Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

SEGURANÇA DE PORTÕES ELETRÔNICOS

Leonardo Correia da Silva

Prof. Dr. Fabrício Sérgio de Paula (Orientador)

DOURADOS - MS
2023

SEGURANÇA DE PORTÕES ELETRÔNICOS

Leonardo Correia da Silva

Este exemplar corresponde à redação final da monografia da disciplina Projeto Final de Curso, devidamente corrigida e defendida por Leonardo Correia da Silva e aprovada pela Banca Examinadora, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Dourados, 22 de novembro de 2023.

Prof. Dr. Fabrício Sérgio de Paula (Orientador)

S581s Silva, Leonardo Correia da

Segurança de portões eletrônicos / Leonardo Correia da Silva – Dourados, MS: UEMS, 2023.
42 p.

Trabalho de Conclusão de Curso (Graduação) - Ciência da Computação - Universidade Estadual de Mato Grosso do Sul (UEMS), 2023.

Orientador: Prof.º Dr.º Fabrício Sérgio de Paula

1. Computadores - Medidas de segurança 2. Portões eletrônicos - Segurança 3. Plataforma programável de prototipagem eletrônica - Arduino 4. Criptografia I. Paula, Fabrício Sérgio de. II. Título.

CDD 23 ed. 005.8

Ficha Catalográfica elaborada pela bibliotecária da Universidade Estadual de Mato Grosso do Sul (UEMS) Aline Perdomo Soutelo, inscrita sob o CRB Provisório n° 714 - 1ª Região.

Curso Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

SEGURANÇA DE PORTÕES ELETRÔNICOS

Leonardo Correia da Silva

Novembro de 2023

Banca Examinadora:

Prof. Dr. Fabrício Sérgio de Paula (Orientador)
Área de Computação – UEMS

Profa. [Titulação] Nome do professor
Área de Computação – UEMS

Profa. [Titulação] Nome do professor
Área de Computação – UEMS

AGRADECIMENTOS

Primeiramente gostaria de agradecer a todos que estiveram me apoiando durante todo o decorrer deste estudo. Meus sinceros agradecimentos ao meu orientador e professor, Dr. Fabrício Sérgio de Paula, pela oportunidade, por estar presente e por me acompanhar durante todo o desenvolvimento deste trabalho e também por me orientar em projeto de iniciação científica; agradeço também pela paciência durante esse trabalho e agradecer também ao professor Dr. Nilton Cézar de Paula, por me orientar em um projeto de extensão onde pude desenvolver minhas capacidades de falar em público e agradecer também por ter me orientado em uma monitoria de algoritmos de estrutura de dados I; ao professor MSc. Delair Osvaldo Martinelli Júnior pela oportunidade da monitoria da disciplina de paradigmas de programação.

A minha família, ao meu pai Moacyr Rodrigues da Silva e a minha mãe Marilene Alves Correia, pelo apoio durante a minha formação e durante o desenvolvimento deste trabalho, o quão são importantes para mim, são a base que eu preciso para caminhar e conquistar meus objetivos. Agradeço também aos meus amigos que também estiveram comigo nessa jornada me dando forças.

Muito obrigado a todos!

RESUMO

Este trabalho tem por objetivo estudar conceitos e mecanismos de segurança computacional utilizados para dar suporte ao desenvolvimento de portões eletrônicos seguros. Nesse sentido, é proposto e discutido um protocolo de autenticação baseado em um sistema embarcado feito na plataforma Arduino. No trabalho é feita uma revisão bibliográfica de segurança computacional bem como da plataforma Arduino, abordando suas principais características, componentes a serem utilizados e o funcionamento do circuito eletrônico em questão. O protocolo de autenticação proposto visa mitigar ataques de repetição usando comunicação unidirecional (do controle para o portão) através de radiofrequência. São abordados a viabilidade do protocolo, as dificuldades encontradas, e os resultados experimentais obtidos.

SUMÁRIO

lista de figuras:

1. INTRODUÇÃO

Nos últimos anos, a proliferação de dispositivos conectados à internet e a expansão das redes sem fio tornaram a segurança da informação uma prioridade crítica. Tratando-se de segurança, existem preocupações como: a integridade, isto é, se um dado é autêntico ou falso, ou se ele foi alterado; a confidencialidade, ou seja, se apenas entidades autorizadas têm acesso à informação em questão; e a disponibilidade, ou seja, se o sistema está disponível. Na realidade atual, na qual todos fazemos uso da tecnologia, o não atendimento a essas preocupações pode gerar muitos riscos.

Em um sistema inseguro podem ocorrer situações onde dados falsos são aceitos como verdadeiros, ocasionando problemas diversos como, por exemplo, um banco aceitar um depósito falso, uma fechadura eletrônica ser aberta por um usuário não autorizado. A preocupação com segurança deve ser ainda maior quando se lida com a comunicação sem fio, algo extremamente comum na atualidade.

Para garantir um nível desejado da segurança computacional são empregados recursos chamados de mecanismos de segurança. Para garantir um nível desejado da segurança computacional são empregados recursos chamados de mecanismos. Um deles é a criptografia onde a mensagem a ser enviada é embaralhada, desse jeito mesmo que algo importante esteja circulando em um meio não confiável, caso alguma entidade não autorizada tente ler essa mensagem, não será possível entendê-la, atendendo assim o requisito de confidencialidade [1].

Ataques de repetição, nos quais uma mensagem é interceptada entre duas entidades e retransmitida posteriormente pela entidade atacante quando conveniente [10], têm surgido em um contexto de comunicação não confiável, representando uma ameaça significativa. Esses ataques, que exploram vulnerabilidades em sistemas de segurança, apresentam sérias implicações para a integridade e a confidencialidade das informações transmitidas. Dessa forma, representam uma vulnerabilidade potencial em sistemas e comunicações que dependem de autenticação e integridade de dados.

Esse tipo de vulnerabilidade pode ser observado em portões eletrônicos que utilizam autenticação através da comunicação sem fio. O usuário tem um dispositivo emissor que envia um sinal que faz o portão abrir. Nessa situação, um ataque de repetição que burle a autenticação faria um indivíduo não autorizado ter livre acesso ao portão. Recentemente esse

tipo de ataque foi aplicado por criminosas que furtaram lojas de dois *shopping centers* de Campo Grande-MS, com prejuízo estimado em R$ 211.000,00 [2].

O objetivo deste trabalho é estudar segurança computacional visando aplicar conceitos e mecanismos para melhorar a segurança de portões eletrônicos controlados por radiofrequência. Para a implementação foi estudada e utilizada a plataforma Arduino[5], que por suas características de projeto permite uma prototipação facilitada. Dessa forma, foi possível não apenas estudar mas prototipar uma solução para portões eletrônicos seguros.

Este trabalho está organizado da seguinte forma:

- O Capítulo 2 revisa conceitos de segurança computacional;
- O Capítulo 3 descreve a plataforma Arduino;
- O Capítulo 4 apresenta uma solução para melhorar a segurança de portões eletrônicos.
- Por fim, é feita a conclusão deste trabalho.

2. SEGURANÇA COMPUTACIONAL

A segurança computacional é essencial para viabilizar a disponibilização de novas tecnologias. Em algumas situações, há a necessidade de ocultar informações sensíveis. Em outras, deseja-se que dados armazenados ou trafegados não sejam indevidamente alterados. Além disso, é essencial que os serviços providos através dessas novas tecnologias estejam sempre disponíveis aos usuários pretendidos. Em termos formais, a segurança computacional pode ser alcançada através do cumprimento de três requisitos: confidencialidade, integridade e disponibilidade [1], onde a interpretação desses requisitos varia de acordo com o contexto de aplicação.

2.1 Confidencialidade

A confidencialidade visa a ocultação de informação ou recursos para os indivíduos não autorizados [2]. A necessidade de manter a informação em segredo surge no uso de tecnologias em áreas sensíveis, como no governo ou indústrias que retém informações sigilosas. Por exemplo, é comum empresas ocultarem o conteúdo de seus projetos em desenvolvimento, uma vez que empresas concorrentes podem tirar proveito desses projetos sem o alto investimento necessário para desenvolvê-los.

