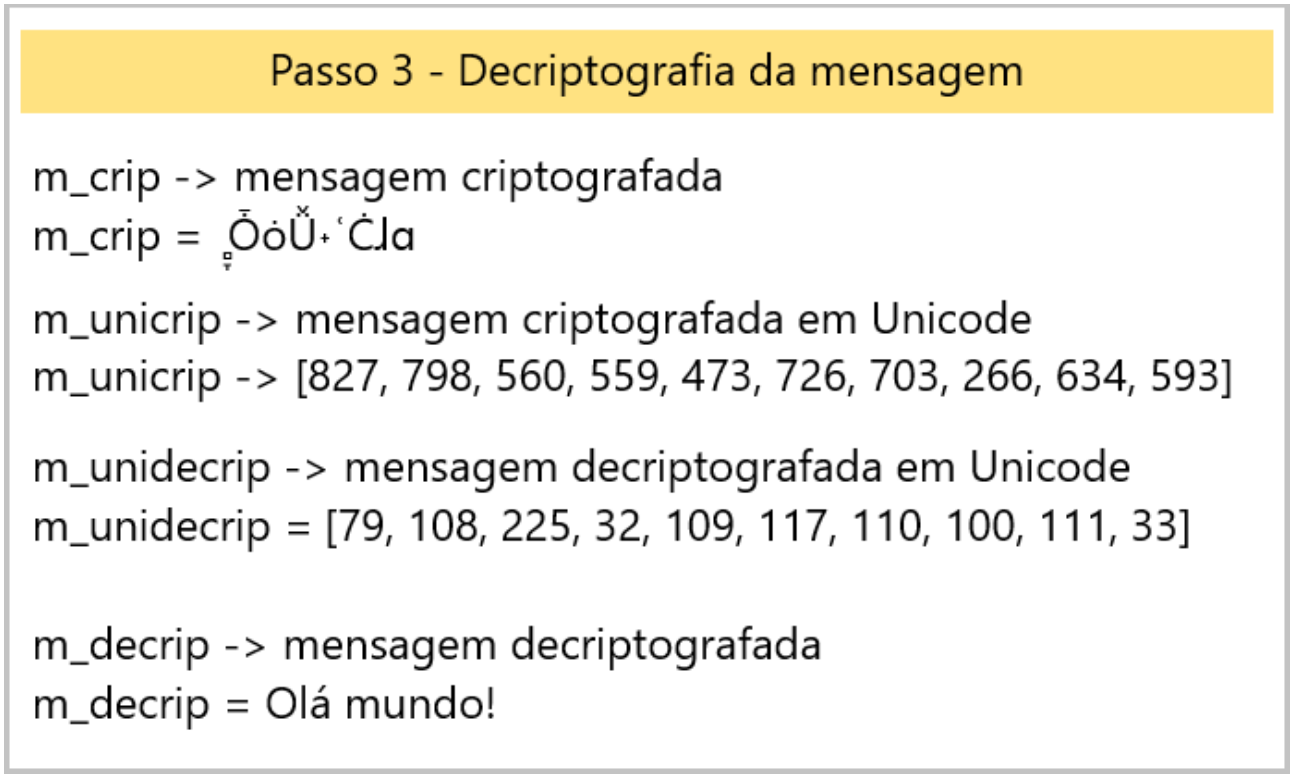
Fonte: Autoria própria

6.3 Parte 3: Decriptografia da mensagem

A decriptografia da mensagem ocorre de forma semelhante ao processo de encriptação, porém de forma inversa, pois no primeiro passo cada caractere é tratado individualmente, no segundo acontece a transformação do caractere em seu valor numérico inteiro no Unicode ou em seu valor em bytes, no terceiro é processada a decriptação através da fórmula de decriptação do RSA: $\mathbf{num^d\ mod\ n}$, sendo **num** o valor inteiro do caractere criptografado, **d** o valor principal da chave privada de decriptografia, **mod** a operação “módulo” que retorna o resto da divisão entre 2 números, e **n** o produto dos 2 números primos gerados no início do algoritmo RSA.

