MELHORANDO A SEGURANÇA CIBERNÉTICA: USO DO ARDUINO PARA CRIPTOGRAFIA COMO UM MIDDLEWARE DO TECLADO

Augusto Savi[1], Giácomo Antônio Althoff Bolan [2]

Resumo: A segurança da informação é crucial para garantir a proteção das informações contra ameaças. Com o crescimento das Tecnologias da Informação e Comunicação (TICs), surgem desafios adicionais para manter a segurança. Neste contexto, o objetivo deste artigo foi desenvolver um middleware utilizando Arduino para criptografar a comunicação entre o teclado do usuário e o computador em comunicadores instantâneos. Foram realizados experimentos que demonstraram a eficácia da criptografia na proteção dos dados sensíveis durante a transmissão. Embora tenham sido identificados alguns desafios, o middleware trouxe resultados satisfatórios e potencial para melhorar a segurança na comunicação ponta a ponta.

Palavras-chave: Criptografia. Arduino. Middleware. Malware.

ABSTRACT: Information security is crucial to ensure the protection of information against threats. With the growth of Information and Communication Technologies (ICTs), additional challenges arise to maintain security. In this context, the objective of this article was to develop a middleware using Arduino to encrypt the communication between the user's keyboard and the computer in instant messengers. Experiments were conducted that demonstrated the effectiveness of encryption in protecting sensitive data during transmission. Although some challenges were identified, the middleware yielded satisfactory results and has the potential to improve end-to-end communication security.

Keywords: Cryptography. Arduino. Middleware. Malware.

1 INTRODUÇÃO

A Segurança da Informação visa garantir a disponibilidade, integridade, confidencialidade e autenticidade das informações (BRASIL, 2019). Quando as informações são adulteradas, não estão disponíveis quando necessárias ou estão sob o controle de indivíduos mal-intencionados, a organização e civis podem ser expostos a prejuízos consideráveis.

Com a abrangência das Tecnologias da Informação e Comunicação (TICs), é encontrado comunicadores instantâneos gratuitos que permitem a troca de mensagens de texto através de computadores conectados à internet.

[1] guto_savi@unesc.net
[2] kinhobolan@gmail.com

Internet que por definição do Oxford Learner's Dictionaries (2023) é uma internacional de computadores conectando outras redes e computadores que permite que as pessoas compartilhem informações em todo o mundo.

Ao mesmo tempo, os desafios de segurança também aumentam, pois estão disponíveis várias ferramentas de software que ajudam hackers a atacar computadores facilmente sem muito conhecimento da área de informática (PACHGHARE, 2019), fazendo com que a maioria dos comunicadores instantâneos aplicassem criptografia no tráfego das mensagens visando assegurar sua segurança.

Um exemplo de uso de criptografia nos comunicadores instantâneos é o uso do Signal Protocol pelo aplicativo de mensagens Whatsapp que usa a End-to-End Encryption (E2EE), para encriptar as mensagens e permitir com que apenas as pontas consigam acessar as mensagens, escondendo-as de qualquer indivíduo que poderia monitorar toda a comunicação entre as pontas (RASTOGI, 2017).

Porém, o malware, software malicioso com alta propagação, tem como objetivo roubar informações, corromper arquivos ou apenas fazer atividades maliciosas para incomodar os usuários e sempre vêm com novas ideias para dificultar a sua detecção por antivírus (BERGERO, 2001) são capazes de contornar a criptografia de ponta a ponta (E2EE) sem comprometer sua integridade.

Um exemplo de malware que consegue burlar o E2EE é o Keylogger, uma ferramenta com finalidade de gravar todas as teclas digitadas no teclado do computador e as envia para a pessoa que está realizando o ataque (R. VENKATESH, R. K. SEKHAR, 2015).

Então todas as mensagens enviadas, e-mails, senhas digitadas através do teclado serão roubadas sem o usuário do computador perceber (SINGH, 2021).

Com isso, mesmo que o meio de comunicação esteja criptografado, o fato do computador estar infectado por um malware quebra todo o princípio da segurança da informação.

Sabendo que os teclados ainda permanecem como o meio de entrada mais popular para inserir grandes quantidades de textos sem erros (ZHANG, YAN, NARAYANAN, 2017) esta pesquisa, teve como objetivo geral desenvolver uma solução para aumentar a segurança na comunicação ponta a ponta feita através de comunicadores instantâneos, utilizando o Arduino como meio de encriptação entre o teclado do usuário e o computador. Os objetivos específicos desta pesquisa são os

seguintes: a) Compreender o conceito de segurança da informação; b) Estudar criptografia; c) Investigar comunicação ponto a ponto e d) Explorar interfaces de comunicação com o computador. Sua metodologia se realizou através de testes para validar a efetividade da solução desenvolvida. Por fim, analisou-se os resultados obtidos com os testes realizados.