A criptografia é um mecanismo de segurança que permite implementar a confidencialidade [1]. Esse mecanismo consiste em codificar os dados de tal forma que a mensagem se torne compreensível apenas para os usuários que possuam acesso à chave de decifragem desses dados. Para os demais usuários, que não possuem acesso à chave, os dados tornam-se incompreensíveis. Há outras formas de impedir o acesso não autorizado aos dados sem usar criptografia, como a combinação de um mecanismo de autenticação baseado em usuário/senha combinado com um mecanismo de prevenção de acesso para leitura/escrita associado ao usuário autenticado [1]. Nesse caso, os dados são armazenados sem nenhuma codificação, porém o sistema limita o acesso apenas aos usuários autorizados.

2.2 Integridade

A integridade é o requisito de segurança para a prevenção de mudanças impróprias ou não autorizadas em dados ou sistemas [1]. Em outras palavras, os dados são íntegros enquanto permanecerem no estado que o último usuário autorizado os deixou [2]. Caso os

dados sejam alterados por um usuário não autorizado, eles deixam de ser íntegros. O conceito de integridade envolve a integridade dos dados, relacionada ao conteúdo da informação, e a integridade da origem, relacionada à fonte da informação(também chamada de autenticação) [1]. A fonte da informação pode afetar a precisão, credibilidade e a confiança que as pessoas depositam nessa informação, sendo, portanto, fundamental para o bom funcionamento de um sistema [1].

Mecanismos de integridade podem ser divididos em duas classes: mecanismos de prevenção e mecanismos de detecção [1]. Os mecanismos de prevenção têm por objetivo manter a integridade dos dados, bloqueando qualquer tentativa na qual um usuário tenta alterar dados para os quais ele não tem permissão para isso. Os mecanismos de detecção monitoram o sistema em busca de identificar situações onde há violações da integridade. Nessas situações de violação, simplesmente relatam que a integridade dos dados não é mais confiável. Esses mecanismos podem analisar eventos do sistema (ações do usuário ou do próprio sistema) para identificar as violações, ou também podem analisar os próprios dados para verificar se as restrições necessárias ou esperadas ainda estão sendo cumpridas.

2.3 Disponibilidade

A disponibilidade está relacionada à capacidade de usar as informações ou recursos desejados [1]. Um recurso ou informação está disponível quando é acessível por um usuário autorizado em um formato conveniente e em um tempo razoável [2]. A disponibilidade é um aspecto importante no projeto de sistemas porque um sistema indisponível é pelo menos tão ruim quanto a ausência desse sistema.

No cumprimento desse requisito de segurança é essencial considerar situações onde alguém pode deliberadamente planejar negar o acesso a dados ou serviços de forma a torná-los indisponíveis [1]. Para lidar com essas situações, os projetos de sistemas costumam construir modelos estatísticos para modelar os padrões esperados de uso dos recursos computacionais. Com base nesses modelos, é possível verificar se a disponibilidade está sendo ou não garantida. No entanto, é necessário se preocupar com a possibilidade dos modelos estatísticos serem manipulados a ponto de não retratarem a realidade do uso legítimo e, por consequência, falharem [1].

2.4 Conceitos relacionados à segurança computacional

Diversos conceitos surgem no estudo e prática da segurança computacional, sendo que inicialmente serão apresentados os conceitos de vulnerabilidades, ameaças e contramedidas. A vulnerabilidade trata de um ponto do sistema suscetível a um ataque [2]. Uma ameaça é uma potencial violação da segurança, nem sempre intencional [1]. A violação não precisa ter ocorrido de fato para que haja uma ameaça. O fato de que a violação possa ocorrer significa que o sistema deve ser protegido contra as ações que poderiam causar essa violação. Um ataque ocorre quando há, de fato, uma violação intencional da segurança explorando uma ou mais vulnerabilidades no sistema [1]. Aqueles que executam as ações de ataques são chamados de atacantes.

Os três requisitos de segurança—confidencialidade, integridade e disponibilidade—se contrapõem às ameaças à segurança de um sistema, que podem ser divididas em quatro classes [1]: disclosure, ou acesso não autorizado a informações; decepção, ou aceitação de dados falsos; interrupção ou prevenção da operação correta; e usurpação, ou controle não autorizado de alguma parte de um sistema. Essas classes possuem uma grande abrangência e podem englobar diversas ameaças em comum.

Por exemplo, um *snooping* que envolve a intercepção não autorizada de informação, é uma forma de *disclosure*. É uma atividade passiva e implica em alguma entidade estar "ouvindo" uma comunicação ou informações do sistema. Uma variação dessa ameaça é o *wiretapping*, onde uma rede é monitorada. Para lidar com essas ameaças devem ser empregados mecanismos para garantir que a confidencialidade seja garantida e, portanto, se alcance a segurança desejada [1].

As contramedidas são as técnicas usadas para proteger o sistema das possíveis ameaças [2]. Quando se trata de contramedidas, é comum a presença de dois termos: políticas de segurança e mecanismos. As políticas de segurança são as declarações do que é ou não permitido em um sistema [1]. Já os mecanismos são os métodos, ferramentas ou processos que irão reforçar e dar suporte às políticas de segurança [1].

As políticas exigem frequentemente alguns mecanismos processuais que a tecnologia não consegue impor. Por exemplo, suponha que o laboratório de computação de uma universidade tenha uma política que proíba qualquer aluno de copiar os arquivos de outro aluno. O sistema desse laboratório deve, portanto, prover mecanismos para impedir que outras pessoas leiam os arquivos de um usuário. Uma forma comum de implementar esses mecanismos é delegar aos usuários a responsabilidade de proteger seus arquivos contra leitura

de terceiros. No entanto, essa responsabilidade nem sempre é cumprida, seja por falta de conscientização de todos os usuários, descuido ou até mesmo intencionalmente.

Quando uma instituição define nas políticas de segurança o que é uma ação "segura" e "não segura", os mecanismos de segurança podem prevenir, detectar ou até mesmo se recuperar de um ataque [1].

A prevenção visa frustrar as tentativas de ataque. Normalmente, a prevenção envolve a implementação de mecanismos que os usuários não podem ignorar e que são confiáveis para serem implementados de maneira correta e inalterável. Alguns mecanismos preventivos podem ser complicados e interferir na utilização do sistema como por exemplo quando uma entidade falha repetidamente em se autenticar, bloquear futuras tentativas por um tempo ou até que o usuário se comunique com o administrador do sistema. Mecanismos preventivos simples, como a autenticação através de usuário/senha, que visam impedir o acesso de usuários não autorizados a sistemas, tornaram-se amplamente aceitos.

A detecção serve para ataques que não podem ser evitados ou indicar a eficácia de medidas preventivas. Nesses mecanismos se aceita que o ataque irá acontecer e o objetivo é identificar quando o ataque está em andamento ou se o ataque já ocorreu e relatá-lo.

Já a recuperação pode ter duas formas. A primeira é parar um ataque e avaliar e reparar quaisquer danos causados por esse ataque. A segunda consiste garantir que o sistema continue funcionando corretamente mesmo quando um ataque está em curso, o que é bastante difícil de implementar devido à complexidade dos sistemas[1].

2.5 Criptografia

A criptografia é um importante mecanismo para garantia do requisito da confidencialidade. Esse mecanismo faz uso de algoritmos de cifragem e chaves secretas para transformar dados inteligíveis (dados sem cifragem) em ininteligíveis (dados cifrados). Os algoritmos cifragem devem ser reversíveis, de forma que os dados cifrados possam ser restaurados para a sua forma original antes da cifragem [10].