Após isso, o caractere é convertido novamente para o símbolo original da mensagem e armazenado numa variável junto dos demais símbolos, originando a mensagem original do emissor sem nenhuma perda de informação.

Ilustração 3 – Visualização da estrutura do programa – Passo 3



Fonte: Autoria própria

7 RELATÓRIO COM AS LINHAS DE CÓDIGO DO PROGRAMA

Assim como na descrição da estrutura, as linhas de código podem ser divididas em 3 etapas: criação das chaves, criptografia da mensagem e decriptografia da mensagem.

7.1 Criação das Chaves

Primeiro são feitos os preparativos para a criação das chaves, sendo gerados os valores das variáveis p, q, n e phi_n.

Imagem 4 – Parte 1 do programa - Preparativos

```
print("\n --- Parte 1: Receptor - Criação das chaves --- \n")

# Preparando a criação das chaves:
p = 29
q = 31
n = p * q
phi_n = (p - 1) * (q - 1)

print("Valores primos utilizados:\np =", p, "\nq =", q)
```

Fonte: Autoria própria

Após isso, os possíveis valores de **e** são decididos através de um “for” loop.

Imagem 5 – Parte 1 do programa – Achando o valor de **e**

```
# Achando o valor de e:
e = 0

invalid_list = []
valid_list = []

for i in range(2, phi_n - 1):
    if phi_n % i == 0:
        invalid_list.append(i)
    else:
        for j in invalid_list:
            if i % j == 0:
                invalid_list.append(i)
                break
    if i not in invalid_list:
        valid_list.append(i)

print("\nValores possíveis de e:", valid_list, sep="\n\n")

inp_e = int(input("\nEscolha um valor da lista para e: "))

if inp_e in valid_list:
    e = inp_e
else:
    print("Valor inválido.")
```

Fonte: Autoria própria

Essa parte do programa gerará uma lista com todos os possíveis valores de **e**, de acordo com os devidos requisitos do algoritmo de criptografia RSA. Ao terminar esse processo, o programa exibirá os possíveis valores e requisitará do usuário que escolha um dos valores da lista para prosseguir com a execução do programa, caso seja digitado um valor fora da lista, o programa exibirá na tela “Valor inválido.”

Após gerado o valor de **e**, o programa dá início ao cálculo da variável **d**. Por questões de praticidade e simplificação, dos possíveis valores de **d** apenas o maior número disponível será utilizado.

Imagem 6 – Parte 1 do programa – Achando o valor de **d**

```
# Achando o valor de d:
d = 0

for x in range(1, phi_n):
    if (e * x) % phi_n == 1:
        d = x

print("d =", d)
print('{:-^40}'.format("-"))
print("Chave pública (encriptar): {e =", e, ", n =", n, "}")
print("Chave privada (decriptar): {d =", d, ", n =", n, "}")
print('{:-^40}'.format("-"))
```

Fonte: Autoria própria

Decididos os valores de **e** e **d**, o programa exibe as chaves pública e privada para o usuário, com suas respectivas descrições e valores.

7.2 Criptografia da mensagem

Com o valor da chave pública **{e, n}**, o programa dá início a criptografia da mensagem. Primeiro, o programa pede para o “emissor” que entre com a mensagem desejada, e assim que recebida o programa começa o processo de conversão da mensagem em caracteres Unicode, ou no valor dos bytes do caractere.

Imagem 7 – Parte 2 do programa – Recebendo e convertendo a mensagem

```
print("\n --- Parte 2: Emissor - Criptografar a mensagem com a chave pública --- \n")

# Receber a mensagem e as chaves
m = input("Escreva a mensagem a ser codificada: ")
e = int(input("Digite o valor número de e da chave para criptografar: "))
n = int(input("Digite o valor número de n da chave para criptografar: "))

# Transformar a mensagem clara em texto cifrado
mb = []

for i in range(0, len(m)):
    mb.append(ord(m[i]))
```

Fonte: Autoria própria

Tendo convertido a mensagem, o programa dá início ao processo de criptografia da mensagem.