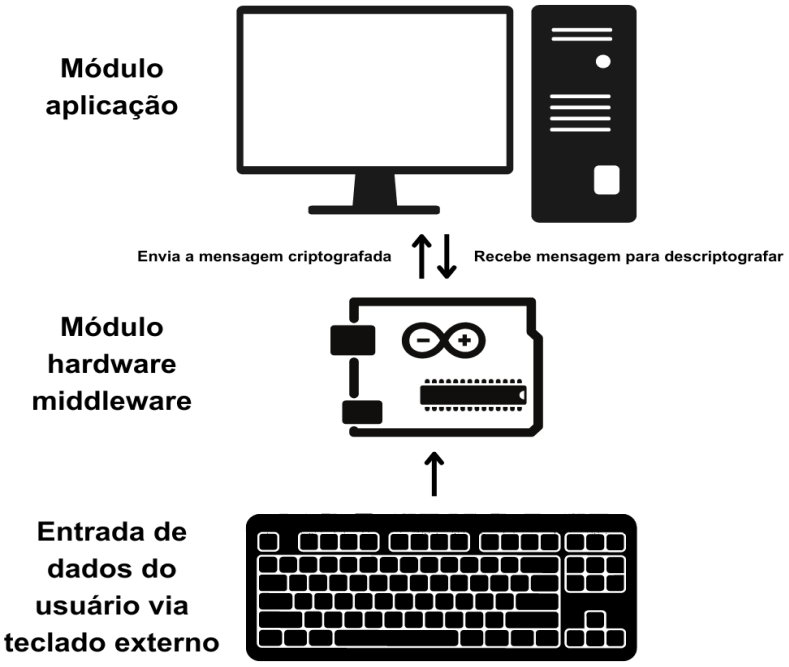
2 MATERIAIS E MÉTODOS

Este artigo se trata de uma pesquisa aplicada e de base tecnológica. Desenvolveu-se o protótipo de um *middleware* capaz de criptografar mensagens enviadas através de comunicadores instantâneos. O protótipo consiste em dois módulos, sendo eles: *hardware middleware* e aplicação, como demonstrado através da Figura 1.

O módulo de aplicação consistiu no desenvolvimento de um programa que gera a seleção dos modos do *hardware middleware*, sendo eles: criptografia, descriptografia e normal, além de receber e enviar mensagens para descriptografia e demonstração das mesmas utilizando-se da comunicação serial.

O módulo *hardware middleware* consiste em receber e gerenciar os dados recebidos tanto pelo usuário através do teclado externo e os da aplicação.

Figura 1 - Representação gráfica da arquitetura do protótipo



Fonte: Autores (2023).

2.1 Módulo aplicação

A aplicação foi desenvolvida utilizando a linguagem de programação Python, selecionada não apenas por sua versatilidade, mas também pela sua compatibilidade com o ambiente de desenvolvimento integrado (IDE) Visual Studio Code. O sistema operacional escolhido para o processo de construção foi o Ubuntu, reconhecido pela sua confiabilidade e aderência na comunidade de desenvolvimento.

A biblioteca pySerial foi utilizada para estabelecer uma comunicação serial eficaz com o hardware. A pySerial é notória por sua rica funcionalidade encapsulada, oferecendo documentação extensa que simplificou significativamente a integração e comunicação com o hardware.

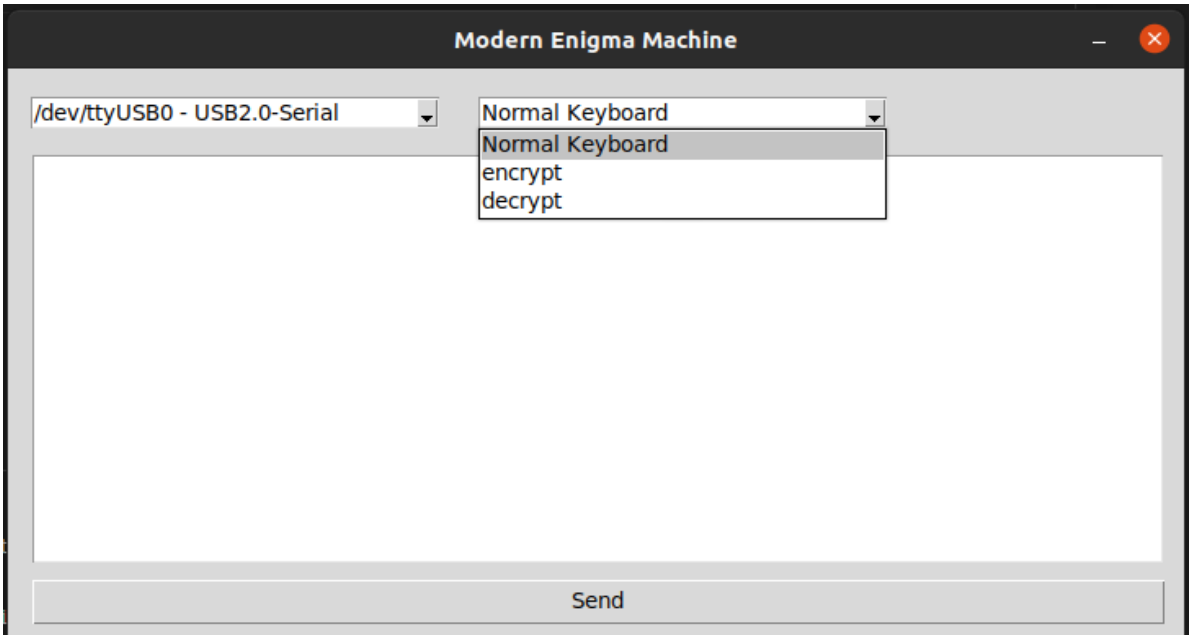
A fim de proporcionar uma interface de usuário funcional, que permite a seleção dos modos de operação do hardware - nomeadamente *'Normal Keyboard'*, *'encrypt'* e *'decrypt'* - a seleção da porta para conectar o Arduino, e a inserção da mensagem para decifrar em uma caixa de texto, foi adotado o pacote Tkinter. Este pacote é a interface padrão do Python para a toolkit GUI Tcl/Tk. Foram incorporadas as bibliotecas PyAutoGUI, pystray, Pillow e multiprocessing, que em conjunto permitiram a construção de uma aplicação assíncrona. Esta aplicação é capaz de enviar e receber mensagens em tempo real, sem causar bloqueios.