Dentre os algoritmos de criptografia temos os algoritmos de chave simétrica onde e de chave usada para cifragem e decifragem dos dados é a mesma e deve ser mantida em sigilo e passada para a outra entidade participante da comunicação por um meio sigiloso[10]. E a pública ou assimétrica onde as chaves são diferentes, desse modo podendo comunicar uma delas em um meio não sigiloso[1].

O componente básico da criptografia é o sistema criptográfico. Por definição, um sistema criptográfico é uma 5-tupla (E, D, M, K, C), onde M é um conjunto de texto, K é um conjunto de chaves, C é um conjunto de texto cifrado, E: M x K → C, é o conjunto de funções de cifragem, e D: C x K →M representa o conjunto de funções de decifragem [1].

De uma maneira mais simples podemos dizer que ao aplicar no algoritmo o texto junto com a chave de cifragem, se obtém o texto cifrado. E ao aplicar no algoritmo o texto cifrado junto a chave de decifragem, o texto cifrado volta ao formato original. Dessa maneira, apenas pessoas autorizadas(que têm a posse da chave) podem entender o texto.

De modo geral, existem três tipos de ataques contra a criptografia [1]. No primeiro tipo, o atacante possui apenas o texto cifrado e seu objetivo é encontrar o texto original. Se possível, ele também pode tentar encontrar a chave. No segundo tipo, o atacante possui o texto cifrado e o texto simples que foi cifrado e seu objetivo é encontrar a chave que foi usada. No terceiro tipo, o atacante solicita que textos específicos sejam criptografados e, com base nos textos cifrados correspondentes, busca encontrar a chave que foi usada. Esses ataques usam tanto matemática como estatística.

2.5.1 Criptografia simétrica

A criptografia simétrica, ou de chave privada, pode ser realizada através de cifras de transposição ou cifras de substituição [1].

A cifragem de transposição reorganiza os caracteres do texto para formar o texto cifrado. As letras não são alteradas. Sendo assim o principal componente para essa cifragem é uma função de permutação. Como a permutação não altera a frequência dos caracteres do texto simples, uma cifra de transposição pode ser detectada comparando as frequências dos caracteres com um modelo da linguagem. Atacar uma cifra de transposição requer o rearranjo das letras do texto cifrado. Este processo, denominado anagrama, utiliza tabelas de frequências de n-gramas para identificar n-gramas comuns.

Já a cifragem de substituição altera caracteres no texto simples para produzir o texto cifrado. Imagine um caso de cifragem utilizando uma chave 3.Nesse caso seria alterando cada letra no texto simples, mapeando-a na letra três caracteres posteriores no alfabeto (e circulando de volta ao início do alfabeto, se necessário). Já o ataque para essa cifragem se baseia em estatística, como por exemplo, levar em consideração a frequência dos caracteres com a frequência que as letras do idioma do texto são usados para tentar reverter o

mapeamento. Porém quanto maior a chave utilizada, mais difícil fica utilizar a probabilidade para reverter o mapeamento.

Com base nessas técnicas mais básicas surgiu O *Data Encryption Standard* (DES). Esse algoritmo é orientado a bits. Nele se usa transposição e substituição e, por esse motivo, às vezes é chamada de cifra de produto. Sua entrada, saída e chave têm 64 bits cada. Os conjuntos de 64 bits são chamados de blocos. Essa técnica consiste em 16 rodadas, ou iterações. Cada rodada usa *round key* separada de 48 bits. Essas *round keys* são geradas a partir do bloco de chave eliminando os bits de paridade (reduzindo o tamanho efetivo da chave para 56 bits), permutando os bits e extraindo 48 bits. Um conjunto diferente de 48 bits é extraído para cada uma das 16 rodadas. Se a ordem em que as *round keys* são usadas for invertida, a entrada será decifrada. [1]

Com o avanço das pesquisas a respeito da criptografia, surgiu também o *Advanced Encryption Standard* (AES). Esse também é um algoritmo de cifragem em bloco, ou seja, ele divide o texto a ser cifrado em blocos e cifra bloco por bloco utilizando um conjunto de chaves. A unidade básica considerada para computação é um byte. O algoritmo irá dividir o texto em blocos de 128-bits. Já a chave pode ter três comprimentos diferentes:128-bits, 192-bits ou 256-bits. Independentemente do comprimento de chave utilizada, será gerada uma subchave de 128-bits (*round key)*. O AES também pode executar um número diferente de rodadas de operação. Para gerar um número suficiente de chaves de rodada, o AES expande as chaves de criptografia dependendo do número de rodadas e do comprimento da chave de criptografia especificada pelos usuários. O tamanho inicial da chave apenas altera a quantidade de rodadas utilizadas, onde cada rodada utiliza uma *round keys* diferentes.

O algoritmo transforma os 128-bits do texto em uma sequência de bytes, dessa maneira formando 16-bytes e os representando como uma matriz 4x4. Iremos nos referir essa como uma matriz de estado. A criptografia AES executa em cada rodada (exceto na última rodada) a mesma sequência de operações simples em matrizes de estado que transforma o bloco de texto simples em um bloco de texto cifrado. Essas operações são bytes substitutos, linhas de deslocamento, colunas mistas e *add-round-key* .

A operação de bytes substitutos é uma operação não linear baseada em uma caixa de substituição especialmente projetada. O objetivo desta operação é resistir a ataques baseados em estatística. Enquanto a operação das linhas de deslocamento é uma operação elementar em matrizes de estado. É uma operação linear. O objetivo desta operação é produzir difusão. A operação de colunas mistas também é uma operação elementar em matrizes de estado. Sua finalidade é a mesma das linhas de deslocamento. Já a operação de *add-round-key*

é um conjunto simples de operações OR exclusivas em matrizes de estado. É uma operação linear. O objetivo desta operação é causar confusão ao atacante. [3]

2.5.2 Criptografia assimétrica

Os algoritmos de criptografia assimétrica também também são chamados de algoritmos de chave pública, uma vez que uma dessas chaves pode ser exposta (chave pública). No entanto, ter conhecimento dessa chave não leva ao conhecimento da outra chave (chave privada). Como uma chave é pública e sua chave complementar deve permanecer secreta, um criptosistema de chave pública deve atender às três condições a seguir [1]:

1. Deve ser computacionalmente fácil cifrar ou decifrar uma mensagem com a chave apropriada.
2. Deve ser computacionalmente inviável derivar a chave privada da pública chave.
3. Deve ser computacionalmente inviável determinar a chave privada de um ataque de texto simples escolhido.

Esse tipo de criptografia faz uso de funções de uma direção com *trapdoor*, nelas podemos dizer que, é fácil calcular f(x), porém é "difícil" ou inviável calcular x partindo de f(x), porém partindo de uma trapdoor, que seria um segredo, é fácil computar x partindo de f(x) junto ao segredo. No caso da criptografia pública, a chave pública seria utilizada para o cálculo de f(x), ou seja, todos podem criptografar os dados, mas o caminho inverso, ou seja, descobrir x, é "difícil", mas partindo da chave privada(trapdoor) é possível fazer a decifragem da mensagem de maneira fácil[9].

2.5.3 Autenticação

Sendo a autenticação o ato de uma entidade externa fornecer informações que permitam ao sistema confirmar a sua identidade. Como por exemplo algo que a entidade sabe, tem posse, faz parte dela, ou até mesmo sua localização[1].Já a autenticação de dados tem dois propósitos: Certificar a origem dos dados e convencer o usuário de que os dados não foram modificados ou fabricados. Existem algoritmos que envolvem criptografia que auxiliam esses objetivos, os algoritmos permitem resolver problemas como:

- Ataque por repetição: alguma entidade grava as comunicações legítimas entre 2 outras entidades, e depois usa parte da gravação para o seu proveito. Se a parte da gravação usada não é alterada este ataque é chamado passivo; caso contrário, é chamado ativo.
- Personificação: alguma entidade simula um outro usuário legítimo. Por exemplo, toma o lugar de uma entidade 'A' sem que a outra entidade que está na comunicação note qualquer diferença.
- Um mal-intencionado lê e altera parte da informação transitando na linha de comunicação antes de chegar ao em seu destino que é a entidade 'B', ou seja, 'B' gostaria de detectar se alguma alteração parcial foi feita na linha — isto é chamado de detecção de integridade de informação[10].

