UNIVERSIDADE PAULISTA

CIÊNCIAS DA COMPUTAÇÃO

ATIVIDADES PRATICAS SUPERVISIONADAS (APS)

INTEGRANTES

JOSÉ EWERTON ALVES DOS SANTOS	**RA: G70JCI9**
EMMANUEL KORZH EMIDIO	RA: R114619
PEDRO HENRIQUE BREIER	RA: R1935F5
PEDRO EDUARDO TEIXEIRA VIEIRA	RA: G1126I0
MARIANA KOBORI GABRIELLI	RA: R192512
ERIC TAKASHI KATAYAMA	RA: G7553I7
MARIANNY STHEFFANNE SANTOS SER	RA: R187187

Tema: IMPLEMENTAÇÃO UM SISTEMA PARA A TRANSMISSÃO DOS VOTOS DE UMA URNA ELETRÔNICA PARA A CENTRAL DE APURAÇÃO

ORIENTADOR(A):

SÃO PAULO

2024

Sumário

1 INTRODUÇÃO

Na computação, informação são dados que foram processados em um computador. Quando se trata de segurança, informação tem significado mais amplo. Informação neste caso é um ativo de uma organização (podendo esta organização ser do governo ou uma empresa privada) e ela pode ser essencial para o sigilo de informações públicas ou negócios de uma empresa privada e por este motivo deve ser protegida. Por este motivo, a segurança da informação tem como meta a proteção dos sistemas de informação contra a invasão e modificação dos dados por pessoas não autorizadas. A segurança da informação deve prevenir, detectar, deter e documentar qualquer ameaça aos seus dados e processamento. Vários fatores podem colocar em risco as informações computacionais como comportamento humano indevido dos detentores da informação, problemas no ambiente no qual a informação está inserida, falhas de infraestrutura e indivíduos mal intencionados que agem com o intuito de alterar, destruir, danificar ou roubar as informações.

Nos sistemas democráticos modernos, a integridade e a segurança dos processos eleitorais são fundamentais para assegurar a confiança da população nos resultados e na legitimidade dos representantes eleitos. Segundo a constituição da república dos estados unidos do Brasil, Título I da organização Federal, Capítulo II do poder legislativo, secção I disposições preliminares art. 38 o voto será secreto nas eleições e nas deliberações sobre vétos e contas do presidente da república o que significa que métodos de segurança devem ser implementados para a integridade dos votos.

À medida que as tecnologias digitais avançam, novos desafios surgem para proteger o sigilo e a autenticidade dos votos, garantindo que cada cidadão exerça seu direito de forma segura e inquestionável. Entre os principais dispositivos utilizados, a urna eletrônica se destaca como uma ferramenta essencial para a coleta digital e automatizada dos votos, facilitando tanto o processo de votação quanto a apuração rápida dos resultados. No entanto, como qualquer sistema informatizado, a urna eletrônica também está suscetível a tentativas de fraudes e manipulações, o que ressalta a importância de implementar técnicas de proteção de dados que sejam robustas e confiáveis.

A criptografia e o hash são duas das técnicas mais eficazes e amplamente utilizadas para esse propósito. A criptografia, por sua vez, é o processo que transforma as informações para que apenas pessoas ou sistemas autorizados possam interpretá-las. Em uma urna eletrônica, a criptografia assegura que os votos, uma vez registrados, sejam mantidos de maneira sigilosa e inacessível, mesmo que a transmissão de dados ocorra por meio de redes sujeitas a interceptações. Já o hash, um algoritmo que gera uma espécie de "impressão digital" dos dados, tem um papel de extrema importância para verificar a integridade das informações, ou seja, para assegurar que os votos não foram alterados ou corrompidos em qualquer ponto do processo. Com o uso do hash, qualquer alteração mínima nos dados de origem é identificada, alertando os sistemas para possíveis tentativas de manipulação.

O objetivo desta Atividade Prática Supervisionada (APS) é desenvolver um sistema simplificado de urna eletrônica que incorpore esses princípios básicos de segurança digital. Através do uso de criptografia para proteger os dados dos votos e do algoritmo de hash SHA-512 para garantir a integridade dos arquivos, o projeto busca simular um processo de votação eletrônica seguro e confiável. Para isso, a atividade foi dividida em duas etapas principais: no primeiro programa, será realizado o cadastro dos votos, seguido pela criptografia desses dados com uma chave simples. Em seguida, o programa gera um hash SHA-512 do arquivo criptografado, que será armazenado e utilizado para verificar a integridade dos votos posteriormente. No segundo programa, a responsabilidade será validar o hash gerado, descriptografar os votos e apurar os resultados finais, assegurando que nenhuma modificação ocorreu desde o momento do registro dos votos.

Essa proposta, além de fornecer uma experiência prática sobre segurança digital aplicada, inclui uma análise dos conceitos teóricos e práticos relacionados à criptografia e às funções de hash. Com isso, busca-se discutir as vantagens e desvantagens das abordagens adotadas, destacando os desafios e limitações que cada técnica pode apresentar em um contexto real. Por exemplo, no caso da criptografia simétrica (a abordagem escolhida para este projeto), a segurança está diretamente vinculada ao sigilo da chave utilizada para codificar e decodificar os votos. Se essa chave for comprometida, o sistema inteiro fica vulnerável, o que enfatiza a necessidade de boas práticas na geração e armazenamento das chaves. Já no caso do hash SHA-512, a técnica oferece uma robusta garantia de integridade, mas não

possui reversibilidade, o que a torna ideal para verificação, mas não para recuperar dados originais.