Quando o usuário altera o modo de operação, a aplicação envia uma única mensagem com o prefixo *"[MODE]"*, seguido do modo selecionado, para o hardware via comunicação serial.

Para enviar uma mensagem para ser decifrada pelo hardware, o usuário simplesmente insere a mensagem na caixa de texto fornecida pela aplicação e clicar no botão *"Send"*.

Quando uma *string* é recebida via porta serial, o software verifica se a *string* contém comandos especiais ou texto. Se for um comando especial, a tecla correspondente é pressionada no computador usando a biblioteca *pyautogui*. Se for texto, o texto é digitado no computador. Isso permite que o Arduino envie comandos de teclado para o programa *Python* em execução no computador. A Figura 2, para melhor compreensão, apresenta como funciona a interface do usuário da aplicação.

Figura 2 - Interface do usuário da aplicação



Fonte: Autores (2023).

2.2 Módulo middleware

O desenvolvimento do *hardware middleware* utilizou a placa de prototipagem Arduino Uno na versão SMD como base. Esta placa é caracterizada por seu microcontrolador ATmega328P, tendo uma Memória Flash de 32KB (dos quais 0,5KB são destinados ao Bootloader). Ademais, apresenta uma SRAM de 2KB, uma EEPROM de 1KB e uma velocidade de Clock de 16MHz.

Além disso, a corrente máxima suportada por cada pino de entrada/saída (I/O) do ATmega328P é de 40 mA (miliamperes), de acordo com o *datasheet* do microcontrolador (ATMEL, 2016). Isso significa que qualquer pino I/O individual não deve ser solicitado a fornecer ou afundar mais de 40 mA, como também o *datasheet* especifica que o total de corrente fornecido ou afundado por todos os pinos I/O combinados não deve exceder 200 mA.