Imagem 8 – Parte 2 do programa – Criptografia da mensagem

```
# Criptografar a mensagem
c = []

print("\nMensagem criptografada: ", end="")

for i in range(0, len(mb)):
    num = mb[i]
    c.append(chr(num**e % n))
    print(c[i], end="")
```

Fonte: Autoria própria

Para exibição clara do processo ao usuário, uma sutil combinação de print com parâmetros “end” modificados e o loop “for” é utilizada, de forma que todos os caracteres são exibidos um seguido do outro, como num texto comum em vez de uma lista.

7.3 Decriptografia da mensagem

Com o valor da chave privada **{d, n}** e a mensagem criptografada já processados, é iniciada a decriptografia da mensagem. Para realizá-la, é seguido uma sequência de comandos praticamente inversa a da encriptação.

Imagem 9 – Parte 3 do programa – Decriptografia da mensagem

```
print("\n\n --- Parte 3: Receptor - Decriptação da mensagem --- \n")
# Decriptografar a mensagem
md = []

d = int(input("Digite o valor número de d da chave para decriptografar: "))
n = int(input("Digite o valor número de n da chave para decriptografar: "))

print("\nMensagem decriptografada: ", end="")
for i in range(0, len(mb)):
    num = c[i]
    md += (chr(ord(num)**d % n))
    print(md[i], end="")
```

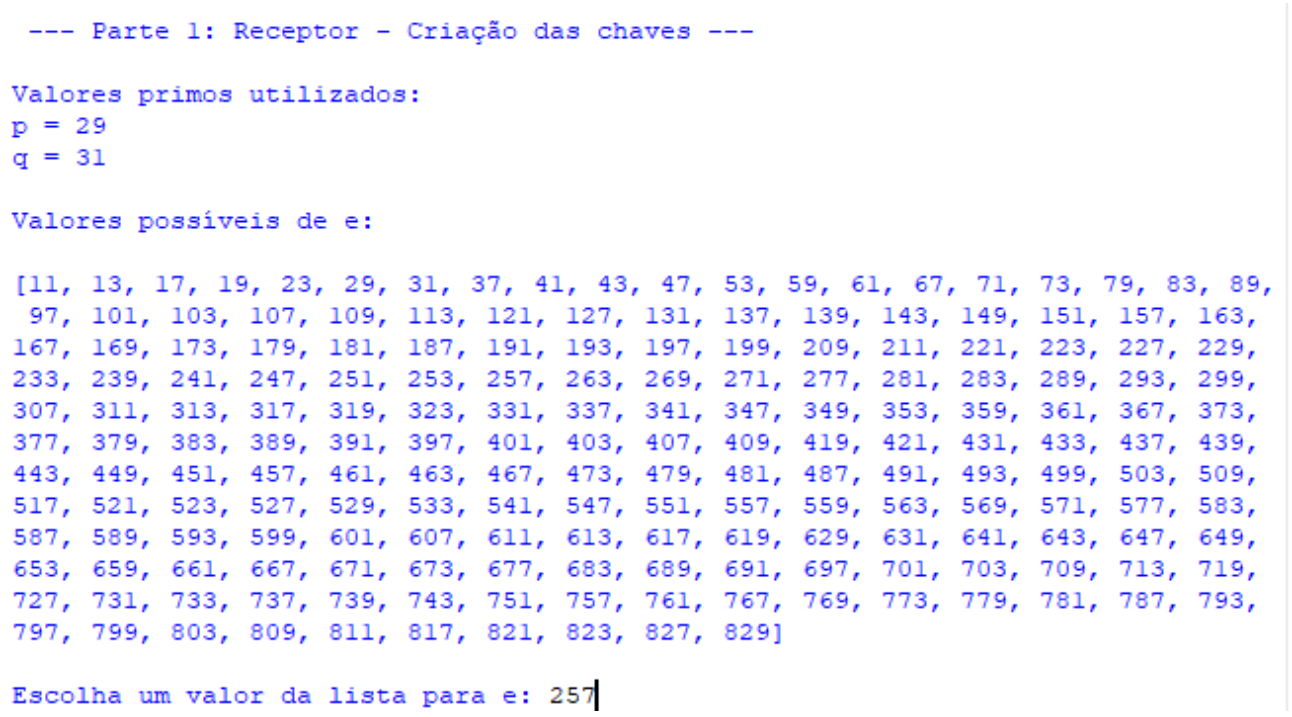
Fonte: Autoria própria

8 APRESENTAÇÃO DO PROGRAMA EM FUNCIONAMENTO

A execução do programa foi realizada num computador desktop com o sistema operacional Windows 10, através do IDLE (Integrated Development and Learning Environment).