Como exemplo temos o Protocolo de identificação Feige, Fiat e Shamir, onde a entidade A, envia as informações que comprovam sua identidade sem revelar qual informação secreta é.Inicialmente há um preparativo de dois parâmetros: n e s da seguinte forma:

1. Uma entidade confiável seleciona dois primos relativamente longos p, q. Então define n = pq mas mantém em segredo os valores p, q.
2. Cada pessoa usuária deste protocolo, 'A', escolhe um inteiro s primo a n tal que $1 \leq s \leq n - 1$ e calcula v = s^2 mod n e registra v na entidade confiável como sendo a chave pública de 'A'.

Feitas essas escolhas, o protocolo consiste em iterar t vezes um protocolo de quatro passos:

1. 'A' escolhe aleatoriamente r :$1 \leq r \leq n - 1$ envia para 'B' $x = r^2$ mod n. x é chamado testemunho. Note que r não é conhecido por 'B', pois é computacionalmente difícil calcular a raiz quadrada de x mod n sem conhecer a fatoração de n.
2. 'B' escolhe aleatoriamente um bit $e = 0$ ou $e = 1$ com probabilidade 1/2 e envia e para 'A'.Vamos chamar e de desafio.
3. 'A' calcula e envia para 'B' uma resposta y tal que $y = r$ *se*, caso $e = 0$, ou $y = rs$ mod n. Se $e = 1$. Note que neste último caso 'A' não revela s para 'B', mas apenas rs mod n e Beto desconhece r.
4. 'B' rejeita a resposta se $y = 0$ ou senão aceita se $y^2 = xv^e$ mod n. Note que ao rejeitar $y = 0$ 'B' está bloqueando o caso $r = 0$.Note também que $y^2 = r^2(s)^2$ mod $n = r^2ve$ mod $n = xv^e$ mod n.

Note que sem o desafio, como por exemplo e sendo sempre 1, 'B' receberia só $y = rs$ nas t iterações. Uma falsa Alice, digamos Alicia, que gravasse todos os pares ($x = r^2$,$y = rs$) válidos

poderia personificar ‘A’ fornecendo esses pares para ‘B’ e ele aceitaria, pois são validados através de $y^2 = xv = r^2s^2$.Portanto, e = 0 aleatoriamente é necessário[10]

3. PLATAFORMA ARDUINO

Arduino é uma plataforma eletrônica de código aberto baseada em *hardware* e *software* fáceis de usar [5]. Ela foi inicialmente criada para uma prototipagem fácil e rápida voltada para alunos sem formação em eletrônica e programação. O fato de ser uma plataforma de código aberto permitiu desenvolver uma grande comunidade em torno desta plataforma, que fornece conhecimento acessível que auxilia grande parte dessa comunidade formada tanto por novatos quanto por especialistas[5].

O software Arduino IDE usado para programação das placas é fácil de usar para iniciantes, mas flexível para atender usuários avançados. Também é possível de ser utilizado em sistemas operacionais como Windows, Linux e MacOS. No mundo acadêmico utilizam-no para construir instrumentos científicos de baixo custo, para provar princípios de química e física ou para iniciar o estudo de programação e robótica. Designers e arquitetos constroem protótipos interativos, músicos e artistas utilizam-nos para instalações e para experimentar novos instrumentos musicais [5].

O software Arduino IDE é uma plataforma de código aberto, o que significa que está acessível para ser aprimorado por programadores experientes. A linguagem pode ser estendida por meio de bibliotecas C++, e as pessoas que desejam entender os detalhes técnicos podem dar o salto do Arduino para a linguagem de programação AVR C na qual ela se baseia. O Hardware é extensível e de código aberto, ou seja, os planos/projetos das placas Arduino são publicados sob uma licença Creative Commons [5], para que projetistas de circuitos experientes possam fazer sua própria versão de módulos, ampliando-os e melhorando-os.

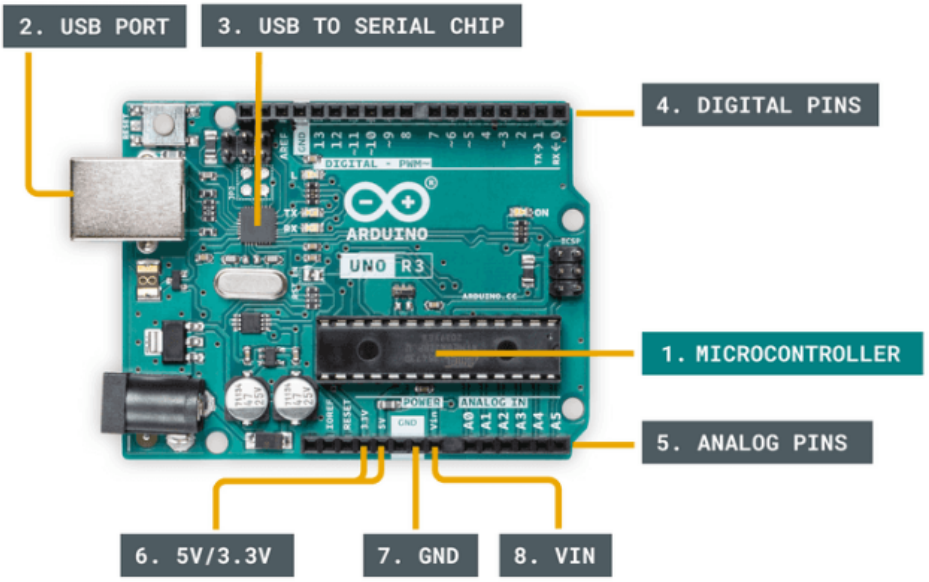

Figura 1 - Anatomia do arduino Uno R3. Fonte: [6]

3.1 Hardware Arduino

Para a descrição do funcionamento de uma placa Arduino, será utilizado como base a placa Uno R3 representada na Figura 1. Nessa representação temos:

1. O microcontrolador: o componente que carrega o programa e faz o processamento necessário que foi programado no programa carregado;
2. porta USB: usada para conectar o Arduino com o computador;
3. Chip Serial para o USB: auxilia na tradução dos dados fornecidos, por exemplo, de um computador para o microcontrolador integrado. É esse chip que permite programar a placa Arduino a partir do computador;
4. Pinos digitais: neles é utilizada lógica digital (0,1 ou LOW/HIGH). Esses pinos normalmente são usados para interruptores ou para operações como ligar/desligar um LED;
5. Pinos analógicos:podem ler valores analógicos em uma resolução de 10 bits (0-1023);
6. Pinos 5V / 3,3V:usados para alimentar componentes externos;
7. GND: também conhecido como ground, negativo ou simplesmente ‘-’, é usado para completar um circuito, onde o nível elétrico está em 0 volt;
8. VIN:significa Voltage In, onde podem ser conectadas fontes de alimentação externas.

A memória do Arduino "padrão" normalmente é constituída de duas memórias: SRAM e memória Flash. A SRAM (Static Random-Access Memory) é usada para, por

exemplo, armazenar o valor de uma variável. Quando desligado, esta memória é reiniciada. A memória Flash é usada principalmente para armazenar o programa principal ou as instruções do microcontrolador. Esta memória não é apagada ao ser desligada, de modo que as instruções do microcontrolador são executadas assim que a placa é energizada.

A quantidade de memória disponível em um Arduino varia de placa para placa. Por exemplo, o Arduino UNO possui flash de 32kB/SRAM de 2kB, enquanto um Nano 33 IoT possui flash de 256kB/SRAM de 32kB. As características presentes nas diversas placas Arduino podem ser observadas na Tabela 1.