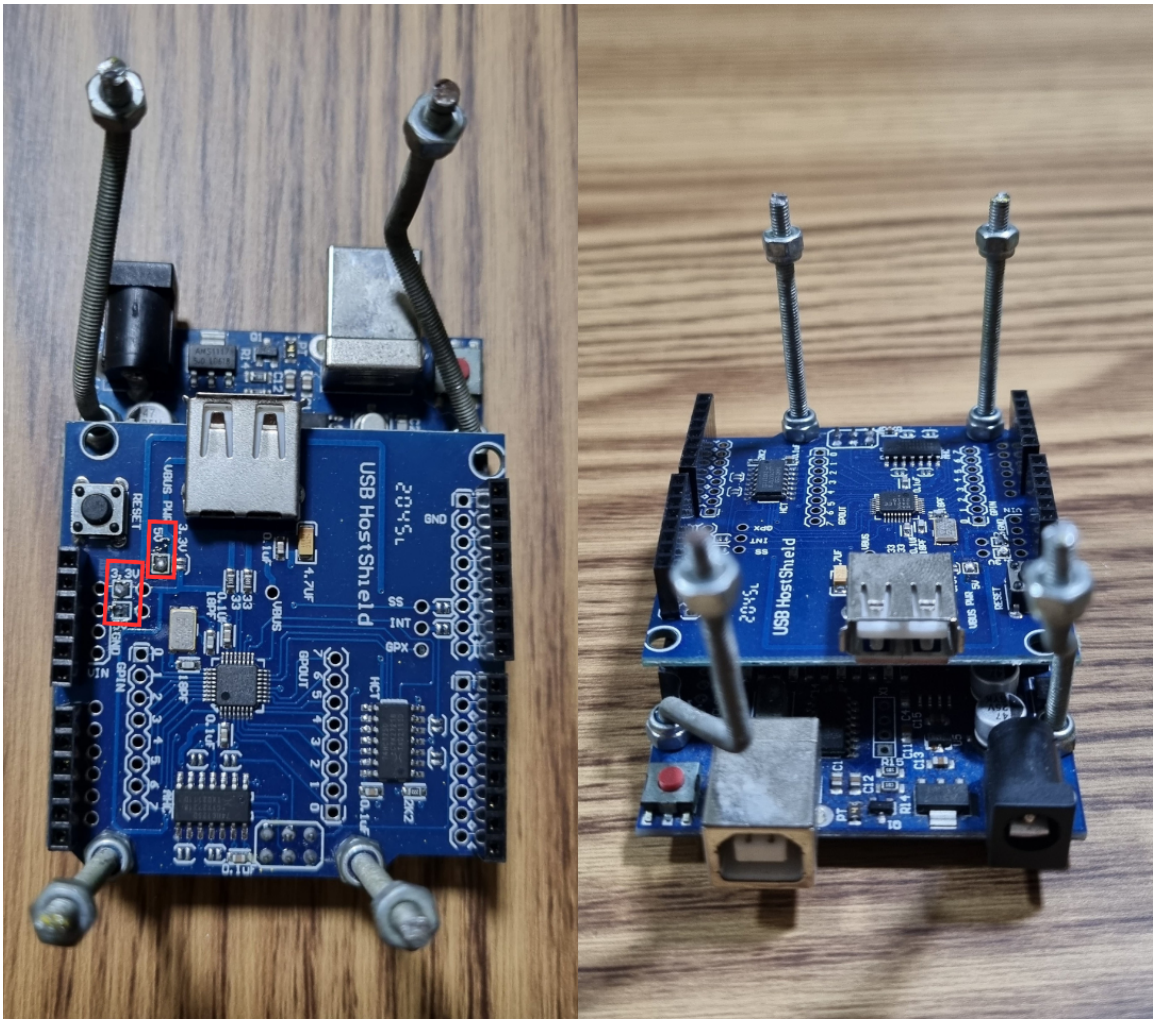
Para adaptar essa capacidade, optou-se pelo uso de uma *shield* USB ADK versão 2.0. Este componente não apenas simplifica a conexão com periféricos externos, mas também dispõe de um conector USB 2.0 A Fêmea, capaz de fornecer até 500 mA. Dessa forma, conseguimos compatibilidade com a maioria dos teclados disponíveis no mercado. No decorrer deste projeto, empregamos o teclado com fio USB Logitech K120, que se conecta via tecnologia USB-A e consome 5v e 100mA de corrente.

Por fim, com o intuito de facilitar a visualização das mensagens criptografadas e descriptografadas, optamos por utilizar o display LCD 1602 da marca QA. Este componente, equipado com o módulo I2C MH, dispõe de 16 colunas por 2 linhas de caracteres alfanuméricos.

2.2.1 Montagem do módulo middleware

Durante o desenvolvimento do *hardware do middleware*, foi necessário implementar algumas etapas para garantir o funcionamento correto do USB *Host Shield*. Uma dessas etapas envolveu a soldagem dos *jumpers* de 3V e 5V, essenciais para o funcionamento adequado do dispositivo. Após a soldagem, posicionamos o USB *Host Shield* sobre o Arduino, como demonstrado na Figura 3.