A tela inicial do programa já mostra partes do passo 1 executados, para o conforto do usuário. O primeiro contato direto do usuário com o programa é através da entrada do valor **e** desejado, desde que esteja dentro da lista de opções exibida.

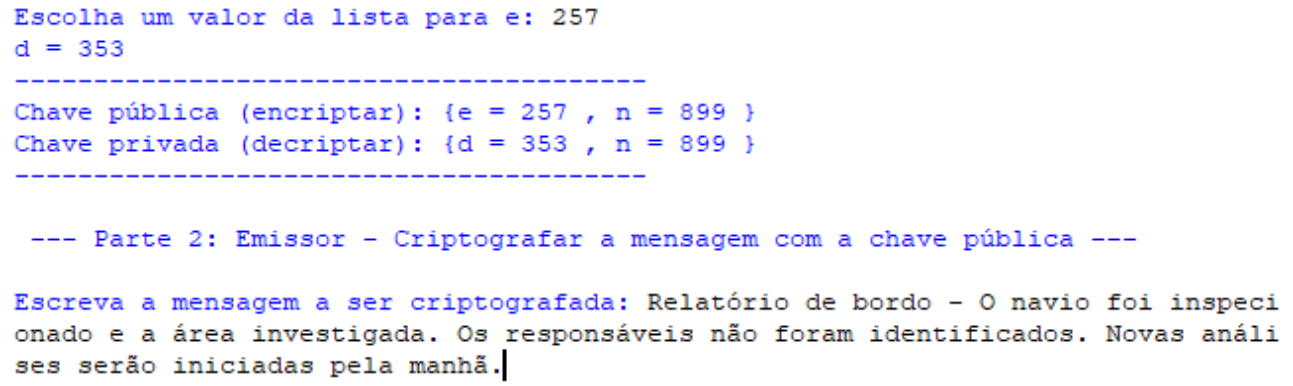
Imagem 10 – Programa em execução – Entrada do valor **e**



Fonte: Autoria própria

Após expressa a escolha, o programa procede com a criação e exibição das chaves pública e privada do algoritmo RSA, seguido da solicitação de entrada da mensagem a ser encriptada.