	UNO	MEGA 2560	LEONARDO	DUE	ADK	NANO	PRO MINI	ESPLORA
Microcontrolador	ATmega328	ATmega2560	ATmega32u4	AT91SAM3X8E	ATmega2560	ATmega168 (versão 2.x) ou ATmega328 (versão3.x)	ATmega168	ATmega32u4
Portas digitais	14	54	20	54	54	14	14	-
Portas PWM	6	15	7	12	15	6	6	-
Portas analógicas	6	16	12	12	16	8	8	-
Memória	32K (0,5K usado pelo bootloader)	256K (8K usado pelo bootloader)	32K (4K usado pelo bootloader)	512K disponível para aplicações	256K (8K usado pelo bootloader)	16K (ATmega168) ou 32K (ATmega328) (bootloader: 2K)	16K (2K usado pelo bootloader)	32K (4K usado pelo bootloader)
Clock	16Mhz	16Mhz	16Mhz	84Mhz	16Mhz	16Mhz	8Mhz (modelo 3.3v) ou 16Mhz (modelo 5v)	16Mhz
Conexão	USB	USB	Micro USB	Micro USB	USB	USB Mini-B	Serial/Módulo USB externo	Micro USB
Conector para alimentação externa	Sim	Sim	Sim	Sim	Sim	Não	Não	Não
Tensão de operação	5V	5V	5V	3.3V	5V	5V	3.3 ou 5V, dependendo do modelo	5V
Corrente máxima portas E/S	40mA	40mA	40mA	130mA	40mA	40mA	40mA	-
Alimentação	7-12Vdc	7-12Vdc	7-12Vdc	7-12Vdc	7-12Vdc	7-12Vdc	3.35-12V (modelo 3.3v) ou 5-12V (modelo 5v)	5V

Tabela 1. Comparação entre placas Arduino. Fonte[13]

3.2 Funcionamento da placa

A maioria das placas Arduino são projetadas para ter um único programa rodando no microcontrolador. Este programa pode ser projetado para executar uma única ação, como piscar um LED. Também pode ser projetado para executar centenas de ações em um ciclo.

O programa carregado no microcontrolador iniciará a execução assim que for ligado. Todo programa possui as funções “setup()” e “loop()” conforme a Figura 2. A função setup()

é executada uma vez ao ligar a placa, é onde é feita a inicialização e definição de componentes. A função loop() é onde se pode implementar o algoritmo a ser processado repetidamente, que pode ser, por exemplo, ler um sensor, acender uma luz, e verificar se uma determinada condição foi atendida.

Figura 2. Estrutura de um programa básico Arduino. Fonte: Autor

A velocidade de um programa depende do tamanho do programa e de quanto tempo o microcontrolador leva para executar suas instruções, mas geralmente é em microssegundos [6].

A construção de um circuito consiste em pelo menos um componente eletrônico ativo e um material condutor, como fios, para que a corrente possa passar. Um exemplo simples de circuito é um circuito de LED . Um fio é conectado de um pino no Arduino(para o caso do LED, um pino digital) a um LED por meio de um resistor (para proteger o LED de alta corrente) e, finalmente, ao pino de aterramento (GND). Quando o pino está definido para o estado HIGH , o microcontrolador na placa Arduino permitirá que uma corrente elétrica flua através do circuito, o que acende o LED. Quando o pino é colocado no estado LOW , o LED se apaga, pois a corrente elétrica não está fluindo pelo circuito. Esse exemplo está

representado na Figura 3. E um código de exemplo seria a Figura 4 onde o led é intercalado a cada 1 segundo entre ligado e desligado.

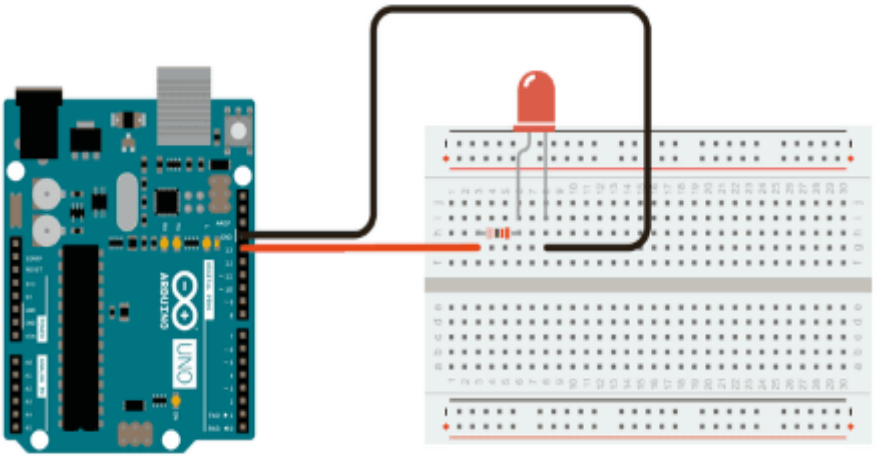

Figura 3.Um circuito de LED com um Arduino.

File Edit Sketch Tools Help

Arduino Uno

sketch_oct16a.ino

```
1   // Definir o número do pino do LED
2   const int ledPin = 13;
3
4   void setup() {
5     // Configurar o pino do LED como saída
6     pinMode(ledPin, OUTPUT);
7   }
8   void loop() {
9     // Ligar o LED
10    digitalWrite(ledPin, HIGH);
11    // Aguardar 1 segundo
12    delay(1000);
13    // Desligar o LED
14    digitalWrite(ledPin, LOW);
15    // Aguardar mais 1 segundo
16    delay(1000);
17  }
```

Figura 4. Código de exemplo do uso de um LED. Fonte: Autor

Toda a comunicação entre componentes eletrônicos é facilitada por sinais eletrônicos. Existem dois tipos principais de sinais eletrônicos: analógicos e digitais.

Um sinal analógico geralmente está vinculado a um intervalo. Em um Arduino, esse intervalo é normalmente de 0 a 5 V ou 0 a 3,3 V. No programa, isso é representado no intervalo de 0 a 1023, que é uma resolução de 10 bits [6]. Se escrevermos um sinal analógico usando modulação por largura de pulso (PWM) [11], podemos usar uma faixa entre 0-255, pois estamos usando uma resolução de 8 bits [6].

Já um sinal digital é representado apenas por estados binários (0 ou 1) que são lidos como estados altos ou baixos no programa. Este é o tipo de sinal mais comum na tecnologia moderna.Você pode ler e escrever facilmente sinais digitais em um Arduino. Essa técnica é eficiente para ler estados de botões ou para ligar ou desligar algo [6]. Eles também podem ser utilizados para criar uma sequência, enviando rapidamente um estado alto ou baixo várias vezes.

Ao trabalhar com Arduino é comum serem utilizados sensores e atuadores. Os sensores são componentes utilizados para detectar o seu ambiente e converter em sinal elétrico. Já os atuadores são componentes usados para atuar ou alterar um estado físico, ou seja, converte sinal elétrico em energia, como, por exemplo, de um motor que produz energia mecânica. Sensores e atuadores são normalmente chamados de entradas e saídas [6]. Quando escrevemos um programa, é comum construir estruturas condicionais que verificam o estado de um sensor e decidem se ele deve acionar algo [6]. Ou seja, um sinal de entrada seria quando se verifica o estado de um componente, já um sinal de saída seria o que ocorre ao acionar um componente.

No desenvolvimento de programas para o Arduino são comumente utilizadas bibliotecas de *software*. Essas bibliotecas são uma extensão da API padrão do Arduino, tanto oficiais quanto contribuídas pela comunidade. As bibliotecas simplificam o uso de códigos complexos, como a leitura de um sensor específico, o controle de um motor ou a conexão com a Internet. Em vez de ter que escrever todo esse código sozinho, é possível simplesmente instalar uma biblioteca, incluí-la no topo do seu código e usar qualquer uma das funcionalidades disponíveis. Todas as bibliotecas do Arduino são de código aberto e gratuitas para uso por qualquer pessoa.

Dentre os diversos componentes externos que podem ser utilizados em um circuito, destaca-se o módulo de radiofrequência [7], que é empregado neste trabalho. Existem diversos módulos tanto para frequências baixas como 433mhz, e outros com frequências mais

altas, cada uma com seus lados positivos e negativos. O módulo que trata do 433mhz, é unidirecional, ou seja, é necessário um receptor e um transmissor para a transmissão das mensagens digitais. O tipo de modulação da portadora de Rádio frequência é o ASK (*amplitude shift keying*) – modulação por chaveamento de amplitude [7]. Isto é, quando existe o Bit 1 a portadora transmite o sinal de 433 MHz. Quando o Bit é zero, nenhum sinal é transmitido. Sendo assim, o sinal fornecido ao transmissor é um sinal digital.

Para um perfeito funcionamento dos módulos de RF 433 MHz, é necessário a instalação das antenas tanto no transmissor quanto no receptor, desse modo fazendo o módulo ter um bom alcance. Com a frequência de 433,92 MHz e velocidade da onda eletromagnética no espaço de 3 x 108 m/s , o comprimento de onda é :

$$\lambda = v/f = 300.000.000 / 433.920.000 = 0,69 \text{ metros}$$

Usando uma antena com 1/4 do comprimento de onda teríamos:

$$D = 0,69 \text{ m} / 4 = 17,28 \text{ cm}$$

Portanto a antena deve ter um tamanho de aproximadamente 17,3 cm para um bom aproveitamento do alcance[7].

4. PORTÃO ELETRÔNICO SEGURO

4.1 Circuito

Neste trabalho foi implementado um circuito para representação de um sistema que mitigar ataques de repetição para portões eletrônicos. Para isso foi utilizado um módulo de rádio frequência (RF) 433mhz, 2 placas de Arduino Uno r3, uma com o receptor e outra com o transmissor.

Por se tratar de um protótipo, foi apenas utilizado um LED, sendo ele o LED embutido na própria placa Arduino, para simbolizar os estados do portão de uma maneira mais abstrata, sendo o LED aceso como o portão aberto e o LED desligado para simbolizar o portão fechado. O enfoque do trabalho foi a comunicação sem fio e a implementação do protocolo. Por ser utilizado apenas um módulo RF (1 receptor + 1 transmissor) o projeto foi desenvolvido de maneira HALF-Duplex, ou seja, a comunicação é unidirecional, deixando inviável boa parte dos protocolos já existentes. Por esse motivo foi criado um protocolo adaptado para esse protótipo. O protótipo pode ser visualizado na Figura 5.

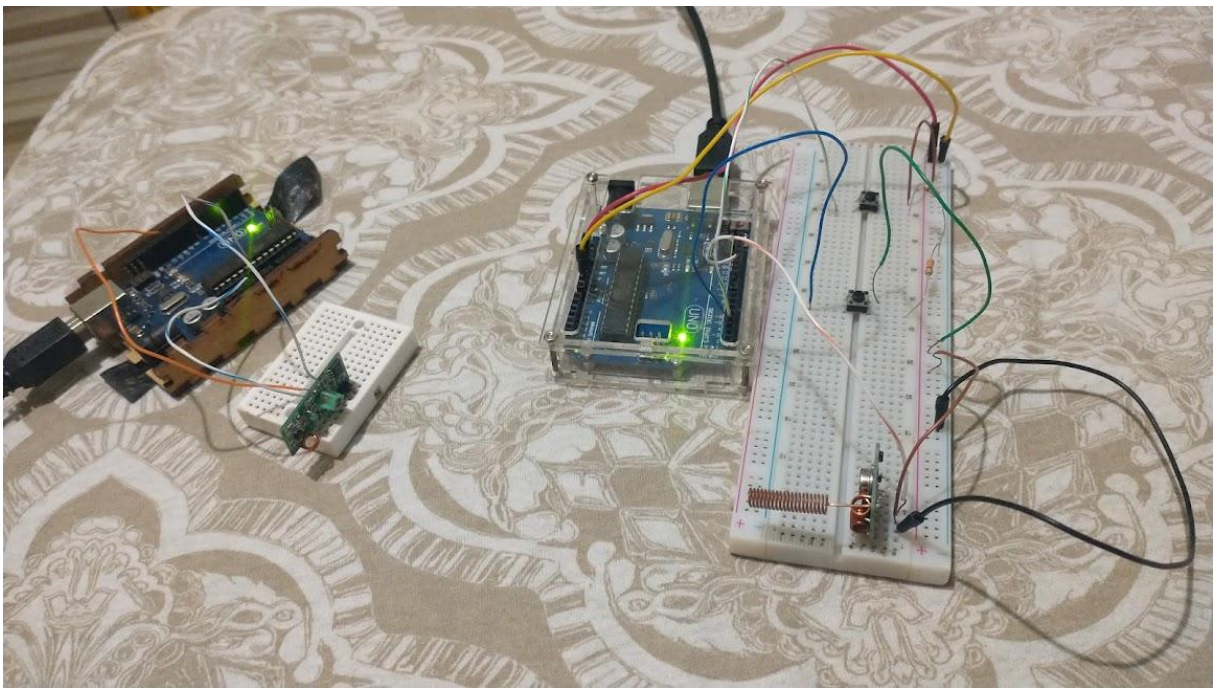

Figura 5. Protótipo do trabalho. Fonte: Autor

Na Figura 6 estão presentes dois botões, sendo um deles designado para abrir o portão (acionando o LED) e o outro para fechar o portão (desligando o LED). Cada botão requer três conexões distintas: uma para a transmissão do estado, uma para conexão de aterramento e uma terceira para fornecimento de tensão de 5V. Além disso, o transmissor é equipado com três pinos correspondentes, dispostos nessa sequência: um para a transmissão de dados, um para a corrente de 5V e um terceiro para conexão à terra. É importante mencionar que, por padrão, o pino de dados está conectado ao pino digital 12 devido à biblioteca de comunicação [14] utilizada. De modo geral, ao pressionar o botão será selecionado o respectivo comando para montar a mensagem junto com o valor do contador incrementado e o número do controle, essa mensagem é criptografada e enviada pelo transmissor.

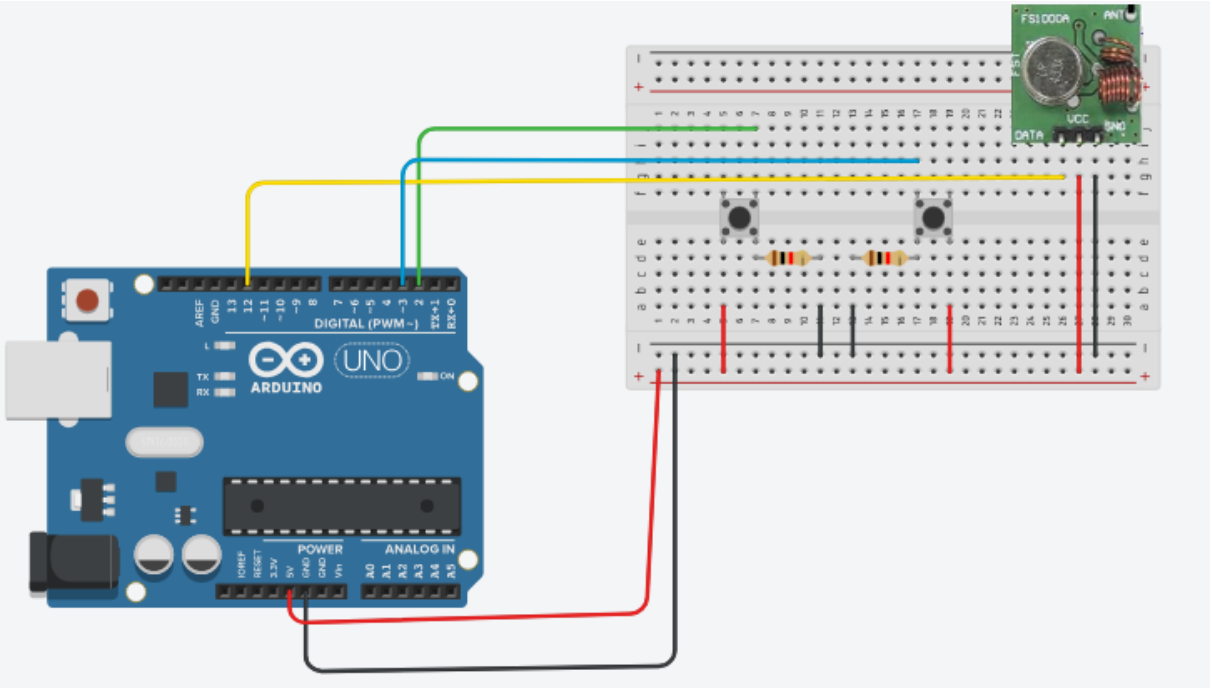

Figura 6. Diagrama do transmissor. Fonte: Autor

Na Figura 7, encontra-se o diagrama do receptor, o qual está alimentado por uma corrente de 5V fornecida pela placa Arduino, juntamente com uma conexão terra, estabelecida pelos pinos mais à esquerda e à direita do receptor. Os dois pinos internos são dedicados à transmissão de dados provenientes do receptor, sendo onde o cabo amarelo se conecta no diagrama, se pode ser utilizado qualquer um dos dois pinos de dados. O pino de dados é conectado pelo cabo amarelo ao o pino digital 11 da placa Arduino, conforme estabelecido pela biblioteca de transmissão utilizada[14], que, por padrão, utiliza esse pino para o processamento dos dados recebidos. Esses dados ao serem recebidos, ocorre um o

processamento da mensagem, que inclui a decodificação do Base64 e a descriptografia da mensagem usando o AES128, seguido de uma análise das divisões da mensagem. Primeiramente, verifica-se se os primeiros 16 bytes correspondem a algum dos códigos de abertura ou fechamento do portão. Caso haja correspondência, a mensagem é considerada válida. Posteriormente, é realizado um segundo controle, no qual se verifica a validade do número de sequência. Esse controle visa mitigar possíveis ataques de repetição, fazendo uso do número do portão para a validação do número de sequência correto. Uma vez aprovada na segunda validação, a ação de abertura ou fechamento do portão, ou seja, o acionamento ou desligamento do LED, é executada.

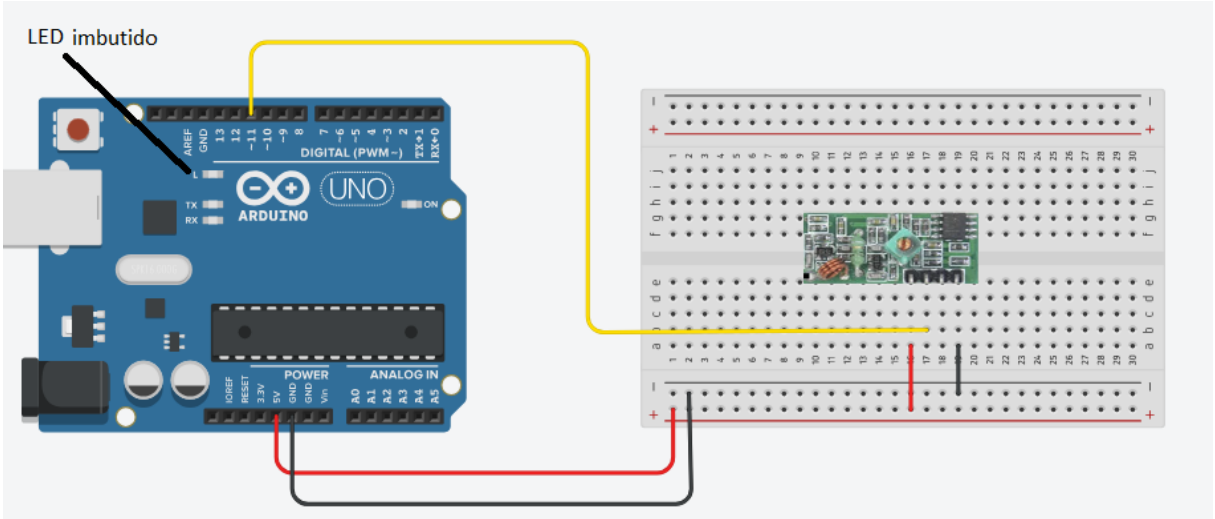

Figura 7. Diagrama do receptor. Fonte: Autor

4.2 Protocolo de autenticação

O protocolo desenvolvido para este trabalho é composto por dois comandos distintos que podem ser executados. Um deles é destinado à abertura do portão, enquanto o outro é utilizado para o fechamento do portão. A representação desse comando se assemelha a uma senha que está armazenada tanto no Arduino transmissor (controle remoto) quanto no Arduino receptor (portão).

Contudo, mesmo que essa senha seja enviada de forma cifrada, surge a possibilidade de ataques de repetição. Portanto, é necessário concatenar a essa senha mais alguma informação variável para garantir que as mensagens cifradas sejam distintas a cada utilização

válida do portão, dificultando assim a descoberta dessa senha/comando. Nesse sentido, a transmissão do comando para o portão incluirá um contador.

Para que uma solicitação seja validada no portão, o contador deve apresentar um valor superior ao último valor recebido, com um intervalo de 32 unidades entre eles. Esse espaço de 32 números é destinado a acomodar possíveis interferências, que podem ser causadas tanto por perturbações no sinal quanto por obstáculos físicos, como uma parede, que podem dificultar a chegada do sinal até o portão. Também deve-se considerar que há situações acidentes onde o botão do controle é pressionado acidentalmente.

Também será necessário ter na mensagem enviada um número que represente a identidade do controle, permitindo que exista mais de um controle. Ou seja, o contador base no Arduino receptor terá um arranjo com o último valor válido recebido de cada portão.

Já a criptografia é feita por meio de uma chave simétrica com o algoritmo AES com chave de 128bits de uma biblioteca chamada crypto criada para o Arduino[12]. As chaves já serão implementadas em todos componentes Arduino, transmissores e receptor, de maneira fixa. Com isso não existe a necessidade de troca de chaves, que é algo inviável devido a comunicação unidirecional.

A mensagem em questão trabalhada tem 32 bytes. Os primeiros 16 bytes representam o código único que representa o comando a ser executado, como por exemplo o comando “20i5cPUewFJ3msGe”. Os demais 16 bytes envolvem o número de sequência(contador) que precisa estar no intervalo de 32 números desde a última mensagem válida e o número do controle, a separação do comando, número de sequência e número do portão na mensagem é feito pelo caráter “/”. Vendo a possibilidade de a junção do número de sequência com o número do portão não ser o suficiente para atingir os 16bytes. Foi implementada a geração de ‘0’s à esquerda do número do controle para preenchimento da mensagem de modo que alcance os 16 bytes necessários para cifrar a segunda parte da mensagem, um exemplo da mensagem montada pode ser visto na Figura 8.

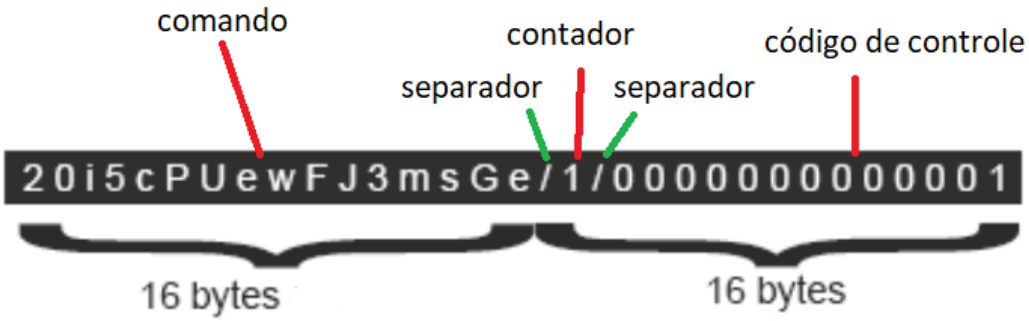

Figura 8. Mensagem montada utilizada. Fonte: Autor.

Como o algoritmo usado é o AES 128bits, para a cifragem da mensagem é necessário chamar o algoritmo de cifragem duas vezes, uma para os primeiros 16 bytes e outra vez para os 16 bytes restantes.

Após a cifragem de dados para o envio, pode ocorrer de ser gerado bytes de dados que representam caracteres especiais não imprimíveis ou não ASCII. Em sistemas de comunicação ou protocolos que não tratam esses caracteres de maneira adequada, eles podem ser corrompidos ou interpretados de forma incorreta durante a transmissão. Ainda mais por caracteres especiais poderem ter a representação por mais de 1byte, para evitar problemas relacionados a isso uma boa prática é utilizar Base64[8], transformando os bytes criptografados em caracteres seguros. Esses caracteres em base64 será a mensagem a ser transmitida. Um exemplo de mensagem criptografada e codificada em base64 pode ser visto na Figura 9.

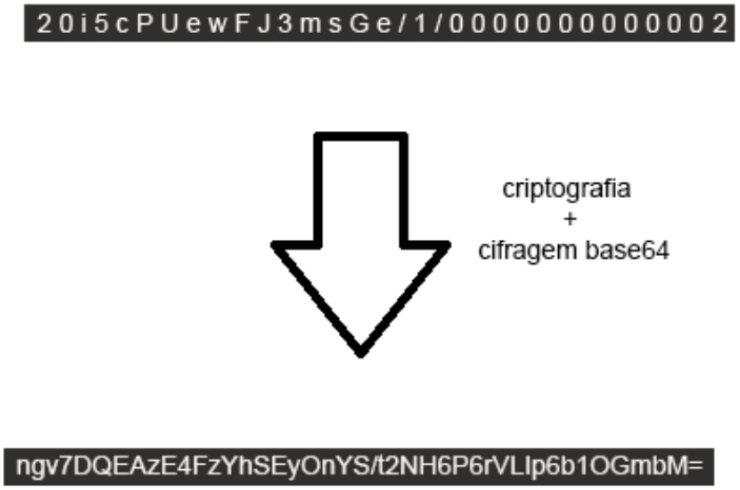

Figura 9. Cifragem de uma mensagem. Fonte: Autor.

4.3 Testes e resultados obtidos

No experimento, o tempo para montar a mensagem, que seria uma concatenação de *strings* junto a um código de conversão de inteiro para string, somado com o tempo de efetuar a cifragem foi de aproximadamente 3ms. Separando para analisar apenas o tempo de cifragem por AES, temos apenas 1ms, o que condiz com a documentação. E nos testes, o tempo para montagem da mensagem utilizando operações de concatenação e conversão de valores

numéricos em texto junto ao preenchimento de mensagem, demora aproximadamente 2ms. Deste modo, somando 3ms.

Mesmo contando com o tempo utilizado para o codificado em base64, por esse tempo ser muito baixo, ainda se tem o tempo de 3ms. No receptor o tempo de decifragem junto a decodificação em base 64 deu 3ms. Analisando apenas a decifragem em do AES temos aproximadamente 2ms o que condiz com a documentação. A distância de teste foi 1 metro, isso ocorre para o RF 433mhz adquirido, a pinagem para acoplar a antena era pequena o que gerou problemas para administrar o peso da antena com a solda. Com isso, uma solução a curto prazo foi utilizar uma antena menor de aproximadamente 2 cm.

Com isso o resultado de alcance não atendeu a expectativa de pelo menos 10 metros para se ter um de uso viável. Porém com um investimento maior isso pode ser conquistado visto que o problema encontrado se dá pelo tamanho da antena utilizado. Com um módulo de radiofrequência mais robusto ou com um investimento maior em equipamentos de solda é possível aumentar esse alcance baseado na teoria, pois a teoria diz que a antena deveria ter 17cm, porém neste trabalho foi utilizado uma antena de apenas 2 cm.

Já o tempo de envio não foi calculado por ser muito pequeno para se capturar com um meio externo, e como os Arduinos não tem conectividade com um relógio global, não é possível calcular a diferença de tempo de envio e recebimento da mensagem.

5. CONCLUSÃO

Com esse trabalho foi possível desenvolver um sistema de mitigação para os ataques de repetição com o mecanismo de criptografia que auxilia o requisito de confidencialidade, ou seja, foi obtido um protocolo que atende ao que foi proposto. E ele se difere por ser um protocolo que se adequa a um ambiente mais limitado. Pois a comunicação utilizada é unidirecional, onde na maioria das fontes encontradas os protocolos faziam uso de uma comunicação de ida e volta. Além de fazer uso de um Arduino UNO que apesar de sua simplicidade, é inferior em questão de desempenho a placas outras placas mais robustas. Também foi possível durante esse trabalho se aprofundar em conhecimentos de segurança, criptografia, e desenvolvimento de um software embarcado usando como plataforma o Arduino.

Infelizmente foram encontrados problemas como a questão do alcance e a aparente necessidade de uma antena de 17cm, tornando um controle pelo menos 17cm inviável para ser portátil. É interessante para trabalhos futuros melhorar o alcance e incrementar o trabalho para uma maior versatilidade como por exemplo encontrar um tamanho de antena que não seja demasiado grande e tenha um alcance satisfatório.

Um outro exemplo de melhoria seria o uso de um relógio global para cálculo de tempo de envio de mensagem, ou até mesmo o uso do relógio global incorporado à mensagem para evitar ataques de repetição ao invés do uso de um contador.

Também é possível verificar se o aumento no custo que for gerado para essas melhorias é significativo. Ou seja, se incrementar o projeto para que se tenha uma comunicação bidirecional, e alterar o algoritmo para um protocolo visando esse tipo de comunicação, se é obtida uma diferença no nível de segurança que vale o gasto gerado, sendo esse gasto, o tempo de comunicação junto com o gasto adicional de componentes.

REFERÊNCIAS BIBLIOGRÁFICAS

[1]Matt Bishop - Introduction to computer security-Addison-Wesley Professional (2004)

[2]Rick-Lehtinen; G.T.-Gangemi-Sr. - Computer Security Basics - 2006

[3]Jie Wang - Computer Network Security. Theory and Practice (2009, Springer)

[4]noticia sobre ataque envolvendo rede sem fio, disponivel em: https://noticias.uol.com.br/cotidiano/ultimas-noticias/2023/08/01/roubo-de-codigo-e-prejuizo-de-r-211-mil-como-agem-as-irmas-criminosas.htm acesso em 15/08/2023 as 14:53.

[5]https://docs.arduino.cc/learn/starting-guide/whats-arduino acesso em 27/09/2023 as 09:57.

[6]https://docs.arduino.cc/learn/starting-guide/getting-started-arduino#anatomy-of-an-arduino-board acesso em 29/09/2023 as 10:40

[7]https://blog.eletrogate.com/guia-basico-dos-modulos-tx-rx-rf-433mhz/ acesso em 29/09/2023 as 10:50

[8]RFC 3548: The Base16, Base32, and Base64 Data Encodings, disponivel em : https://datatracker.ietf.org/doc/html/rfc3548, acesso em 03/10/2023 as 8:44

[9]Bruce Schneier - Applied Cryptography: Protocols, Algorithms, and Source Code in C

[10]Routo Terada - Segurança de dados

[11]https://docs.arduino.cc/learn/microcontrollers/analog-output acesso em 27/09/2023 as 10:55.

[12]https://rweather.github.io/arduinolibs/crypto.html 29/09/2023 as 10:57

[13]https://www.electrofun.pt/blog/qual-arduino-comprar-conheca-os-tipos-de-arduino/ 29/09/2023 as 13:34

[14]https://www.airspayce.com/mikem/arduino/RadioHead/ 29/09/2023 as 14:20