Figura 3 - Pontos de soldagem no USB Host Shield



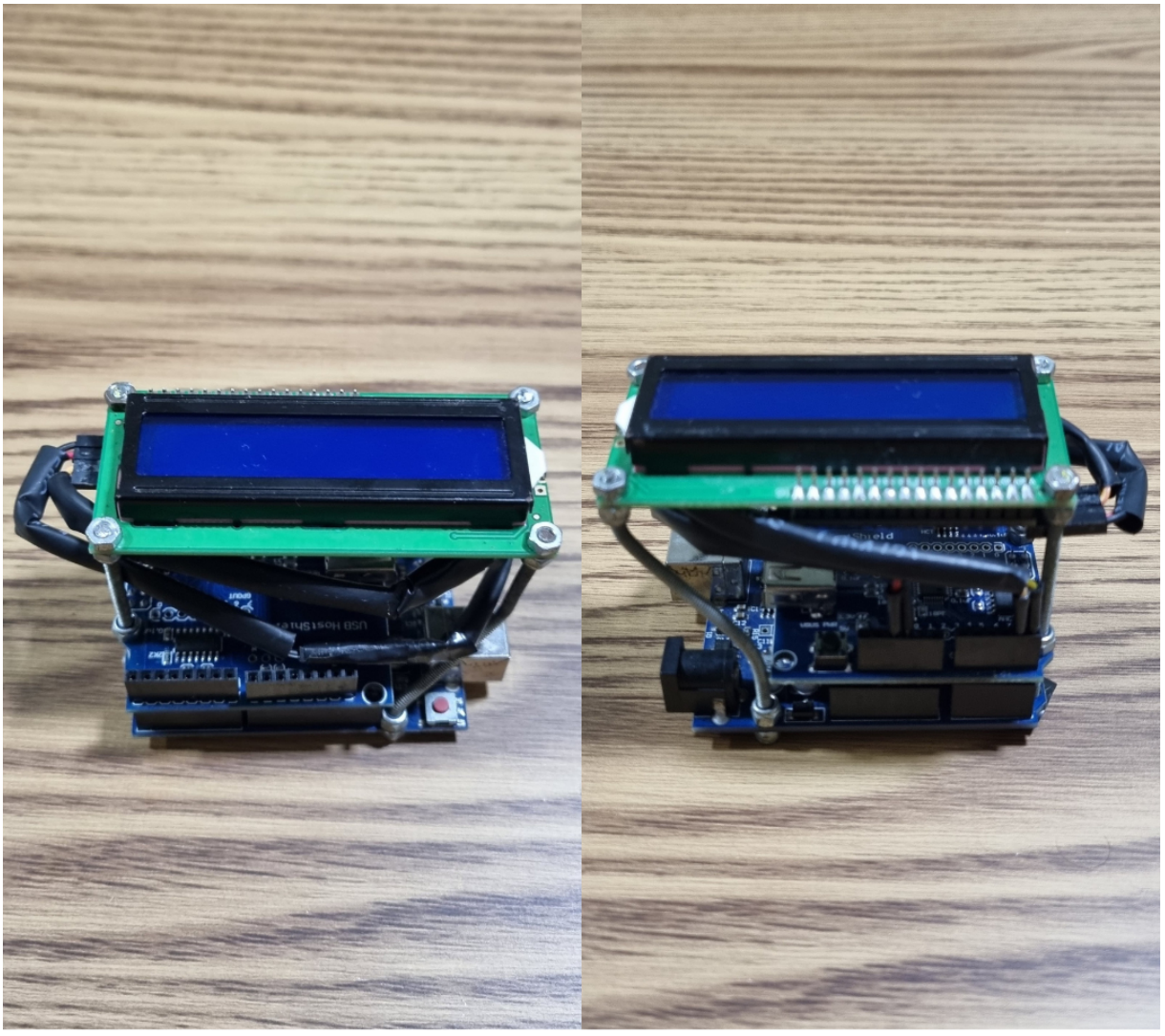
Fonte: Autores (2023).

Para a exibição das mensagens, foi optado pela utilização de um display LCD 1602, o qual foi conectado ao módulo I2C (*Inter-Integrated Circuit*). Para permitir a comunicação com o Arduino, foi feita a conexão dos pinos SDA e SCL do módulo I2C aos pinos A4 e A5 do USB *Host Shield*.

Para garantir a alimentação adequada do LCD, conectamos os pinos VCC e GND do módulo I2C aos respectivos pinos VCC e GND do USB *Host Shield*. Essa medida assegurou que a energia fosse adequadamente fornecida ao display, possibilitando a exibição clara e contínua das mensagens.

Os componentes foram fixados com 4 pedaços de barras com rosca e porcas autotravantes, oferecendo uma abordagem com uma conexão sólida, como demonstrado na Figura 4.

Figura 4 - Módulo middleware montado



Fonte: Autores (2023).

2.2.2 Seleção da criptografia

A escolha da criptografia ROT47, para este projeto de criptografia no Arduino, foi realizada ao levar em consideração a limitação de memória do dispositivo e a necessidade de demonstrar a eficácia do *middleware*, em vez de implementar uma criptografia robusta e segura.

O Arduino é uma plataforma de *hardware* de recursos limitados, incluindo capacidade de memória. Em dispositivos com recursos limitados, como é o caso do Arduino, é importante otimizar o uso de memória para garantir que o programa se encaixe na capacidade disponível. Algoritmos de criptografia mais complexos, como AES e RSA, exigem uma quantidade significativa de memória para implementação.

Nesse contexto, a criptografia ROT47 foi escolhida por ser uma técnica simples e de baixo consumo de recursos. Ela é uma variação do cifra de substituição ROT13, na qual os caracteres ASCII imprimíveis são rotacionados em 47 posições. Essa técnica é fácil de ser implementada e não requer tabelas de pesquisa complexas ou operações matemáticas intensivas. Na Figura 5 é possível compreender como foi realizada a implementação do ROT47 utilizando a linguagem do Arduino.

Figura 5 - Implementação do ROT47 na linguagem Arduino utilizando tabela de mapeamento

```
String rot47_decode(String input)
{
  String table = "`!\"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_abcdefghijklmnopqrstuvwxyz{|}~";
  String output = "";

  for (char& _char : input) {
    int index = table.indexOf(_char);
    if (index != -1) { // Se o caractere estiver na tabela
      output += table[(index + 47) % 94]; // Deslocar 47 posições
    } else {
      output += _char; // Se o caractere não estiver na tabela, não altera
    }
  }
  return output;
}
```

Fonte: Autores (2023).

Embora a criptografia ROT47 seja considerada fraca e facilmente quebrável, ela pode ser útil para fins de demonstração e para provar a eficácia do *middleware* em lidar com a criptografia de dados. Ao usar ROT47, é possível mostrar o

funcionamento básico do processo de criptografia e descriptografia, mesmo que a segurança oferecida não seja alta.