Imagem 11 – Programa em execução – Entrada da mensagem

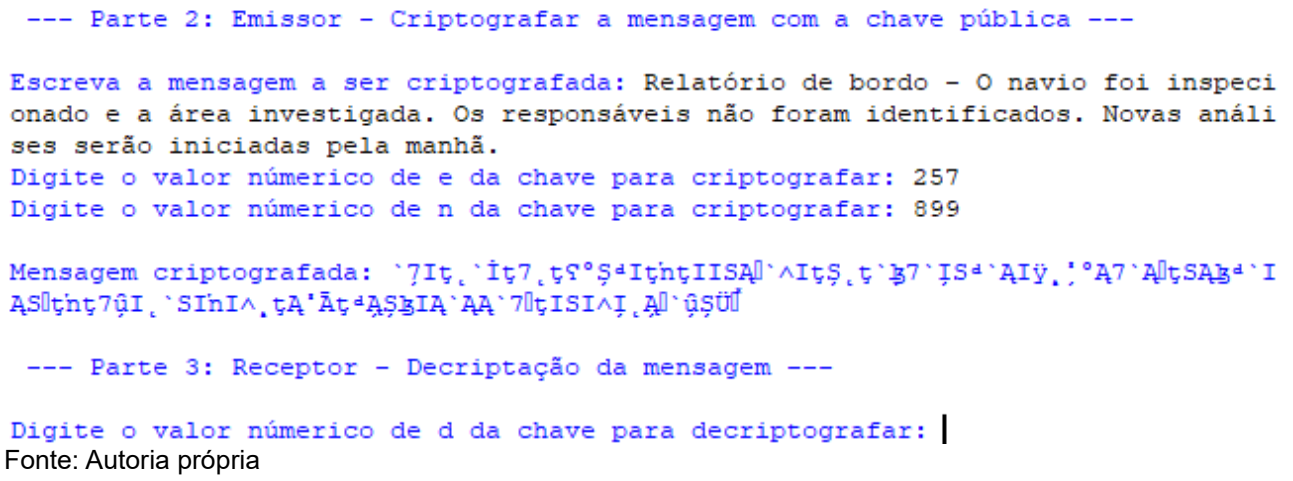


Fonte: Autoria própria

Dado o contexto do trabalho da APS, uma mensagem secreta contando a situação das investigações foi gerada usando a criptografia RSA.

Após passada a mensagem, o programa pede a entrada dos valores de **e** e **n** da chave pública, seguida da mensagem criptografada utilizando esses 2 valores passados.

Imagem 12 – Programa em execução – Mensagem criptografada



Fonte: Autoria própria

Após exibição da mensagem criptografada, inicia-se a decriptação da mensagem. Primeiro o programa pede os valores **d** e **n** da chave privada, e dados os valores, é então exibida na tela a mensagem decriptografada, idêntica à original.

Imagem 13 – Programa em execução – Mensagem decriptografada

```
 --- Parte 2: Emissor - Criptografar a mensagem com a chave pública ---

Escreva a mensagem a ser criptografada: Relatório de bordo - O navio foi inspeci
onado e a área investigada. Os responsáveis não foram identificados. Novas análi
ses serão iniciadas pela manhã.
Digite o valor númerico de e da chave para criptografar: 257
Digite o valor númerico de n da chave para criptografar: 899

Mensagem criptografada: `ȚIţ˛`İţ7˛ţʕ°Ş˞IţńţIISĄ`^IţŞ˛ţ`Ʒ7`ĮS˞`ĄIÿ˛¦°Ą7`ĄţSĄƷ˞`I
ĄSţńţ7ûI˛`SIńI^˛ţĄ'Āţ˞ĄŞƷIĄ`ĄĄ`7ţISI^Į˛Ą`ûŞÜ

 --- Parte 3: Receptor - Decriptação da mensagem ---

Digite o valor númerico de d da chave para decriptografar: 353
Digite o valor númerico de n da chave para decriptografar: 899

Mensagem decriptografada: Relatório de bordo - O navio foi inspecionado e a área
 investigada. Os responsáveis não foram identificados. Novas análises serão inic
iadas pela manhã.
>>>
```

Fonte: Autoria própria

9. BIBLIOGRAFIA

STALLINGS, Willians. Criptografia E Segurança De Redes: PRINCÍPIOS E PRÁTICAS. Editora Pearson, 2008

BONEH, Dan. Twenty Years of Attacks on the RSA Cryptosystem. Notices of the AMS, 1999

https://cryptoid.com.br/banco-de-noticias/29196criptografia-simetrica-e-assimetrica/

https://web.cs.wpi.edu/~guttman/pubs/jcs_strand_spaces.pdf

https://www.venafi.com/blog/how-diffie-hellman-key-exchange-different-rsa

https://www.usenix.org/system/files/conference/usenixsecurity13/sec13-paper_alfardan.pdf

https://cyberleninka.org/article/n/593680.pdf

https://pt.wikipedia.org/wiki/Data_Encryption_Standard#:~:text=O%20Data%20Encryption%20Standard%20(DES,utilizado%20em%20larga%20escala%20internacionalmente.