Em paralelo, o desenvolvimento desta urna eletrônica simplificada também permite uma aplicação prática dos conceitos de programação estruturada em Python, promovendo um alinhamento entre teoria e prática. Com isso, o projeto pretende não só demonstrar a aplicabilidade e importância da segurança digital em sistemas críticos, como o eleitoral, mas também explorar os elementos da programação de uma maneira acessível e funcional. Através do código, o objetivo é capacitar os envolvidos com o entendimento da estrutura lógica de um sistema de votação, promovendo uma base sólida para quem deseja aprofundar-se no tema.

Este projeto visa, portanto, contribuir para o desenvolvimento de uma solução que seja prática e eficaz, apresentando de forma didática o impacto e a relevância da criptografia e da integridade de dados em sistemas de alto nível de criticidade. Além disso, a aplicação dos conceitos neste sistema eleitoral fictício demonstra como é possível garantir que cada voto registrado se mantenha fiel ao que o eleitor escolheu, servindo como uma base de estudo para aplicações mais complexas de segurança digital.

2 REFERENCIAL TEÓRICO

2.1 Descrição do problema computacional

Não é uma tarefa trivial abordar o tema da lisura do processo eleitoral, especialmente devido ao impacto que ele tem sobre a legitimidade do sistema democrático. Qualquer brecha nesse processo pode comprometer a confiança da população e levantar questionamentos sobre a legitimidade das instituições. Para garantir a segurança do sufrágio, a Justiça Eleitoral, por meio do Tribunal Superior Eleitoral (TSE), implementou diversos mecanismos de criptografia ao longo dos anos.

Entre as ferramentas utilizadas estão a **criptografia assimétrica**, que emprega chaves públicas para autenticar dados e chaves privadas para descriptografar informações sensíveis. Em conjunto com algoritmos de hash, essas técnicas garantem que os dados inseridos pelos eleitores sejam transformados em uma sequência única de caracteres, a qual só pode ser replicada se o processo for realizado sob as mesmas condições.

2.2 O que é Criptografia?

Segundo a Google, "a criptografia é o processo de proteger informações ou dados utilizando modelos matemáticos para embaralhá-los, de forma que apenas as partes que possuem a chave correta possam acessá-los" (O que é criptografia?, 2024). Embora a criptografia seja amplamente vista como uma inovação moderna, sua origem remonta a períodos anteriores a Cristo, como discutido no livro *The Codebreakers*, que oferece uma análise histórica detalhada do uso da criptografia ao longo dos séculos.

2.3 Integridade de Dados

A **integridade de dados** é a capacidade de garantir que uma mensagem ou informação chegue ao seu destino sem alterações em relação ao estado original. A integridade é essencial na criptografia, pois assegura que os dados transmitidos não sejam modificados durante o processo de comunicação. Sem essa garantia, a criptografia perderia sua utilidade, já que a autenticidade das mensagens estaria comprometida.

2.4 Autenticação

Autenticação é o processo de verificar se algo ou alguém é genuíno e confiável, seja um interlocutor ou uma mensagem. No contexto da urna eletrônica, a autenticação garante que os votos recebidos sejam legítimos e que apenas fontes autorizadas possam acessá-los.

2.5 Criptografia Simétrica

De acordo com a Amazon, "a criptografia simétrica utiliza a mesma chave tanto para encriptar quanto para descriptografar os dados" (O que é criptografia? 2024). Em outras palavras, a mesma chave é usada na origem e no destino da mensagem, desempenhando um papel duplo.

2.6 Criptografia Assimétrica

A **criptografia assimétrica** utiliza um par de chaves: uma pública e uma privada. A chave pública é usada para verificar a autenticidade da mensagem, enquanto a chave privada, mantida em segredo, é necessária para descriptografar os dados.

2.7 Hash

Um **hash** é uma função criptográfica que transforma uma informação em um valor único e de tamanho fixo. A principal característica do hash é que ele não pode ser revertido: uma vez que o dado é convertido, não há como recuperá-lo a partir do valor gerado. A única maneira de obter o mesmo hash seria fornecer exatamente a mesma entrada original.

2.8 Exemplos de Aplicação no Contexto da Urna Eletrônica

2.8.1 Criptografia Simétrica

Na urna eletrônica, a criptografia simétrica pode ser utilizada para criptografar rapidamente os votos antes de armazená-los. Um exemplo seria o uso da operação **XOR**:

Texto original (em binário): 1101 (representando um voto simplificado).

Chave: 1010

Texto criptografado (XOR): 0111

Durante a apuração, o mesmo processo de XOR é usado para descriptografar e recuperar o valor original.

Vantagens:

Simplicidade e alta eficiência computacional.

Desvantagens:

A segurança depende da proteção da chave. Se comprometida, todo o sistema é vulnerável.

2.8.2 Criptografia Assimétrica

Com a criptografia assimétrica, o arquivo de votos pode ser criptografado usando uma chave pública do tribunal eleitoral. Mesmo que alguém intercepte o arquivo, ele só poderá ser descriptografado com a chave privada do tribunal.

Vantagens:

Maior segurança no transporte de dados.

Controle mais refinado de acesso às informações.

Desvantagens:

Mais lenta e consome mais recursos computacionais.

2.9 Comparação de Vantagens e Desvantagens dos Métodos

A Tabela 1 apresenta uma comparação de cada método criptográfico incluindo suas vantagens e desvantagens.

Tabela 1 Vantagens e desvantagens das metodologias de cada método de criptografia

Método	**Vantagens**	**Desvantagens**
Simétrica	Rápida e fácil de implementar.	Requer que a chave seja mantida em segredo.
	Consome poucos recursos computacionais.	Se a chave for comprometida, o sistema falha.
Assimétrica	Maior segurança, mesmo com a chave pública exposta.	Mais lenta e consome mais recursos.
	Comunicação segura entre partes desconhecidas.	Complexa de implementar.
Hash	Garante a integridade dos dados.	Não fornece confidencialidade.
	Uma alteração mínima nos dados muda completamente o hash.	Não é reversível.

2.10 Aplicação do Hash no Contexto da Urna

O hash pode ser usado para verificar se os votos foram alterados durante a transmissão. O processo funciona da seguinte forma:

O programa calcula o hash do arquivo de votos no momento em que ele é salvo.

Durante a apuração, o hash é recalculado e comparado ao valor original. Se os hashes coincidirem, a integridade dos dados está preservada.

Exemplo:

Arquivo original: "votoA,votoB,votoC"

Hash gerado: 12345

Se o arquivo for alterado para "votoA,votoX,votoC", o novo hash será diferente, como 67890, indicando que os dados foram adulterados.

Vantagens:

Simples e eficiente para garantir a integridade dos dados.

Desvantagens:

Não garante confidencialidade. Mesmo que o hash detecte alterações, qualquer pessoa pode ler os dados originais.

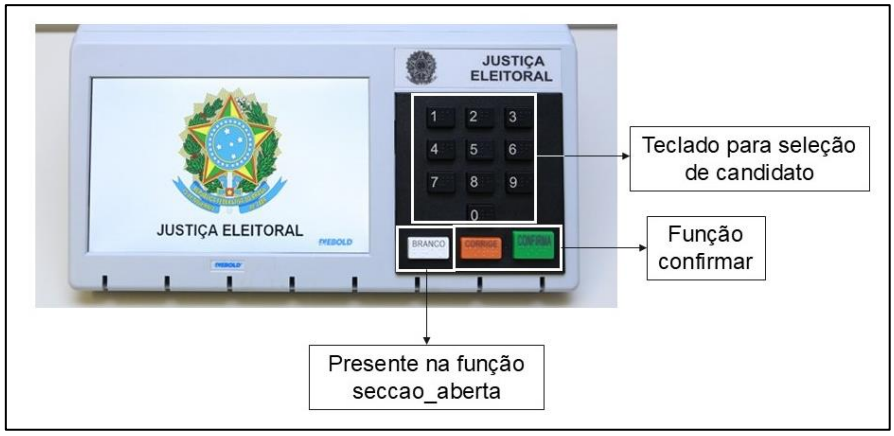
3 DESENVOLVIMENTO

3.1 Detalhamento da solução

Para o desenvolvimento de uma simulação de uma urna eletrônica foi utilizado a linguagem de programação Python pelo Visual Studio Code. Para este trabalho, foi considerado uma situação de segundo turno entre dois candidatos hipotéticos: Tião do Gás e Zé da Feira. Para o programa de urna eletrônica foram criados várias funções e procedimentos que foram utilizadas dentro dos programas principais. A seguir foi descrito todas as função e procedimento.

Para garantir a segurança das informações, no caso o voto do eleitor, foi aplicado o método de criptografia simétrica por sua facilidade de implementação em conjunto com o Hash que por ser considerado um método irreversível o que garante que os dados criptografados se mantenham íntegros.

Figura 1 – Imagem de uma urna eletrônica e o que cada os botões representam no programa da urna.



Fonte: adaptado de Tribunal Superior Eleitoral (2022)

Uma das funções presente no programa é a RESULTADO que contabiliza a quantidade de votos recebida por cada candidato. Nesta função criou-se uma lista (equivalente a um vetor) e nela estava contida cada candidato que podia receber voto e criou-se também um dicionário armazenou a quantidade de votos que cada candidato podia receber. Tanto na lista quanto no dicionário foi também incluído a quantidade de votos brancos, nulos e o número total de votos. Esta função retorna um valor de contagem para um candidato.

O botão confirmar (verde) e corrigir (vermelho) presentes na urna eletrônica (Figura 1) foram representados no programa pela função CONFIRMACAO. O objetivo desta função foi para permitir que o eleitor corrija ou confirme um voto. Esta função retorna um valor que define que um candidato (ou nenhum por branco e nulo) receba um voto.

A escolha de um candidato pelo eleitor foi determinada pela função VOTO. Nesta função foi determinado a numeração de cada candidato que o eleitor digitaria na urna eletrônica (Figura 1) e também o botão voto em branco. Para este trabalho foi determinado que o candidato número 1 seria Tião do gás, o candidato número 2 Zé da feira, o número 3 o “candidato” branco e qualquer outro valor o “candidato” nulo. Esta função chama a função CONFIRMACAO, o que significa que ela fornece um voto a um dos candidatos. Essa função retorna o voto_registrado que registra no dicionário um voto a um candidato.

A próxima função do programa foi CRIPTOGRAFAR_VOTOS. Essa função chama a função RESULTADO chamando os votos de cada candidato, votos brancos, nulos e votos totais e transformando em um texto string. Cada caractere do texto foi transformado em algum outro caractere segundo alguma metodologia e chave criptográfica. O retorno desta função foi o texto contendo a informação dos votos criptografados.

A função GERAR_OS_HASH utilizou o identificador do processo (PID) junto com as informações criptografadas para gerar o valor hash. O valor de hash foi utilizado para validar os arquivos criptografados e rejeitar alterados e corrompidos. O retorno desta função foi o valor hash.

Em seguida foi feito o procedimento SALVAR_ARQUIVO no qual um arquivo foi gerado contendo as informações importantes para a validação da criptografia e descriptografia. Dentre essas informações estavam os resultados dos votos criptografados, o valor de hash e o PID.

O primeiro programa principal foi responsável pela contagem de voto, pela criptografia das informações de votos, pela geração de hash e para arquivamento das informações. Este primeiro programa principal foi realizado por um procedimento chamado de FUNCAO_PRINCIPAL. Em uma situação de eleição real este procedimento é realizado pelos eleitores que decidem em quem irão votar e pelos mesários que determinam se há ou não mais eleitores para votar. O programa primeiramente pergunta se há um eleitor ou se deve encerrar a secção. Havendo

eleitores, o procedimento chama a função VOTO para que o eleitor possa votar em seu candidato. Quando não houver mais eleitores, o mesário finaliza a seção. Com a seção finalizada, o programa chama automaticamente a FUNÇÃO RESULTADO, e em seguida a função CRIPTOGRAFAR_VOTOS para criptografar o resultado da suposta eleição. Em seguida o programa chama a função GERAR_OS_HASH para gerar o hash que tem o objetivo de validar as informações e por último chama o procedimento SALVAR_ARQUIVO para salvar todas as informações importantes.

A descriptografia das informações e a validade do mesmo foi realizado pelo procedimento DESCRIPTOGRAFAR_E_VALIDAR. O procedimento verificou as informações salvas após a criptografia e realizou a descriptografia utilizando a chave criptográfica. Simultaneamente, ele recalculou o hash chamando a função GERAR_OS_HASH e fez uma comparação deste novo hash calculado com o hash salvo no arquivo após a criptografia. Caso o hash seja o mesmo, o programa finaliza a descriptografia. Caso contrário, ele considera o arquivo como corrompido e não realiza a descodificação.

O segundo programa principal é um procedimento chamado de MAIN. Este programa teria que ser acionado antes de iniciar a votação e no momento em que for apurar os votos. O programa é um menu que decide a eleição iria começar ou fazer a descriptografia junto com a validação dos resultados. Se a opção selecionada for o de iniciar votação o programa chama o procedimento FUNÇÃO PRINCIPAL. Caso seja para apurar os votos, o programa chama o procedimento DESCRIPTOGRAFAR_E_VALIDAR.

3.2 Técnicas utilizadas

Para realizar a criptografia dos resultados da eleição hipotética e a descriptografia do mesmo foi utilizado a chave criptográfica e a metodologia XOR para a substituição de caractere. A criptografia dos resultados ocorre pela Equação 1, já na linguagem Python.

```
votos_criptografados +=chr(ord(char)^chave)
```

Equação 1

Na Equação 1, é possível verificar que a criptografia vai se formando por caractere por caractere. O símbolo de circunflexo (^) indica que o caractere está sendo

substituído utilizando o método XOR e o que define a distância para o próximo caractere a ser substituído é o valor da chave.

O valor hash foi determinado sequencialmente pela Equação 2 e pela Equação 3. A Equação 2 determina o valor de hash de acordo com uma randomização do valor de PID. A Equação 3 realiza uma rotação de bits para determinar em qual posição está o caractere que irá substituir (os operadores >> e << determinam para a direita e para a esquerda). O valore de hash retornado funciona como uma string hexadecimal.

$$hash\_valor=(hash\_valor+ord(char)+os.getpid())\%253 \quad \text{Equação 2}$$

$$hash\_valor=(hash\_valor\ll 1))|(hash\_valor\gg 7) \quad \text{Equação 3}$$

Para a seleção de candidatos na função VOTO e para o menu de opções no procedimento MAIN, o comando “if/elif/else” foi utilizado. Em ambos os casos, este comando equivale ao comando case. O comando “for” foi utilizado na função CRIPTOGRAFAR_VOTOS, na função GERAR_OS_HASH e no procedimento DESCRIPTOGRAFAR_E_VALIDAR para gerar tanto a informação criptografada quanto para gerar um texto com a informação descriptografada. O comando “while” foi utilizado para manter o voto aberto e para manter a secção aberta respectivamente nas funções VOTO e FUNCAO_PRINCIPAL e para adicionar voto a um candidato na FUNCAO_PRINCIPAL. Listas e dicionários foram utilizados para armazenar informações da contabilização de votos e para guardar os nomes dos candidatos.

O código do programa principal utilizado neste trabalho se encontra no Apendice. Nele está detalhado cada procedimento utilizado para a contabilização de votos na urna eletrônica informações de como foi feito a criptografia e a descriptografia e a geração de hash para a validação dos resultados.

4 DISCUSSÃO

4.1 Testes realizados

Para a realização de testes da urna eletrônica, foram selecionadas três situações hipotéticas de resultados eleitorais. Essas condições estão apresentadas na Tabela 2.

Tabela 2 – Simulação de resultado de eleições

	Número de votos		
	Simulação 1 e 4	Simulação 2 e 5	Simulação 3 e 6
Tião do gás	52	54	39
Zé da Feira	48	46	61
Branco	11	7	14
Nulos	19	18	8
Total	130	125	113

Os resultados gerados a partir da simulação de resultados da Tabela 2 utilizando a chave criptográfica com valor de 12345 estão apresentados na Tabela 3. Foi possível observar que com metodologia de aplicar XOR em conjunto com a chave criptográfica e a Equação 1 em cada caractere resultou em um conjunto de letras e símbolos diferente do alfabeto latino. Os anagramas gerados em questão são letras do alfabeto japonês.

O valor de hash foi gerado através da Equação 2. Esta equação gera valores aleatórios utilizando o valor de PID. Por este motivo, era de se esperar que em cada simulação de votos fossem gerados diferentes valores de hash. Para verificar se os valores de hash eram únicos a simulação 3 foi realizada em duplicata. Esperava-se que nas simulações em duplicatas fossem gerados a mesma informação criptografada com diferentes valores de PID e hash. Os resultados obtidos pela codificação dessas simulações estão presentes na Tabela 3 e foram nomeadas de simulação 1, simulação 2, simulação 3-1 e simulação 3-2, sendo que esses dois últimos deveriam possuir as mesmas informações criptografadas.

Outro teste também foi realizado utilizando outra chave criptográfica com o valor 23648. As relações de votos utilizados nestas simulações foram o da Tabela 2.

O motivo desta simulação foi para verificar a variação dos caracteres com a mudança do valor da chave criptográfica. Para este caso, as simulações foram chamadas de simulação 4, simulação 5, simulação 6-1 e simulação 6-2. A simulação 6 foi realizada em duplicata para comparar dois resultados criptografados iguais e para verificar se os hash gerados foram diferentes. Os dados criptografados para esta nova chave podem ser visualizados na Tabela 4.

Todos os arquivos das simulações foram salvos. Nesses arquivos eram contidas informações criptografadas junto com o PID e o hash. Com esses arquivos foram realizadas outras simulações para verificar a validade do hash e do PID e caso o arquivo estivesse intacto a descriptografia seria realizada. Uma outra simulação também foi feita no qual foi alterado propositalmente as informações criptografadas no arquivo salvo mantendo o PID e o hash. Neste caso, era de se esperar que a validação não ocorresse pois o arquivo foi corrompido.

4.2 Resultados obtidos

Os resultados das simulações estão apresentados na Tabela 3 com a chave criptográfica 12345 e na Tabela 4 com a chave criptográfica 23648. Pode-se notar que em ambos os casos, as informações criptografadas não estão dispostas no alfabeto latino e isso pode ser considerado uma vantagem de se utilizar o método XOR com as chaves criptográficas, já que desta forma uma descriptografia por força bruta torna-se mais difícil e improvável, pois é necessário que um cracker tenha conhecimento não só em descriptografar, mas também de diferentes simbologias e diferentes alfabetos e anagramas.

Na Tabela 3, pode-se notar que as informações criptografadas com a chave 12345 foram convertidas em informações com o alfabeto japonês. Com essa chave combinado com a criptografia XOR, pôde-se encontrar um padrão nas informações criptografadas. As letras formadas em nas simulações de 1, 2 e 3 foram praticamente as mesmas. O que difere na criptografia de uma simulação para outra são os símbolos (〈 〈〕 〃〗). Por esse padrão, foi possível deduzir que as letras do alfabeto japonês representam o nome dos candidatos e os símbolos representam o número de votos.

Tabela 3 – Resultados apresentados com após a simulação dos votos apresentados na Tabela 2 utilizando a chave 12345.

	Resultados criptografados
Simulação 1	ねぐぺざ〗そざ〗まへお〃〗「》〕〗っバ〗そじ〗みぜぐかじ〃〗」、〕〗ほかじしずざ〃〗〈〈〕〗ぷがさざ〃〗〈　〕〗ねざきじさ〃〗〈《〉 Hash: 0x115 PID: 11852
Simulação 2	ねぐぺざ〗そざ〗まへお〃〗「」〕〗っバ〗そじ〗みぜぐかじ〃〗」』〕〗ほかじしずざ〃〗『〕〗ぷがさざ〃〗〈、〕〗ねざきじさ〃〗〈》「 Hash: 0xe PID: 20316
Simulação 3-1	ねぐぺざ〗そざ〗まへお〃〗《　〕〗っバ〗そじ〗みぜぐかじ〃〗』〈〕〗ほかじしずざ〃〗〈」〕〗ぷがさざ〃〗、〕〗ねざきじさ〃〗〈》》 Hash: 0x133 PID: 3908
Simulação 3-2	ねぐぺざ〗そざ〗まへお〃〗《　〕〗っバ〗そじ〗みぜぐかじ〃〗』〈〕〗ほかじしずざ〃〗〈」〕〗ぷがさざ〃〗、〕〗ねざきじさ〃〗〈》》 Hash: 0x1b9 PID: 11936

De maneira similar, a Tabela 4 apresentou a informação criptografada com um outro alfabeto, que no caso foi o anagrama chinês. As informações das simulações 4, 5 e 6 também foram similares entre elas, com a maioria das letras iguais e com alguns ideogramas diferentes que possivelmente representam o número de votos. Esses ideogramas que provavelmente representam números de votos estão sublinhados na Tabela 4.

No caso das simulações em duplicatas, as informações criptografadas foram idênticas entre as simulações 3-1 e 3-2 e entre as simulações 6-1 e 6-2. Isso significa que todas as informações que foram criptografadas foram as mesmas, tanto o nome dos candidatos quanto o número de votos de cada candidato. Pôde-se notar também que PID e o hash foram únicos para cada simulação. Esse resultado também era o esperado, já que este resultado é único para cada simulação já que estes valores foram os responsáveis para validar a criptografia e a descriptografia.

Como já mencionado, o fato de haver um padrão perceptível das informações criptografadas pode ser considerado uma fraqueza, já que aumenta a possibilidade de uma decriptação por força bruta. Há duas maneiras que de se contornar este

problema. Um deles é utilizando chaves diferentes em cada eleição. Ao se utilizar chaves diferentes o padrão de informações criptografadas iria se alterar o que dificultaria a quebra da criptografia. Outra alternativa é alterar o método de criptografia em cada eleição, no caso, utilizar um método diferente de XOR.

Tabela 4 – Resultados apresentados com após a simulação dos votos apresentados na Tabela 2 utilizando a chave 23648.

	Resultados criptografados
Simulação 4	瞭晅昏晖墅暈晖墅暫胆暟曖墅曙曞暟墅璇昅墅暈暍墅暪暉晅暝暍曖墅曘暸暟墅暮暝暍暫暩晖曖墅曝曝暟墅暢暔眐晖曖墅曝瞻暟墅暸晖暘暍眐曖墅曝晨曜 Hash: 0x1d1 PID: 3776
Simulação 5	瞭晅昏晖墅暈晖墅暫胆暟曖墅曙曘暟墅璇昅墅暈暍墅暪暉晅暝暍曖墅曘曚暟墅暮暝暍暫暩晖曖墅曛暟墅暢眐晖曖墅曝曙 Hash: 0x18f PID: 24248
Simulação 6-1	瞭晅昏晖墅暈晖墅暫胆暟曖墅晨瞻暟墅璇昅墅暈暍墅暪暉晅暝暍曖墅曚曝暟墅暮暝暍暫暩晖曖墅曝曘暟墅暢眐晖曖墅曝曞 Hash: 0x66 PID: 20256
Simulação 6-2	瞭晅昏晖墅暈晖墅暫胆暟曖墅晨瞻暟墅璇昅墅暈暍墅暪暉晅暝暍曖墅曚曝暟墅暮暝暍暫暩晖曖墅曝曘暟墅暢眐晖曖墅曝曞 Hash: 0x18f PID: 8044

Para descriptografar as informações, foi necessário utilizar a mesma chave utilizada para criptografar e aplicar novamente o método XOR em cada caractere. Com isso a informação foi descriptografada. Para validar os votos, foi necessário utilizar o PID e o hash das informações que foram arquivadas após o fechamento da secção eleitoral. Com o PID arquivado calculou-se um novo hash e este novo hash foi comparado com o hash gerado no momento da criptografia dos votos. Se os dois hash apresentassem os mesmos valores, as informações seriam descriptografadas e validadas. A Tabela 5 mostra que as informações das simulações de 1 a 6 foram validadas e consequentemente descriptografadas.

Tabela 5 – Votos validados após descriptografar os votos.

	Validação e descriptografia
Simulação 1 **Simulação 5**	Arquivo verificado com sucesso. Hash corresponde. Votos descriptografados: Tião do Gás: 52, Zé da Feira: 48, Branco: 11, Nulo: 19, Total: 130
Simulação 2 **Simulação 5**	Arquivo verificado com sucesso. Hash corresponde. Votos descriptografados: Tião do Gás: 54, Zé da Feira: 46, Branco: 7, Nulo: 18, Total: 125
Simulação 3 1 e 3-2 **Simulação 6-1 e 6-2**	Arquivo verificado com sucesso. Hash corresponde. Votos descriptografados: Tião do Gás: 39, Zé da Feira: 61, Branco: 14, Nulo: 8, Total: 122
Simulação 7	Erro: Hash não corresponde. Arquivo pode estar corrompido.

Foi realizado um teste adicional, nomeada simulação 7 (Tabela 5), no qual foi modificada as informações criptografadas contida nos arquivos salvos. Neste teste, as informações não foram descriptografadas já que o programa identifica corretamente que o as informações criptografadas foram corrompidas. Pode-se confirmar que a simulação do programa de urna eletrônica mantém o sigilo e confiabilidade e não permite que as informações após serem criptografadas sejam validadas.

5 CONCLUSÃO

A realização deste projeto de urna eletrônica com criptografia e hash proporcionou uma compreensão mais profunda sobre os desafios e soluções na segurança digital aplicada a sistemas críticos. A integração da criptografia com o método XOR e a implementação do hash SHA-512 baseado no identificador de processo (PID) foram passos importantes para simular, de forma simplificada, um sistema eleitoral capaz de proteger os dados dos votos e assegurar sua integridade contra alterações.

Ao longo do desenvolvimento, dividimos o projeto em duas etapas principais: o registro e criptografia dos votos e a posterior validação e apuração dos resultados. Essa divisão foi essencial para criar um fluxo seguro, onde a verificação de integridade é aplicada no momento adequado para garantir que os votos não sejam adulterados após serem criptografados. Durante os testes, a funcionalidade de hash demonstrou sua eficácia em identificar mudanças nos dados originais, o que reforça a importância desse mecanismo como uma camada de segurança indispensável em sistemas eleitorais.

Além disso, a experiência evidenciou a relevância da escolha cuidadosa das técnicas de criptografia. O uso do método XOR, embora eficiente para simulações como esta, apresentou vulnerabilidades em termos de previsibilidade. Este tipo de criptografia é vulnerável quando a mesma chave é aplicada de forma contínua, o que gera padrões que podem ser explorados. Em um cenário real, a adoção de algoritmos de criptografia mais robustos, como o AES, ou a troca periódica de chaves criptográficas seria necessária para evitar esses padrões e oferecer uma camada adicional de segurança. Tais alternativas destacam como a tecnologia deve sempre evoluir para acompanhar as ameaças emergentes e manter a confiabilidade dos sistemas críticos.

Este estudo prático também ressaltou a importância da transparência e da verificabilidade dos processos eleitorais, fatores essenciais para a confiança da população no sistema democrático. Ao garantir que cada voto seja registrado, transmitido e armazenado de forma segura, tecnologias como criptografia e hash ajudam a construir um sistema eleitoral em que os resultados são inquestionáveis. Embora nosso sistema seja uma implementação simplificada, ele serve como um

modelo pedagógico para entender como técnicas de segurança digital podem ser aplicadas para evitar manipulações e acessos não autorizados.

Concluindo, este projeto mostrou que, mesmo em um cenário simplificado, é possível criar um sistema de votação digital seguro com o uso de programação estruturada e técnicas básicas de segurança. A experiência contribuiu significativamente para a formação em programação e segurança digital, e reforçou a importância de desenvolver sistemas que, além de funcionais, sejam projetados para proteger a integridade dos dados. Em um contexto de eleição, onde a legitimidade dos resultados é crucial para a democracia, a aplicação dessas técnicas ajuda a aumentar a confiança dos eleitores e a proteger a voz de cada cidadão. Esse projeto deixa claro que a segurança digital é uma área vital para o avanço e fortalecimento dos processos democráticos e que, com o avanço da tecnologia, a busca por soluções mais seguras e eficientes deve continuar sendo uma prioridade para as sociedades modernas.

6 ESTRUTURA DO PROGRAMA

Neste tópico, apresentaremos a estrutura do programa da urna eletrônica desenvolvida, descrevendo suas principais funcionalidades e como elas se integram para compor o fluxo de criptografia, validação e apuração dos votos. O programa foi implementado em duas etapas principais: o registro e a criptografia dos votos e, posteriormente, a validação e a apuração dos resultados, ambos descritos em detalhes a seguir.

6.1 Registro e Criptografia dos Votos

Na primeira etapa, o programa realiza o registro e a criptografia dos votos. Esse processo inicia com a entrada de votos, onde cada voto é registrado e associado ao candidato escolhido. Após o registro, os dados dos votos são submetidos ao algoritmo de criptografia.

- **Criptografia com XOR**: Para proteger os dados dos votos, utilizamos a técnica de criptografia XOR, combinada com uma chave criptográfica que o usuário deve fornecer. Essa chave é aplicada em cada caractere dos votos registrados, convertendo-os em uma sequência de caracteres irreconhecíveis que simulam letras e símbolos do alfabeto japonês ou chinês, dependendo da chave escolhida. Esse processo gera um padrão criptográfico, dificultando a leitura e interpretação dos dados sem a chave correta.
    - **Geração de Hash SHA-512**: Simultaneamente, o programa gera um valor de hash único para cada registro criptografado. Esse hash é gerado utilizando o identificador de processo (PID), garantindo que cada registro seja único e associando o voto ao contexto do sistema no momento da criptografia. Esse valor de hash é armazenado junto com os dados criptografados para ser utilizado na etapa de validação.

O arquivo resultante contém os votos criptografados, o PID e o hash, sendo salvo no sistema de arquivos para garantir a integridade dos dados após o encerramento da seção eleitoral.

6.2 Testes e Resultados Observados

Durante o desenvolvimento do programa, foram realizados testes simulando diferentes situações eleitorais, variando o número de votos e a chave de criptografia. Essas simulações evidenciaram a funcionalidade do sistema, mostrando que:

Os dados criptografados apresentaram padrões distintos ao serem processados com chaves diferentes, simulando caracteres de alfabetos não latinos.

Em duplicatas de simulações, os dados criptografados se mantiveram iguais, mas com diferentes valores de hash, conforme esperado.

A validação foi bem-sucedida em arquivos intactos e falhou ao detectar alterações manuais nos arquivos, garantindo a integridade dos dados.

Essa estrutura comprova que o programa é capaz de criptografar, armazenar, validar e apurar votos de maneira confiável, respeitando a integridade dos dados e simulando uma urna eletrônica segura.

7 BIBLIOGRAFIA

AMAZON. O que é criptografia? Disponível em: https://aws.amazon.com/pt/what-is/cryptography/. Acesso em: 03 out. 2024.

AMAZON. Chaves assimétricas em AWS KMS Disponível em: https://docs.aws.amazon.com/pt_br/kms/latest/developerguide/symmetric-asymmetric.html. Acesso em: 03 out. 2024.

COMPUTERPHILE. Hashing Algorithms and Security - Computerphile Disponível em: https://www.youtube.com/watch?v=b4b8ktEV4Bg. Acesso em: 03 out. 2024.

GOOGLE. O que é criptografia? Disponível em: https://cloud.google.com/learn/what-is-encryption?hl=pt-BR. Acesso em: 03 out. 2024.

MICROSOFT. O que é autenticação? Disponível em: https://www.microsoft.com/pt-br/security/business/security-101/what-is-authentication. Acesso em: 03 out. 2024.

SIMPLY EXPLAINED. Asymmetric Encryption - Simply explained Disponível em: https://youtu.be/AQDCe585Lnc?si=fqUnvea84QixcYtL. Acesso em: 03 out. 2024.

Stallings, W. (2018). Criptografia e Segurança de Redes: Princípios e Práticas. 7ª ed. São Paulo: Pearson.

Downey, A. B. (2015). Think Python: How to Think Like a Computer Scientist. 2ª ed. O'Reilly Media.

GALVÃO, Michele da Costa (org.). Fundamentos em segurança da informação. São Paulo: Pearson, 2015. *E-book*. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 11 nov. 2024.

Madsen, C. (2018). Python Cryptography. Apress.

Kaufman, C., Perlman, R., & Speciner, M. (2002). Network Security: Private Communication in a Public World. 2ª ed. Prentice Hall.

8 APENDICE

8.1 Código fonte da urna eletrônica

```
import os

# Função para a contagem de votos
def resultado(tiao, ze, branco, nulo, total):
    candidatos = ["Tião do gás", "Zé da feira", "Branco", "Nulo", "Total"]
    contagem = {candidatos[0]: tiao,
                candidatos[1]: ze,
                candidatos[2]: branco,
                candidatos[3]: nulo,
                candidatos[4]: total}
    return contagem

# Função para confirmar voto assim como o BOTÃO VERDE na urna
def confirmacao():
    confirma = input("Digite \"c\" para confirmar o voto: ")
    if confirma == "c":
        print("-=-" * 8)
        print("FIM")
        print("-=-" * 8)
    else:
        print("-=-" * 8)
        print("Voto não confirmado")
        print("-=-" * 8)
    return confirma
```

```python
# Função que simula UM voto
def voto():
    voto_aberto = True
    voto_registrado = ""
    while voto_aberto:
        print("\n" + "-=-" * 8)
        print("Opções de Voto:")
        print("-=-" * 8)
        print("1 - Tião do Gás")
        print("2 - Zé da Feira")
        print("3 - Branco")
        print("-=-" * 8)
        voto_registrado = input("Digite sua opção de voto: ").strip()
        print("-=-" * 8)

        if voto_registrado == '1':
            print("Você votou no Tião do Gás")
            if confirmacao() == "c":
                voto_aberto = False
            else:
                print("Retornando à seleção de candidatos")
        elif voto_registrado == '2':
            print("Você votou no Zé da Feira")
            if confirmacao() == "c":
                voto_aberto = False
            else:
                print("Retornando à seleção de candidatos")
        elif voto_registrado == '3':
            print("VOTO BRANCO")
            if confirmacao() == "c":
                voto_aberto = False
            else:
                print("Retornando à seleção de candidatos")
        else:
            print("VOTO NULO")
            if confirmacao() == "c":
                voto_aberto = False
            else:
                print("Retornando à seleção de candidatos")

    return voto_registrado
```

```python
# Função para criptografar os votos usando XOR
def criptografar_votos(resultados, chave):
    votos_criptografados = ""
    dados_votos = (
        f"Tião do Gás: {resultados['Tião do gás']}, "
        f"Zé da Feira: {resultados['Zé da feira']}, "
        f"Branco: {resultados['Branco']}, "
        f"Nulo: {resultados['Nulo']}, "
        f"Total: {resultados['Total']}"
    )

    for char in dados_votos:
        votos_criptografados += chr(ord(char) ^ chave)
    return votos_criptografados

# Função para gerar um hash usando o PID do processo
def gerar_os_hash(dados, pid):
    hash_valor = 0
    for char in dados:
        hash_valor = (hash_valor + ord(char) + pid) % 256
        hash_valor = (hash_valor << 1) | (hash_valor >> 7)
    return hex(hash_valor)

# Função para salvar os dados criptografados, hash e PID em um arquivo
def salvar_arquivo(votos_criptografados, hash_valor, pid):
    try:
        with open("votos_criptografados.txt", "w", encoding="utf-8") as arquivo:
            arquivo.write(votos_criptografados + "\n")
            arquivo.write(f"Hash: {hash_valor}\n")
            arquivo.write(f"PID: {pid}\n")
        print("Dados criptografados e hash salvos no arquivo 'votos_criptografados.txt'.")
    except Exception as e:
        print(f"Erro ao salvar o arquivo: {e}")
```

```python
# Função principal de votação e criptografia
def funcao_principal():
    seccao_aberta = True
    votos_tiao = votos_ze = votos_branco = votos_nulo = 0

    print("-=-" * 8)
    print("Iniciando Votação")

    while seccao_aberta:
        print("-=-" * 8)
        print("1 - Novo Votante")
        print("2 - Encerrar Secção")
        print("-=-" * 8)
        novo_voto = input("\n")

        if novo_voto == "1":
            voto_individual = voto()
            if voto_individual == '1':
                votos_tiao += 1
            elif voto_individual == '2':
                votos_ze += 1
            elif voto_individual == '3':
                votos_branco += 1
            else:
                votos_nulo += 1
            votos_totais = votos_tiao + votos_ze + votos_branco + votos_nulo

        elif novo_voto == "2":
            print("Secção Encerrada")

            results = resultado(votos_tiao, votos_ze, votos_branco, votos_nulo, votos_totais)
            chave_criptografia = 12345
            votos_criptografados = criptografar_votos(results, chave_criptografia)
            pid = os.getpid()
            hash_valor = gerar_os_hash(votos_criptografados, pid)

            salvar_arquivo(votos_criptografados, hash_valor, pid)
            seccao_aberta = False
```

```python
# Função para descriptografar e validar os votos
def descriptografar_e_validar(arquivo_votos, chave=12345):
    try:
        with open(arquivo_votos, 'r', encoding='utf-8') as file:
            dados_votos = file.readline().strip()
            hash_arquivo = file.readline().replace("Hash: ", "").strip()
            pid_arquivo = int(file.readline().replace("PID: ", "").strip())

        dados_descriptografados = ""
        for char in dados_votos:
            dados_descriptografados += chr(ord(char) ^ chave)

        hash_calculado = gerar_os_hash(dados_votos, pid_arquivo)
        if hash_calculado == hash_arquivo:
            print("Arquivo verificado com sucesso. Hash corresponde.")
            print("Votos descriptografados:", dados_descriptografados)
        else:
            print("Erro: Hash não corresponde. Arquivo pode estar corrompido.")
    except FileNotFoundError:
        print(f"Erro: O arquivo '{arquivo_votos}' não foi encontrado.")
    except Exception as e:
        print(f"Ocorreu um erro: {e}")

# Menu de opções
def main():
    print("Escolha uma opção:")
    print("1 - Iniciar votação")
    print("2 - Descriptografar e validar resultados")
    opcao = input("Opção: ")

    if opcao == "1":
        funcao_principal()
    elif opcao == "2":
        nome_arquivo = input("Digite o nome do arquivo para descriptografar (ex: votos_criptografados.txt): ")
        descriptografar_e_validar(nome_arquivo)
    else:
        print("Opção inválida.")

if __name__ == "__main__":
    main()
```