Portanto, a escolha da criptografia ROT47 neste projeto se baseia em um compromisso entre a limitação de recursos do Arduino e a necessidade de demonstrar a funcionalidade do *middleware*. Para fins educacionais ou de prova de conceito, a criptografia ROT47 pode ser suficiente, mas em cenários reais, é recomendável utilizar algoritmos de criptografia mais robustos e seguros, levando em consideração os requisitos de segurança e os recursos disponíveis.

2.2.3 Desenvolvimento da lógica do middleware

Para realizar a programação necessária no Arduino foi utilizado a IDE oficial do Arduino juntamente da linguagem de programação Arduino.

Para a comunicação da placa Arduino com o shield USB Host, foi utilizado a biblioteca USB Host Shield Library disponibilizada por Kristian Sloth Lauszus no gerenciador de bibliotecas oficial do Arduino. Para o controle do display LCD, foi utilizado a biblioteca oficial LiquidCrystal disponibilizada no gerenciador de bibliotecas oficial do Arduino.

O código utiliza as bibliotecas hidboot.h e LiquidCrystal_I2C para configurar a comunicação com o teclado e o display LCD, respectivamente. O endereço I2C do display é definido como 0x27.

Nesta implementação, foram utilizados apenas os 128 caracteres da tabela ASCII. A tabela ASCII é uma codificação de caracteres que utiliza um conjunto de 128 caracteres, incluindo letras maiúsculas e minúsculas, dígitos numéricos, símbolos de pontuação e caracteres de controle. Esses caracteres são representados por valores numéricos de 0 a 127 e são amplamente utilizados em comunicações e programação.

Utilizando da classe KbdRptParser que é uma subclasse da classe *KeyboardReportParser* e é responsável por analisar os relatórios do teclado USB. Dependendo do modo selecionado no módulo da aplicação que fica no computador, diferentes ações são executadas.

No modo normal, quando uma tecla é pressionada é impressa a correspondente mensagem no monitor serial, mandando a mensagem para ser interpretada pelo módulo no computador.

No modo *encrypt*, quando uma tecla é pressionada, o código verifica se é uma tecla especial. Se for Enter, a mensagem digitada até o momento é criptografada utilizando o algoritmo ROT47 e a mensagem criptografada é enviada via comunicação serial para o computador. Se for uma tecla de seta para a esquerda, o número de rolagem do texto na tela é diminuído em 1. Se for uma tecla de seta para a direita, o número de rolagem é aumentado em 1. Se for uma tecla de exclusão, o último caractere da mensagem é removido. Se for um caractere imprimível, o caractere é adicionado à mensagem.

No modo *decrypt*, a mensagem recebida via comunicação serial é descriptografada utilizando o algoritmo ROT47 e a mensagem descriptografada é exibida no display LCD. Se for uma tecla de seta para a esquerda, o número de rolagem do texto na tela é diminuído em 1. Se for uma tecla de seta para a direita, o número de rolagem é aumentado em 1, permitindo a visualização do restante da mensagem recebida se a mesma conter mais de 32 caracteres.

Utilizando-se da função especial da linguagem de programação do Arduino chamada *setup* que é executada uma vez no início do programa e ficou responsável por inicializar as configurações iniciais, como iniciar a comunicação serial, o display LCD e o modo de exibição.

A função loop também é uma função especial do arduino e executada continuamente em um loop após a função *setup* e é responsável por ler os comandos recebidos pela porta serial (por exemplo, comandos para alterar o modo ou descriptografar uma mensagem) e executar as ações correspondentes. Também é responsável por atualizar as tarefas do USB.

3 EXPERIMENTOS

A presente pesquisa realizou 3 experimentos para investigar as ameaças de segurança num ambiente de chat TCP e avaliar a eficiência do Arduino servindo como um *middleware* de criptografia. Para tal, foi feita uma simulação em um computador equipado com 32GB de memória RAM e um processador 11ª Geração Intel® Core™ i5-1135G7 @2.40GHz, que estava contaminado por dois tipos de *spyware*, sendo um *keylogger* e um *screenlogger*, ambos com o objetivo de coletar dados distintos.

Adicionalmente, foi feito uso do chat TCP presente no repositório do GitHub intitulado "100 Red Team Projects for Pentesters and Network Managers"

[KUROGAI]. Essa implementação possibilita a conexão e a troca de mensagens entre os clientes. Entretanto, devido à ausência de segurança HTTPS, esse ambiente de chat fica exposto a potenciais ataques de interceptação. Adicionalmente, todas as mensagens trocadas no servidor são registradas em um arquivo de log, viabilizando análises futuras.

Para simular a troca de mensagens com dados sensíveis, empregamos um conjunto de cinco mensagens, sendo elas:

- Meu numero de cartao de credito e 4532 8765 1234 5678, validade 10/25, CVV e 123.
- A senha do meu email teste.teste@gmail.com e senhaSegura123.
- O numero do meu CPF e 12345678900.
- Meu endereco residencial e Rua das Flores, 123, Sao Paulo, SP, 01000-000.
- Minha data de nascimento e 12 de julho de 1980.

Esses dados são, naturalmente, fictícios e foram usados apenas para fins de simulação.

A coleta de dados foi realizada em três etapas. Na primeira etapa, coletamos os dados das 5 mensagens trocadas por um cliente no servidor de chat TCP, com a utilização do *middleware* no modo Normal (Sem o emprego de criptografia), no computador infectado com *keylogger* e *screenlogger*. Dessa forma, conseguimos obter informações sobre as teclas digitadas pelos usuários por meio do *keylogger* e capturas de tela durante as interações por meio do *screenlogger*.

Na segunda etapa, repetimos a coleta de dados das mesmas 5 mensagens no servidor de chat TCP, porém, dessa vez, as mensagens passaram pela criptografia do *middleware*. O computador infectado pelos *spyware* também estava presente nessa etapa. Essa etapa nos permitiu avaliar o impacto da criptografia no ambiente infectado.

Na terceira etapa, recuperamos as 5 mensagens criptografadas e as reenviamos ao *middleware* utilizando o modo *decrypt* para verificar a possibilidade de leitura das mensagens novamente.

Com esses experimentos, obtivemos um registro detalhado das teclas digitadas pelos usuários, capturas de tela do computador durante as trocas de mensagens e também o arquivo de log com todas as mensagens enviadas pelos clientes no servidor de chat.

4 RESULTADOS E DISCUSSÃO

Os dados analisados nesta pesquisa provêm do *keyloggers*, do *screenlogger* e do servidor do chat TCP, obtendo-se os resultados em cada etapa.

Na primeira etapa dos experimentos, onde o *middleware* não foi utilizado (modo normal), foi observado que o computador infectado com *keylogger* e *screenlogger* conseguiu registrar todas as teclas digitadas pelos usuários e capturar *screenshots* durante as interações no servidor de chat TCP como podem ser vistas pelas imagens dispostas no apêndice A. Isso evidencia a vulnerabilidade do ambiente de comunicação sem a criptografia adequada.

Já na segunda etapa, em que o *middleware* foi empregado no modo de criptografia, houve uma melhoria considerável na segurança. As mensagens trocadas no servidor de chat TCP foram criptografadas utilizando o algoritmo ROT47 antes de serem transmitidas, garantindo que os dados sensíveis fossem protegidos durante o processo de transmissão. Isso dificultou a leitura dos dados por parte do *keylogger* e *screenlogger*, tornando-os praticamente inúteis para capturar informações confidenciais.

Além disso, a utilização do display LCD no módulo de *hardware middleware* permitiu que as mensagens criptografadas e descriptografadas fossem exibidas de forma clara e legível. Isso facilitou o acompanhamento das mensagens pelos usuários, garantindo que a criptografia estivesse funcionando corretamente.

Na terceira etapa, realizamos o teste de descriptografia das mensagens criptografadas. O *middleware* foi configurado no modo de descriptografia, permitindo que as mensagens criptografadas recebidas fossem descriptografadas novamente e exibidas no display LCD. Demonstrando a capacidade do *middleware* de realizar a criptografia e descriptografia corretamente, fornecendo uma solução completa para a segurança na comunicação de ponta a ponta.

No entanto, durante os testes, identificamos um problema relacionado ao uso do teclado externo com o *middleware* no modo normal. O *input lag*, que é o atraso de tempo entre a ação do usuário e a resposta correspondente exibida na tela, mostrou-se significativo, chegando a aproximadamente mil milissegundos. Podendo impedir o uso do *middleware* em aplicações que exigem uma resposta imediata do sistema. Essa limitação deve ser considerada ao implementar o *middleware* em ambientes em tempo real. Uma restrição adicional decorrente da utilização do Arduino Uno é o tamanho máximo das mensagens enviadas e recebidas, que não

pode exceder 200 caracteres. Essa limitação ocorre devido à alocação de recursos do programa em execução. Constatou-se que o uso de variáveis globais consome 978 bytes (47%) de memória dinâmica, restando apenas 1070 bytes para variáveis locais. Essa limitação foi identificada por meio de testes realizados.

5 CONCLUSÃO

O objetivo geral desta pesquisa foi desenvolver uma solução que aumentasse o nível de segurança na comunicação ponta a ponta por meio de comunicadores instantâneos, utilizando o Arduino para criptografar a interação entre o teclado do usuário e o computador.

O middleware desenvolvido apresentou potencial para melhorar a segurança na troca de mensagens realizadas através do computador, proporcionando uma proteção aprimorada aos usuários. Os experimentos realizados demonstraram que a utilização deste middleware, baseado no Arduino para criptografia, resultou em um reforço significativo na segurança da comunicação ponta a ponta em aplicativos de mensagens instantâneas.