https://br.ccm.net/contents/132-introducao-a-codificacao-des

http://penta2.ufrgs.br/gere96/segur2/des.htm

https://www.oficinadanet.com.br/post/9424-quais-os-principais-tipos-de-criptografia

https://blog.validcertificadora.com.br/tipos-de-criptografia-conheca-os-10-mais-usados-e-como-funciona-cada-um/

https://pt.wikipedia.org/wiki/Algoritmo_de_chave_sim%C3%A9trica

https://pt.wikipedia.org/wiki/International_Data_Encryption_Algorithm

https://www.netinbag.com/pt/internet/what-is-idea-encryption.html

https://pt.wikipedia.org/wiki/RSA_(sistema_criptogr%C3%A1fico)

https://trustcentral.com/about/about-dr-david-w-kravitz/

http://www.dsc.ufcg.edu.br/~pet/jornal/abril2014/materias/historia_da_computacao.html#:~:text=A%20história%20da%20criptografia%20começa,letra%20do%20alfabeto%20pela%20última

https://pt.wikipedia.org/wiki/RSA_(sistema_criptográfico)

https://www.kaspersky.com.br/resource-center/definitions/encryption

10. FICHAS DE ATIVIDADES PRÁTICAS SUPERVISIONADAS

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Vitor de Souza Portella | TURMA: CC2A33 | RA: N6231D3

CURSO: Ciência da Computação | CAMPUS: Tatuapé | SEMESTRE: 2º | TURNO: Diurno

CÓDIGO DA ATIVIDADE: 76B8 | SEMESTRE: 2º | ANO GRADE: 2020/02

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
14/11/2020	Confecção do projeto destinado a Atividades Práticas Supervisionadas	75h	Vitor de Souza Portella	75h	

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS:_____

AVALIAÇÃO:_____

Aprovado ou Reprovado

NOTA:_____

DATA:_____/_____/_____

_____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Daniel Claros Prada **TURMA:** CC2A33 **RA:** N583185

CURSO: Ciencia da computação **CAMPUS:** Tatuapé **SEMESTRE:** 2ª semestre **TURNO:** Manhã

CÓDIGO DA ATIVIDADE: 76B8 **SEMESTRE:** 2ª semestre **ANO GRADE:** 2020

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
14/nov	Confecção do projeto detinado a Atividade Prática Supervesionada	75 Horas	Daniel Claros Prada	75 Horas	

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 75 Horas

AVALIAÇÃO: _____
Aprovado ou Reprovado

NOTA: _____

DATA: _____/_____/_____

_____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Guilherme Augusto Sbizero Correa TURMA: CC2A33 RA: F23528-9

CURSO: Ciência da Computação CAMPUS: Tatuapé SEMESTRE: 2º Semestre TURNO: Manhã

CÓDIGO DA ATIVIDADE: SEMESTRE: 2º Semestre ANO GRADE: 2020/21

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
14/11/2020	Confecção do projeto detinado a Atividade Prática Supervesionada	75h	Guilherme Auugusto Sbizero Correa	75h	

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS:_____

AVALIAÇÃO:_____
Aprovado ou Reprovado

NOTA:_____

DATA:____/_____/_____

_____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Matheus Moreira Dias **TURMA:** CC2A33 **RA:** F2265B8

CURSO: Ciencia da computação **CAMPUS:** Tatuapé **SEMESTRE:** 2ª semestre **TURNO:** Manhã

CÓDIGO DA ATIVIDADE: 76B8 **SEMESTRE:** 2ª semestre **ANO GRADE:** 2020

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
14/nov	Confecção do projeto destinado a atividades práticas supervisionadas	75h	Matheus Moreira Dias	75	

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 75 Horas

AVALIAÇÃO: _____
Aprovado ou Reprovado

NOTA: _____

DATA: _____/_____/_____

_____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Vinicius Alves Serafim TURMA: CC2A33 RA: N6238E-2

CURSO: Ciência da Computação CAMPUS: Tatuapé SEMESTRE: 2º Semestre TURNO: Manhã

CÓDIGO DA ATIVIDADE:76B8 SEMESTRE: 2º Semestre ANO GRADE: 2020/21

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
14/11/2020	Confecção do projeto detinado a Atividade Prática Supervesionada	75h	Vinicius Alves Serafim	75h	

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS:_____

AVALIAÇÃO:_____

Aprovado ou Reprovado

NOTA:_____

DATA:_____/_____/_____

_____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO