UNIVERSIDADE UNIP TATUAPÉ

CURSO DE GRADUAÇÃO CIÊNCIAS DA COMPUTAÇÃO

DAVI COUTINHO PIRES

BRUNO SOARES CARVALHO

CAIO SILVA PIRES

GUSTAVO BEHRING ARAÚJO

LUIZ FERNANDO SILVA SANTINELLI

ATIVIDADES PRÁTICAS SUPERVISIONADAS

CRIPTOGRAFIA

SÃO PAULO

2020

Sumário

INTRODUÇÃO

Quando as pessoas escutam a palavra criptografia, logo imaginam sistemas eletrônicos e computadores, senhas, segredos ou informações. Mas a criptografia já passou por diversas mudanças ao longo da história, e teve uma crescente evolução nos métodos designados para esconder informações confidenciais.

Dentre diversos tipos de técnicas criadas para alcançar o sigilo inerente de dados, esse documento faz observações acerca do algoritmo criptográfico RSA. Um sistema assimétrico que utiliza conceitos matemáticos complexos para sua utilização, fornecendo características como segurança e praticidade em troca de informações.

Este documento, portanto, busca adequar o leitor ao entendimento dos conceitos que regem esse ramo da computação, apresentar o funcionamento e as características que envolvem o RSA e criação de um modelo experimental funcional do algoritmo em python 3, levando em consideração toda a estrutura desse modelo criptográfico.

OBJETIVO

Este trabalho tem por objetivo identificar e explicar os conceitos básicos que determinam os tipos e o funcionamento acerca dos algoritmos de criptografia. Além de, estabelecer conexão entre esses conceitos e a proposta de alguns algoritmos mais conhecidos no mundo.

Escolher uma técnica dentre as mais utilizadas atualmente, expor sua estrutura, particularidades e vulnerabilidades. E acerca disso, propor hipóteses para melhoria no nível de segurança do algoritmo em um contexto real.

Projetar e construir um modelo experimental de um sistema, em linguagem de programação Python 3, que cifre e decifre uma mensagem de acordo com os critérios definidos pela técnica.

CRIPTOGRAFIA CONCEITOS GERAIS

Não é de hoje que há a necessidade de pesquisar e elaborar métodos para tornar uma mensagem ou informação secreta entre o remetente e o destinatário. No entanto, no meio termo entre a informação sair do remetente e chegar ao destinatário, pode ser interceptada. Dessa forma, ao ser interceptada de alguma maneira, esta informação é facilmente adquirida pelo interceptador e passa a ser quebrado o sigilo inerente desejado pelos dois praticantes do segredo.

Com o tempo, a humanidade percebeu que era importante estabelecer métodos e medidas para que mesmo que a informação fosse parar em mãos indesejadas, fosse o mais difícil possível de ser compreendida, para que o sigilo da informação fosse mantido. Com isso, o conceito de criptografia surge.

Basicamente criptografia é um procedimento em que um certo dado é processado por um sistema pré-definido, para gerar um dado completamente diferente na saída do processamento, no intuito de tornar esse dado, mesmo que interceptado, indecifrável por terceiros.

Ao longo do tempo, o conceito criptografia se tornou cada vez mais necessário, não só pela necessidade de guardar informações confidenciais de grande relevância em um cenário mundial, como também garantir a privacidade da sociedade como um todo, principalmente com o avanço da tecnologia e da tendência ascendente à integração do meio físico, com o meio digital. Diante desse contexto, cada vez mais tem-se estudado conceitos e métodos para garantir a confidencialidade dos dados em sistemas computacionais.

Alguns conceitos foram determinados para classificar tanto as funcionalidades de cada modelo criptográfico, como também, seus tipos. Entre eles, os conceitos seguintes tipificam os modelos e usos de todos os algoritmos criados. São eles: Secret, cipher, plaintext, ciphertext e initialization vector.

No mesmo passo que foi requisitado a cifragem de informações confidenciais, também havia muitas pessoas que tinham o desejo ou necessitavam saber essas informações para o objetivo proposto a cada contexto. Dessa forma, deu-se origem ao conceito criptanálise, que é alguma estratégia seguida por alguém que quer decifrar algum código baseado no modelo aplicado em questão.

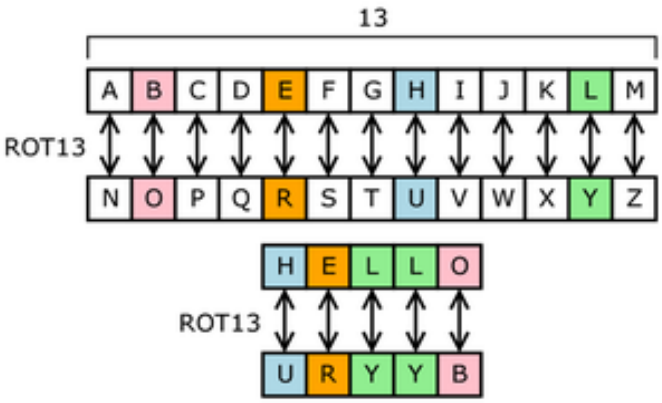
No início, como os algoritmos tinham chaves que proporcionavam possibilidades de combinações “pequenas” era possível utilizar de Brute Force Atack para descobrir o segredo. Brute Force Atack é uma técnica simples, que consiste em gerar várias chaves aleatórias, que em um determinado tempo é capaz de coincidir com a chave utilizada pelo sistema. Porém com o conhecimento adequado dos passos que um algoritmo segue, é possível reduzir o tempo que se levaria para fazer coincidir uma chave aleatória com a chave original, fazendo o processo inverso de um algoritmo ou até estabelecendo condições condizentes com o método aplicado, que faça com que o tempo de decifragem seja drasticamente diminuído

Antes de abordar e entender como funciona os processamentos dos dados de acordo com o modelo correspondente, é importantíssimo esclarecer o que é cada definição supracitada. O conceito secret nada mais é do que uma “chave” ou “segredo”, que é utilizado em diversos algoritmos de forma a concatenar com a informação já processada para cifrar ou decifrar uma informação de acordo com o tipo de algoritmo.

Já chiper se refere a esconder a informação em si ou codificá-la, esse processo pode ser denominado como cifra. Os algoritmos utilizaram diversas cifras ao longo dos anos, mas as mais conhecidas que deram origem a algoritmos mais famosos como cifra de césar, são cifra de transposição e cifra de substituição. Na cifra de transposição, a informação conserva seu comprimento, porém, altera a ordem dos seus elementos reordenando de acordo com o algoritmo aplicado, dessa forma, formando anagramas ou até distribuindo os elementos em espécies de vetores.

Como por exemplo, considerando uma simples função que realiza um anagrama com a palavra “amora”, nesse caso, pode ser formadas 60 combinações diferentes, considerando que ela tem 5 letras, porém, 1 repetida, expressando matematicamente: 5! / 2! (5*4*3*2*1 / 2*1) = 60 Combinações diferentes podem ser obtidas.

No entanto, para entender a cifra de substituição é necessário considerar como base um alfabeto, ou alguma lista de elementos seguindo uma ordem. Inicialmente este método de algoritmo seleciona um shift, que é um número fixo que determinará quantos elementos o caractere selecionado avançará ou retornará da lista pré-definida para ser substituído pelo elemento correspondente. Exemplo: considerando o alfabeto como a lista pré-determinada, ao definir o shift utilizado como 2, se a letra selecionada for “A”, a cifra transcreveria para “C” ou “y”.



Fonte:https://pt.wikipedia.org/wiki/Cifra_de_substitui%C3%A7%C3%A3o

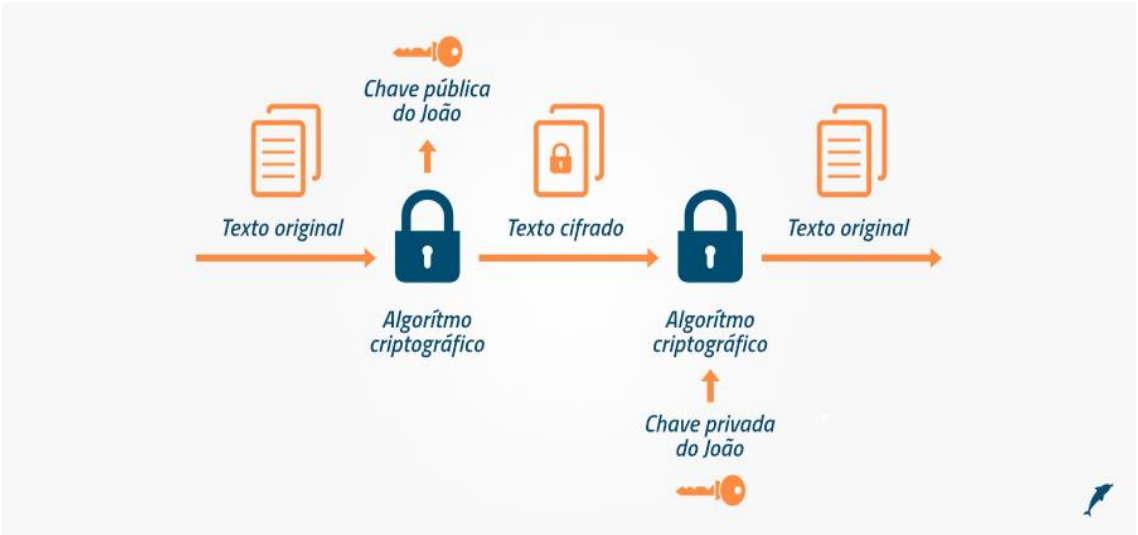
Assim como sua tradução literal do inglês para o português, plaintext se refere a um texto puro, ou seja, a informação que deseja cifrar. Em contrapartida, ciphertext é o oposto do plaintext, ou seja, o texto já processado e criptografado. Por último, o inicialization vector é geralmente uma variável pseudoaleatória utilizada nos algoritmos para tornar mais difícil decifrá-lo.

Essas definições abriram uma quantidade enorme de definições dos tipos de algoritmos de criptografia, entre elas, os tipos simétricos, mais conhecidos como chave privada. E os assimétricos, mais conhecidos como chave pública. Os algoritmos de chave privada utilizam apenas uma chave.

O emissor utiliza uma chave para encriptar a mensagem e o receptor utilizam a mesma chave para descriptografar.

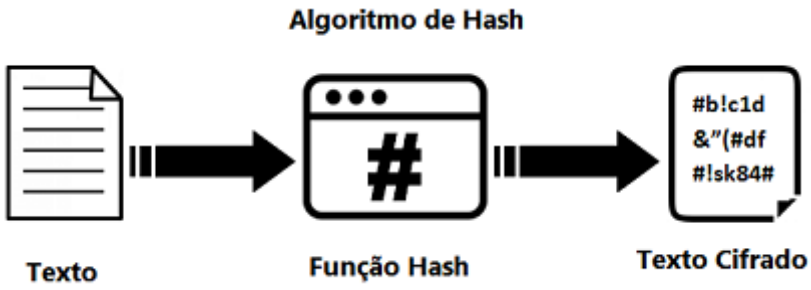
Fonte: https://www.evaltec.com.br/criptografia-de-dados-e-gerenciamento-de-chaves/

Já os algoritmos de chave pública utilizam 2 chaves distintas, para tornar a comunicação mais segura em um ambiente de comunicação inseguro. Cada usuário desta técnica, possui uma chave pública e uma privada. A chave privada só o proprietário dela conhece, enquanto a pública é compartilhada com todos. A fim de um melhor entendimento, segue um exemplo: Um emissor deseja enviar mensagem para uma pessoa de forma segura, então o emissor encripta a mensagem com uma chave pública do destinatário, de forma que a mensagem só pode ser aberta utilizando a chave privada do destinatário que só ele possui.



Fonte: https://www.evaltec.com.br/criptografia-de-dados-e-gerenciamento-de-chaves/

Existe um tipo de algoritmo de criptografia que foge um pouco dos anteriores tanto em conceito, como também, em aplicação. Esse tipo de algoritmo são as funções de hash. Diferente dos anteriores, as funções de hash não permitem que um texto seja criptografado e feito o processo inverso. Para essa função é possível apenas cifrar a mensagem, dessa forma sendo chamado de algoritmo “one-way”.



Fonte: http://www.macoratti.net/16/09/net_cripto2.htm

A função mais comum do hash é gerar uma função de processamento da informação através de diversas operações lógicas. Assim sendo, elas convertem as informações desejadas em código binário, e as separam por grupos de código. Após isso, é realizado uma série de operações lógicas com os grupos separados, e a saída resultante é novamente processada com outro bloco já processado para depois de ter realizado todas as operações desejadas de acordo com o algoritmo correspondente, concatenar todos os conjuntos de bits em um único código, formando um tipo de assinatura digital, que será abordado mais adiante nesse documento.

TÉCNICAS CRIPTOGRÁFICAS MAIS UTILIZADAS E CONHECIDAS

Basicamente todo algoritmo de criptografia ou técnica desse meio, se baseou em alguns conceitos supracitados para atingir seus objetivos. Contudo, é importante citar o funcionamento de algumas das técnicas mais conhecidas e utilizadas não só no passado, como também no presente.

Se tratando de importância, é indispensável o entendimento do modelo criptográfico alemão na segunda guerra mundial, não só pelo avanço tecnológico que representou para a época, como também pelo contexto histórico marcante. Embora os sistemas fossem arcaicos comparados aos modelos atuais, foi de fato impressionante o resultado obtido pelo exército alemão naquela época.

Utilizando o alfabeto comum e máquinas de escrever conectadas a rotores e refletores, eles desenvolveram uma cifragem de bloco eletromecânica que era capaz de gerar uma letra diferente na saída que só poderia ser decifrada com um sistema preparado que entendesse a complexidade gerada. As máquinas de escrever que eles detinham, tinham letras conectadas em blocos, chamados de plugboards, que substituíam as letras dadas como entrada, em uma letra totalmente diferente na saída. Entretanto, para tornar mais difícil de entender o padrão gerado por uma simples cifra de substituição, os plugboards faziam com que cada letra fosse substituída por uma letra qualquer dentro do bloco associada a mesma, fazendo com que nem sempre uma determinada entrada, fosse a mesma saída, dificultando o processo de criptanálise.

Mas como é de conhecimento mundial, a Alemanha perdeu a guerra e o fato dos Estados Unidos ter conseguido interceptar e decodificar suas mensagens teve grande relevância nisso. Alan Turing e uma equipe de cientistas conseguiram quebrar o código com o auxílio de uma máquina de escala gigante, que simulava a cifra de blocos proposta pelos alemães em vários sistemas menores de combinações de letras em cada bloco,

em que, ao momento que um sistema não fazia sentido, era descartado, até o momento em que um sistema fizesse sentido.

Esse foi um dos momentos mais importantes para a história do campo da criptografia e um pontapé inicial para a pesquisa e desenvolvimento de sistemas cada vez mais complexos para manter informações sigilosas.

Antes de tratar dos algoritmos mais relevantes para o contexto desse documento, é importante estabelecer as funções de hash como fundamentais em todo ambiente virtual dado suas características específicas. Por serem algoritmos “one way” eles apenas recebem as informações como entrada de dados, e fazem o processamento desses dados com uma série de operações lógicas e matemáticas complexas, para atribuir um valor na saída normalmente expresso no sistema numérico hexadecimal que seja diferente da saída de qualquer outra informação gerada por esse algoritmo baseado na informação processada.

Esse procedimento tem o intuito de criar um tipo de assinatura digital, capaz de identificar possíveis fraudes de documentos comparando os resumos dos hash’s criados para averiguar a validade da informação. Essa possibilidade atribui uma função essencial para todos os sistemas já criados, que é tornar uma senha criada por um usuário escondida e verificar a validade da mesma digitada pelo usuário ou por algum fraudador.

Diversas outras funções podem ser realizadas com as funções de hash. Principalmente nos tempos atuais com a popularidade das criptomoedas, em que, muitas obtém o valor dos códigos através de funções de hash, e entender esse processo é fundamental no momento de “minerar” essas criptomoedas.

Voltando a tratar de contexto histórico, na década de 70, o NIST (National Institute of Standards and Technology) concluiu após diversos estudos, que o EUA necessitava de uma forma de garantir a segurança das informações do governo norte americano. Naquele contexto, eram necessárias as seguintes características para o algoritmo ser escolhido como principal meio de criptografia das informações do país:

Possuir um nível de segurança elevado ligado a uma pequena chave de codificação e descodificação, ser compreensível, adaptável, econômico, eficaz e exportável. O único algoritmo que atendeu os critérios foi o DES, criado pela IBM.

O DES tinha o princípio de ser um sistema de codificação simétrico por blocos de 64 bits, onde, 8 bits poderiam servir como bits de paridade, ou seja, semelhantes entre si. Um em cada oito bits de paridade serviam como teste, dessa forma, eram ajustados para haver um número ímpar nos bytes onde eles pertenciam. Contudo, o comprimento das chaves se dava por 56 bits, que por sinal era uma característica muito criticada na época, já que os algoritmos já tinham chaves de 64 bits por padrão.

A crítica fazia sentido, uma vez que ao definir o algoritmo com uma chave menor que os algoritmos concorrentes, torna ele mais inseguro. A diferença de possibilidades de chaves é imensa, enquanto outros algoritmos da época detinham 2^64 (1,8*10^19) possibilidades de chaves, o DES apresentava apenas 2^56 (7.2*10^16) possibilidades de chaves. Mas mesmo com as incansáveis críticas, o governo do EUA adotou o DES como modelo criptográfico padrão do país.

Mesmo com todo esse nível de complexidade, o DES já era considerado defasado desde seu surgimento, gerando chaves consideravelmente pequenas e por consequência foi quebrado pouco tempo após sua criação.

Após muito tempo, no ano de 2002, finalmente o DES foi substituído pelo AES após uma competição pública. Também se tratando de um algoritmo de chave simétrica, onde todas as chaves são privadas e não devem ser compartilhadas com membros indesejados, o AES chegou com uma proposta diferente em termos de segurança que o DES.

Operando em blocos matriciais de 128 bits e gerando chaves muito maiores, de até 256 bits. Tem o intuito de ser rápido, fazendo operações matemáticas nos elementos matriciais, além de permutações e operações lógicas, dessa forma, substituindo o até aquele momento, atual sistema padrão de criptografia do maior país norte americano, por um sistema com chaves muito maiores e com processamento mais complexo.

Ao falar de complexidade, é intuitivo destacar o algoritmo mais usado no mundo, o RSA. Um modelo de criptografia assimétrica, logo, possui dois tipos de chaves, públicas e privadas. Possui duas chaves públicas que servem para cifrar uma mensagem, e três privadas, que servem para fazer o caminho inverso.

É um modelo baseado em conceitos matemáticos extremamente complexos, abordando conceitos derivados da teoria dos números e problemas que não podem ser solucionados em um tempo polinomial determinístico. Que aliás, é um problema sem solução no ramo de ciência da computação, rendendo prêmios enormes para aquele que o desvende.

A segurança desse método se dá pela dificuldade em fatorar um número em seus componentes primos, principalmente se forem utilizados números primos grandes na realização das operações das chaves. Dessa forma, gerando números tão grandes e complexos, que se torna inviável tentar um ataque de força bruta.

DISSERTAÇÃO

Levando em consideração o objetivo desejado nesse trabalho, que é um algoritmo prático de certa forma, seguro e que permita a comunicação rápida entre dois pontos distintos. Dessa forma, o algoritmo designado para estudo será o RSA.

A estrutura do RSA se da inicialmente por tornar caracteres que serão usados na mensagem em valores numéricos predefinidos, é possível atribuir qualquer valor numérico, mas para exemplificar considere a tabela ASCII como parâmetro, pois além de ser usualmente aceita, também permite uma rápida conversão dos caracteres para números utilizando um espaço na memória do computador que faz isso automaticamente.

Seguindo, a parte principal de uma criptografia RSA que determinará todo o restante do funcionamento do algoritmo, é escolher dois números primos quaisquer. Mas para maior segurança, é recomendado que se utilize números grandes que formem chaves de 2048 bits pelo menos.

Mas para garantir entendimento total dessa técnica, o exemplo a seguir seguirá com números menores. Dessa forma, P = 11 e Q = 13.

Após isso, é necessário multiplicar os dois números escolhidos, formando N = 17 * 41 = 697.

Com isso, é possível calcular a função totiente de N, que determina a quantidade de co-primos de um número que são menores do que ele mesmo. Essa expressão se dá pela fórmula:

$$Z = (p - 1) * (q - 1).$$

$$Z = (11-1) * (13-1)$$

$$Z = 120$$

Faltando apenas duas variáveis antes de cifrar realmente alguma mensagem, a variável E é obtida testando sequencialmente os números primos entre dois e Z, realizando o Máximo divisor comum entre o número gerado sequencialmente e Z. Caso o MDC seja igual a 1, esse número gerado pode ser utilizado como E. Para esse exemplo, E = 7 satisfaz o problema.

Por último, a variável D é obtida realizando testes com números gerados também de forma sequencial, porém, testando se esse número é o inverso multiplicativo modular de E. Com o auxílio da condição a * x ≡ 1 mod m, o valor D = 103 foi obtido.

Já obtidas todas as variáveis necessárias, a cifragem pode ser realizada. Para manter a didática, será utilizado como exemplo a frase teste “Ola, tudo bem?”. Que ao ser codificada utilizando a tabela ASCII, corresponde a “79, 108, 97, 44, 116, 117, 100, 111, 98, 101, 109, 63”.

Para codificar a mensagem é necessário realizar a fórmula C = (X^E) mod N, em que C é o código cifrado e X é o número obtido pela tabela ASCII.

797 mod (143) = 40

977 mod (143) = 59

1167 mod (143) = 129

1007 mod (143) = 100

987 mod (143) = 32

1087 mod (143) = 4

447 mod (143) = 99

1177 mod (143) = 39

1117 mod (143) = 45

Com isso, codificado em RSA ficará:

“40 4 30 99 129 39 100 45 32 62 21 2”.

De forma parecida, para a decifragem do código cifrado basta aplicar o código cifrado na fórmula: De = (X^D) mod N.

40103 mod (143) = 79 4103 mod (143) = 108

59103 mod (143) = 97 99103 mod (143) = 44

129103 mod (143) = 116 39103 mod (143) = 117

100103 mod (143) = 100 45103 mod (143) = 111

32103 mod (143) = 98 62103 mod (143) = 101

21103 mod (143) = 109 2103 mod (143) = 63

Voltando a mensagem pré-codificada em ASCII:

“79 108 97 44 116 117 100 111 98 101 109 63”.

Essa técnica de criptografia oferece uma segurança baseada na dificuldade ainda existente em resolver os conceitos matemáticos propóstos pela fatoração de números primos em um tempo polinomial não determinístico, o famoso problema ainda sem solução P = NP. De forma simples e aplicada a esse contexto, leva muito menos tempo gerar os componentes necessário para formação do algoritmo, do que gerar um algoritmo que gere uma solução em um tempo que pode ser determinado com uma certa exatidão viável.

Por consequência, esse conceito dificulta e torna inviável a tentativa de criptoanálise, trazendo como alternativa o ataque de força bruta, que se aplicado em um algoritmo RSA com chaves de 2048 bits, pode levar cerca de 30 anos para ser quebrado.

Para tornar o RSA mais seguro, além de selecionar números primos grandes como P e Q, essas variáveis podem ser geradas de forma pseudoaleatória ao invés da forma sequencial tratada de forma exemplar ao explicar esse algoritmo. Soma-se a isso, fazer verificações de números também pseudoaleatórios para a variável E, com base também nos critérios de seleção.

Um dos benefícios que o modelo escolhido detém, é a capacidade de manter as chaves privadas secretas de quem o usa. Isso se deve ao fato de o algoritmo ser assimétrico, tornando a comunicação mais segura e prática, não necessitando do compartilhamento com outra pessoa de qualquer chave, que possa descriptografar as mensagens que seriam recebidas pelo dono da chave.

Contudo, basta o portador da chave cifrar a informação com a chave pública do destinatário, que todos tem acesso. E o destinatário usa sua chave privada que é secreta para fazer o caminho inverso. Diferentemente do AES que é necessário o compartilhamento da chave secreta, possibilitando que essa chave caia em mãos erradas.

Essa é uma das vantagens que o RSA tem para com o AES, porém, vale a pena evidenciar que o AES é seguro da mesma forma e tem maior velocidade. Portanto cada um possui suas particularidades e se sobressaem de acordo com determinado contexto específico.

Embora em tese pareça um sistema a prova de falhas, ainda assim, como qualquer algoritmo já criado, possui suas vulnerabilidades, ainda mais considerando que é um dos mais utilizados do mundo. Dessa forma, as pesquisas desse método tendem a ser mais abrangentes e por consequência mais falhas vão surgindo.

Como já mencionado, uma das coisas que garantem toda a segurança desse método, é a utilização de números primos grandes como P e Q. Como todas as outras operações são geradas através da multiplicação deles, quanto maior eles forem, maiores são os números que abrigam os conjuntos de possibilidades de chaves. Com isso, garantindo uma maior segurança em ataques de força bruta, e em criptoanálise, pois para fazê-la é necessário realizar a fatoração dos potencialmente infinitos números primos.

Um uso indevido que pode ser utilizado para evitar a geração de módulos diferentes para cada usuário do algoritmo, é a utilização do mesmo módulo para todos os usuários. Entretanto, ao fazer isso um usuário poderia usar seus próprios expoentes para fatorar o módulo de outros usuários. Devido a isso, é importante não aderir a prática citada acima.

Se tratando das chaves, é importantíssimo estabelecer valores maiores que o comum para ambos os tipos de chaves. No intuito de reduzir o tempo de decifragem e cifragem da mensagem, é intuitivo que defina valores pequenos para a chave privada D e para a chave pública E respectivamente. Assim como citado acima sobre os números primos P e Q, gerar chaves pequenas podem tornar o algoritmo menos inseguro, pois diminui as possibilidades possíveis, diminuindo o tempo levado para quebrar esse sistema, fazendo com que aderir a esta ação seja impossível para uma aplicação real.

Para finalizar esse tópico, outra vulnerabilidade encontrada são os ataques temporais. Eles consistem em monitorar o tempo preciso que um algoritmo RSA leva para executar uma encriptação, dessa forma, é possível reduzir as possibilidades de valores para a chave privada D, quebrando completamente o algoritmo se precisamente executado.

PROJETO (ESTRUTURA) DO PROGRAMA

Após entender os princípios e fundamentos que compõem e diferenciam o algoritmo de criptografia RSA, o próximo passo é criar a estrutura lógica e interface do programa para que seja implementado no código em Python.

Para começar a definir a estrutura do programa, é necessário fazer uma descrição narrativa de seu funcionamento e definir algumas condições. Por se tratar de um modelo experimental funcional dessa técnica, esse programa receberá valores fixos demonstrativos de chaves, porém, com um trecho de código comentado indicando uma sugestão de implementação para melhoria da segurança.

A princípio serão definidos dois números primos gerados pelo código que estará comentado. Esses valores serão gerados através de uma função randômica que verificará se esse valor corresponde a um número primo ou não.

Após isso, esse trecho será marcado como comentado e os valores P e Q estarão definidos. Os valores de N e Z serão calculados pelo programa, através da chamada de uma função que calcule o N e uma outra função que calcula o Z, seguindo a equação conhecida para obter esses valores.

Para a função N, será necessário estabelecer uma variável local auxiliar que receba o resultado da fórmula N = P * Q. Uma variável local N receberá o valor obtido pelo auxiliar e será retornado N para a chamada da função. Já a função que calcula o Z, terá a mesma estrutura da função N, porém, verificará o resultado obtido pela seguinte fórmula Z = (P – 1) * (Q – 1)

O programa terá variáveis globais que armazenarão esses valores para serem utilizados em outras funções mais adiante no programa. E terá outras duas variáveis E e D que correspondem as chaves para cifragem e decifragem da mensagem. Essas duas chaves serão calculadas e verificadas manualmente. O valor obtido será armazenado nas variáveis globais respectivamente E e D.

Todos esses valores obtidos serão utilizados em duas funções, em que, uma serve para cifrar a mensagem e a outra realiza a operação inversa, ou seja, decifra a mensagem.

A lógica por trás da cifragem é a seguinte, realizar um looping que converta cada caractere um por vez da mensagem que o usuário deseja cifrar, em um valor numérico definido pela tabela ASCII. Após converter o caractere correspondente em um valor numérico, realizar a operação seguinte C = (X^E) mod N com esse valor, sendo X o valor numérico correspondente a letra.

Esse valor é armazenado em uma lista na posição de um contador, em que esse contador será incrementado em 1, cada vez que o looping for realizado. O looping será condicionado a um valor limite, que será o tamanho da mensagem digitada. Essa função retornará uma lista com os seus elementos contendo os valores cifrados armazenados.

Já a função que decifra a mensagem, irá ter uma lógica muito parecida com a função de cifragem, tendo como diferença principal a fórmula empregada para obter o valor da mensagem original, que é, De = (X^D) mod N. Esta função retornará a mensagem decifrada em forma de lista, com cada elemento desta lista sendo um caractere original digitado pelo usuário.

Agora a função principal do programa, que fará a parte de menu de interação com o usuário, que chamará todas as outras funções e mostrará seus resultados na tela, de acordo com a vontade do usuário e os valores digitados por ele. Para esse programa experimental, os valores de chave necessários serão mostrados na tela, para teste do usuário.

Inicilamente, essa função será responsável por receber a entrada de dados do usuário. As funções responsáveis por definir os valores das chaves serão chamadas, e em seguida, o programa pedirá a chave pública do usuário que quer cifrar a mensagem. Para manter a didática, o valor da chave pública será informado, e caso digitado corretamente o programa prosseguirá imprimindo na tela a mensagem cifrada. Caso contrário, o programa mostrará uma mensagem de erro e será necessário tentar digitar um novo valor.

A seguir, o programa pedirá que o usuário digite “sim” para cifrar a mensagem de volta, ou “não” para que tudo se encerre. Caso digite um valor que não sejam os dois mencionados acima, será exibido uma mensagem de erro e o usuário terá que tentar novamente.

Caso seja digitado sim, será pedido que o usuário digite a chave privada, que, para esse modelo, será informada a efeito de teste. Caso seja digitado uma chave errada pelo usuário, o programa será encerrado. Em contrapartida, se digitado o valor correto, será mostrado a mensagem decifrada e encerrando as atividades do programa.

RELATÓRIO COM AS LINHAS DE CÓDIGO

A seguir estará o programa funcional, confeccionado no VS Code.

```python
import random

def separar():
    print("-"*100)

def calcular_N(P, Q):
    N = P * Q
    return N

def calcular_Z(P,Q):
    Z = (P - 1) * (Q - 1)
    return Z

P = 11          #Valor definido para o modelo experimental
Q = 13          #Valor definido para o modelo experimental
N = calcular_N(P,Q)
Z = calcular_Z(P,Q)
E = 7     #Valor definido verificado manualmente
D = 103   #Valor calculado por um algoritmo euclidiano extendido fora desse programa.

calcular_N(P,Q)
calcular_Z(P,Q)
```

```python
def cifra(mensagem,E,N):
    limite = len(mensagem)
    cont = 0
    lista = []
    while(cont < limite):
        letra = mensagem[cont]
        X = ord(letra)
        resultado = (X**E) % N
        lista.append(resultado)
        cont += 1
    return lista

def descifra(cifra,N,D):
    lista = []
    i = 0
    tamanho = len(cifra)
    while i < tamanho:
        result = cifra[i]**D
        texto = result % N
        letra = chr(texto)
        lista.append(letra)
        i += 1
    return lista

def invalido():
    print("Opção inválida, tente novamente!")
```

```
def main():
    separar()
    texto_puro = str(input("Digite sua mensagem: "))
    separar()
    print(f"Chave Pública: {E}")
    chave_publica = int(input("Digite a chave pública para cifrar a mensagem: "))
    while chave_publica not in [E]:
        invalido()
        chave_publica = int(input("Digite uma chave pública para cifrar a mensagem: "))
    texto_cifrado = cifra(texto_puro,E,N)
    separar()
    print(f"Sua mensagem cifrada é: {texto_cifrado}")
    separar()
```

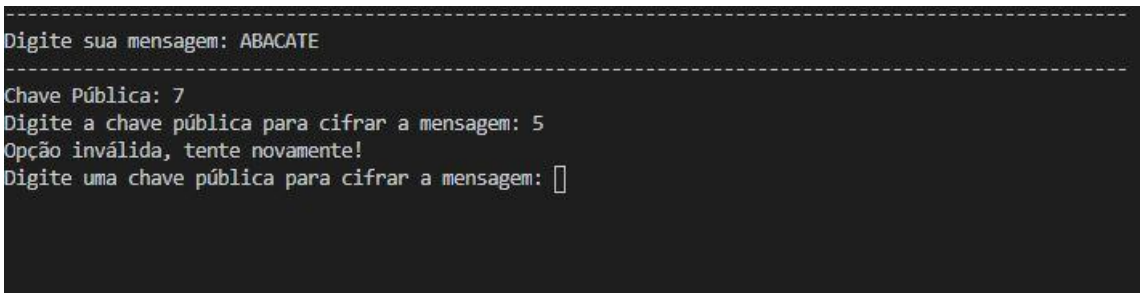
```
    print("Para encerrar o programa digite: nao")
    print("Para decifrar a mensagem digite: sim")
    opcao = str(input("Você gostaria de decifrar essa mensagem?: "))
    if opcao not in ["sim","nao"]:
        while True:
            invalido()
            opcao = str(input("Você gostaria de decifrar essa mensagem?: "))
            if opcao in ["sim", "nao"]:
                break
    if opcao == "sim":
        separar()
        print(f"Chave privada: {D}")
        chave_privada = int(input("Digite a chave privada: "))
        if chave_privada not in [D]:
            print("Você errou, reinicie o programa!")
        else:
            auxiliar_final = descifra(texto_cifrado,N,D)
            separar()
            texto_final = "".join(auxiliar_final)
            print(f"Sua mensagem decifrada é: {texto_final}")
    else:
        print("Fim do programa")

main()
```

Os códigos na página seguinte são soluções de implementações que poderiam ser realizadas futuramente.

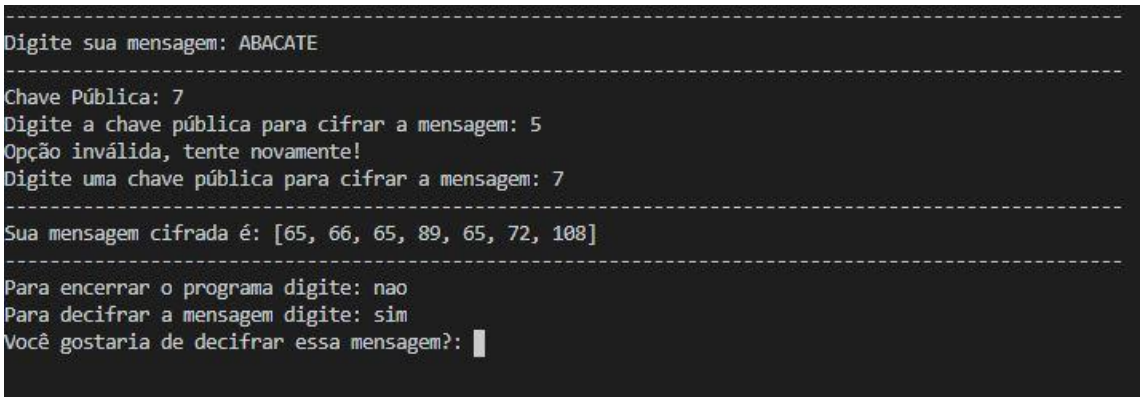
```
"""def gerar_P():"""     #Código ilustrado apenas como
                        #uma possivel implementação para tornar o algoritmo mais seguro
    """auxiliar = random.randint(0,10000)
    while verificar_primo(auxiliar) != 1:
        auxiliar = random.randint(0,10000)
    P = auxiliar"""

"""def gerar_Q():"""         #Código ilustrado apenas como
                        #uma possivel implementação para tornar o algoritmo mais seguro
    """auxiliar = random.randint(0,10000)
    while verificar_primo(auxiliar) != 1:
        auxiliar = random.randint(0,10000)
    Q = auxiliar"""

"""def verificar_primo(x):"""#Código ilustrado apenas como
                        #uma possivel implementação para tornar o algoritmo mais seguro
    """cont = 0
    for c in range(1, x):
        auxiliar = x % c
        if auxiliar == 0:
            cont += 1
    if cont > 1:
        return 0
    else:
        return 1"""
```

```
"""def mdc(a,b):"""         #Código ilustrado apenas como
                        #uma possivel implementação para tornar o algoritmo mais seguro
    """while b !=0:
        resto = a % b
        a = b
        b = resto
    return a"""

"""def calcular_E(Z):"""    #Código ilustrado apenas como
                        #uma possivel implementação para tornar o algoritmo mais seguro
    """while True:
        aleatorio = random.randint(2,Z-1)
        while verificar_primo(aleatorio) != 1:
            break
        else:
            auxiliar = mdc(Z, aleatorio)
            print(f"O mdc deu: {auxiliar}")
            if auxiliar == 1:
                E = aleatorio
                print(f"O e é: {E}")
                return E

calcular_E(Z)"""
```

APRESENTAÇÃO DO PROGRAMA EM FUNCIONAMENTO

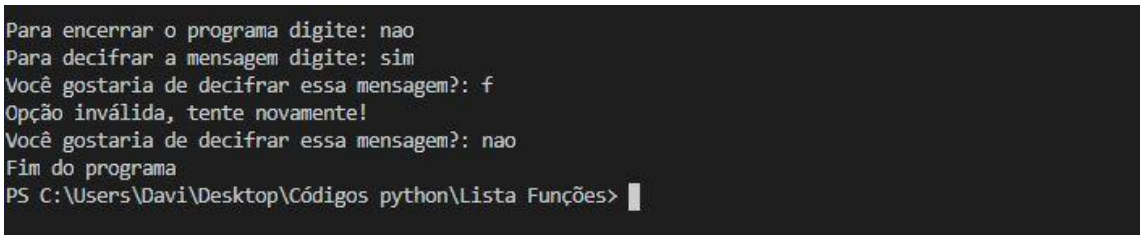


Ao digitar uma chave pública incorreta, o programa pede para tentar novamente.

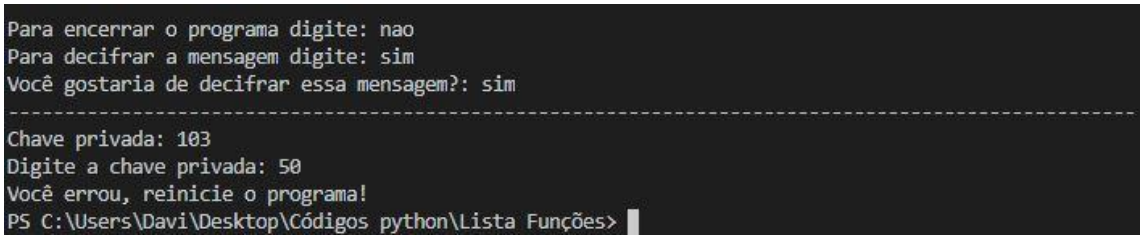


Ao digitar corretamente a chave pública, a mensagem é cifrada e o programa segue.

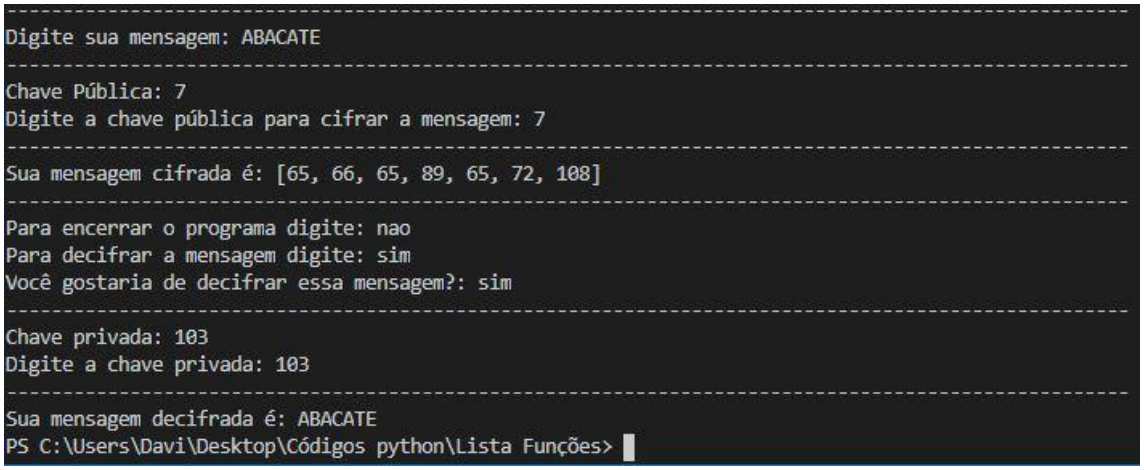
Caso seja digitado uma opção incorreta, o programa pede que tente novamente até digitar “sim” ou “não”.



Se digitado “não”, o programa encerra.



Se digitado “sim”, o programa informa a chave privada, pois se trata de um modelo experimental funcional para teste do algoritmo RSA. Porém se o usuário digitar a chave privada incorreta, o programa pede que ele reinicie o programa para maior segurança.



Por final, o funcionamento completo do programa, cifrando e decifrando uma mensagem.

BIBLIOGRAFIA

https://www.devmedia.com.br/criptografia-conceito-e-aplicacoes-revista-easy-net-magazine-27/26761

https://seguranca-da-informacao.info/criptografia.html

https://medium.com/prognosys/conceitos-b%C3%A1sicos-de-criptografia-624eb6ec3171

https://www.lambda3.com.br/2012/12/entendendo-de-verdade-a-criptografia-rsa/

https://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard

https://www.comparitech.com/blog/information-security/what-is-aes-encryption/

https://br.ccm.net/contents/132-introducao-a-codificacao-des#des-a-codificacao-com-chave-secreta,

https://www.youtube.com/watch?v=CcU5Kc_FN_4

https://www.youtube.com/watch?v=HCHqtpipwu4&t=1265s

https://www.lambda3.com.br/2012/12/entendendo-de-verdade-a-criptografia-rsa-parte-ii/

https://www.lambda3.com.br/2013/01/entendendo-de-verdade-a-criptografia-rsa-parte-iii/

https://pt.wikipedia.org/wiki/MD5

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Davi Coutinho Pires **TURMA:** CC2A33 **RA:** N634774

CURSO: Ciências da computação **CAMPUS:** Tatuapé **SEMESTRE:** 2º **TURNO:** Manhã

CÓDIGO DA ATIVIDADE: 7628 **SEMESTRE:** 2º ano **ANO GRADE:** 2020

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
27/09/2020	Coleta de dados acerca do tema proposto	10			
02/10/2020	Construção dos conceitos gerais	05			
04/10/2020	Pesquisa sobre as técnicas de criptografia mais utilizadas	07			
10/10/2020	Escolha da técnica utilizada	02			
11/10/2020	Dissertação	15			
18/10/2020	Projeto (estrutura) do programa	03			
21/10/2020	Construção do sistema em python 3	25			
25/10/2020	Objetivo	02			
25/10/2020	Introdução	02			
01/11/2020	Formatação da documentação em ABNT	03			
03/11/2020	Verificação de erros	01			

(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: Bruno Soares Carvalho **TURMA:** CC2A33 **RA:** N577690

CURSO: Ciência da Computação **CAMPUS:** Tatuapé **SEMESTRE:** 2º **TURNO:** Matutino

CÓDIGODAATIVIDADE: 76B8 **SEMESTRE:** 2º **ANOGRADE:** 2020/1

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
27/09/2020	Coleta de dados acerca do tema proposto	10			
02/10/2020	Construção dos conceitos gerais	05			
04/10/2020	Pesquisa sobre as técnicas de criptografia mais utilizadas	07			
10/10/2020	Escolha da técnica utilizada	02			
11/10/2020	Dissertação	15			
18/10/2020	Projeto (estrutura) do programa	03			
21/10/2020	Construção do sistema em python 3	25			
25/10/2020	Objetivo	02			
25/10/2020	Introdução	02			
01/11/2020	Formatação da documentação em ABNT	03			
03/11/2020	Verificação de erros	01			

(1)Horasatribuídasdeacordocomoregulamentodas AtividadesPráticasSupervisionadasdocurso.

TOTAL DE HORASATRIBUÍDAS: _____

AVALIAÇÃO: _____

Aprovado ou Reprovado

NOTA: _____

DATA: _____/_____/_____

_____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

UNIP
UNIVERSIDADE PAULISTA

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Caio Silva Pires **TURMA:** CC2A33 **RA:** N661HJ9

CURSO: Ciência da computação **CAMPUS:** Tatuapé **SEMESTRE:** 2º **TURNO:** Matutino

CÓDIGO DA ATIVIDADE: 76B8 **SEMESTRE:** 2º **ANO GRADE:** 2020/1

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
27/09/2020	Coleta de dados acerca do tema proposto	10			
02/10/2020	Construção dos conceitos gerais	05			
04/10/2020	Pesquisa sobre as técnicas de criptografia mais utilizadas	07			
10/10/2020	Escolha da técnica utilizada	02			
11/10/2020	Dissertação	15			
18/10/2020	Projeto (estrutura) do programa	03			
21/10/2020	Construção do sistema em python 3	25			
25/10/2020	Objetivo	02			
25/10/2020	Introdução	02			
01/11/2020	Formatação da documentação em ABNT	03			
03/11/2020	Verificação de erros	01			

(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: Gustavo Behring Araujo **TURMA:** CC2A33 **RA:** N544BH2

CURSO: Ciência da computação **CAMPUS:** Tatuapé **SEMESTRE:** 2º **TURNO:** Matutino

CÓDIGO DA ATIVIDADE: 76B8 **SEMESTRE:** 2º **ANO GRADE:** 2020/1

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
27/09/2020	Coleta de dados acerca do tema proposto	10			
02/10/2020	Construção dos conceitos gerais	05			
04/10/2020	Pesquisa sobre as técnicas de criptografia mais utilizadas	07			
10/10/2020	Escolha da técnica utilizada	02			
11/10/2020	Dissertação	15			
18/10/2020	Projeto (estrutura) do programa	03			
21/10/2020	Construção do sistema em python 3	25			
25/10/2020	Objetivo	02			
25/10/2020	Introdução	02			
01/11/2020	Formatação da documentação em ABNT	03			
03/11/2020	Verificação de erros	01			

(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: Luiz Fernando Silva Santinelli **TURMA:** CC2A33 **RA:** N666099

CURSO: Ciência da computação **CAMPUS:** Tatuapé **SEMESTRE:** 2º **TURNO:** Matutino

CÓDIGO DA ATIVIDADE: 76B8 **SEMESTRE:** 2º **ANO GRADE:** 2020/1

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
27/09/2020	Coleta de dados acerca do tema proposto	10			
02/10/2020	Construção dos conceitos gerais	05			
04/10/2020	Pesquisa sobre as técnicas de criptografia mais utilizadas	07			
10/10/2020	Escolha da técnica utilizada	02			
11/10/2020	Dissertação	15			
18/10/2020	Projeto (estrutura) do programa	03			
21/10/2020	Construção do sistema em python 3	25			
25/10/2020	Objetivo	02			
25/10/2020	Introdução	02			
01/11/2020	Formatação da documentação em ABNT	03			
03/11/2020	Verificação de erros	01			

(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