Assim, todos os objetivos desta pesquisa foram alcançados, e os resultados obtidos foram positivos. Para estudos futuros, sugere-se explorar o uso de algoritmos de criptografia mais robustos e seguros, a fim de aprimorar ainda mais a segurança do sistema. Além disso, é importante considerar a minimização do input lag causado pelo middleware, especialmente em aplicações em tempo real que exigem uma resposta imediata do sistema.

Em suma, esta pesquisa contribuiu para o avanço da segurança na comunicação em aplicativos de mensagens instantâneas, oferecendo uma solução viável e eficaz baseada no uso do Arduino como middleware de criptografia.

REFERÊNCIAS

ATMEL. (2016). "8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash - ATmega328/P". Atmel Corporation.

BERGERON, Jean, et al. Static detection of malicious code in executable programs. Int. J. of Req. Eng 2001.184-189 (2001): 79.

BRASIL. Presidência da República/Gabinete de Segurança Institucional. Portaria nº 93, de 26 de setembro de 2019. Aprova o Glossário de Segurança da Informação. Brasília, 2019.

CHECK POINT RESEARCH. May 2022’s Most Wanted Malware: Snake Keylogger Returns to the Top Ten after a long absence. San Carlos, CA, 09 Jun. 2022. Disponível em: https://www.checkpoint.com/. Acesso em: 24 maio 2023.

GLOBALSIGN. Cyber Attacks To Watch Out For in 2023. Disponível em: https://www.globalsign.com/. Acesso em: 24 maio 2023.

HELLOACM.COM. PHP Rot47 Function (str_rot47). 2023. Disponível em: https://helloacm.com/php-rot47-function/. Acesso em: 4 jun. 2023.

INTERNET. In: Oxford Learner's Dictionaries. Oxônia: Oxford University Press, 2023. Disponível em: https://www.oxfordlearnersdictionaries.com/us/definition/english/internet?q=internet. Acesso em: 06 jun. 2023.

JURASKI, Dairon R.; NUNES, Nathan P. Uma Visão Geral sobre Criptografia.

KUROGAI. 100 Red Team Projects for Pentesters and Network Managers. 2023. Disponível em: https://github.com/kurogai/100-redteam-projects. Acesso em: 03 jun. 2023.

PACHGHARE, V. K. Cryptography and information security. PHI Learning Pvt. Ltd., 2019.

PALLARO, Marynea Aparecida Pereira. Avaliação de desempenho de algoritmos criptográficos em microcontroladores Arduino. 2019. Trabalho de Conclusão de Curso. Universidade Tecnológica Federal do Paraná.

RASTOGI, Nidhi; HENDLER, James. WhatsApp security and role of metadata in preserving privacy. arXiv Prepr. arXiv1701, v. 6817, p. 269-275, 2017.

R. VENKATESH and R. K. Sekhar, User Activity Monitoring Using Keylogger, Asia Journal of Information Technology, vol. 15, no. 23, pp. 4758-4762, 2015.

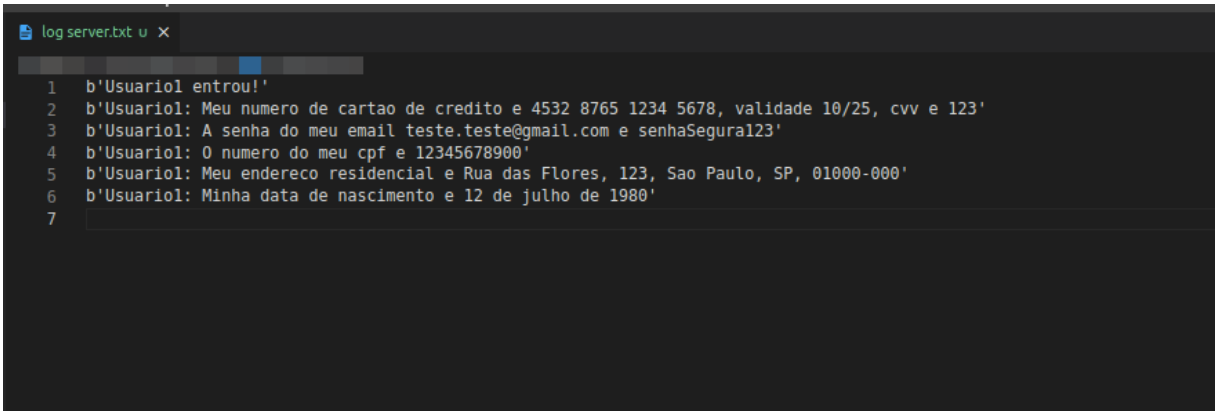
SILVA, Bruna Nunes da. Segurança no WhatsApp Messenger em um estudo de caso com ataque de phishing. 2021. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Pontifícia Universidade Católica de Goiás, Goiânia, 2021

SINGH, Arjun et al. Keylogger detection and prevention. In: Journal of Physics: Conference Series. IOP Publishing, 2021. p. 012005.

ZHANG, Yang; YAN, W.; NARAYANAN, Ajit. A virtual keyboard implementation based on finger recognition. In: 2017 International Conference on Image and Vision Computing New Zealand (IVCNZ). IEEE, 2017. p. 1-6.

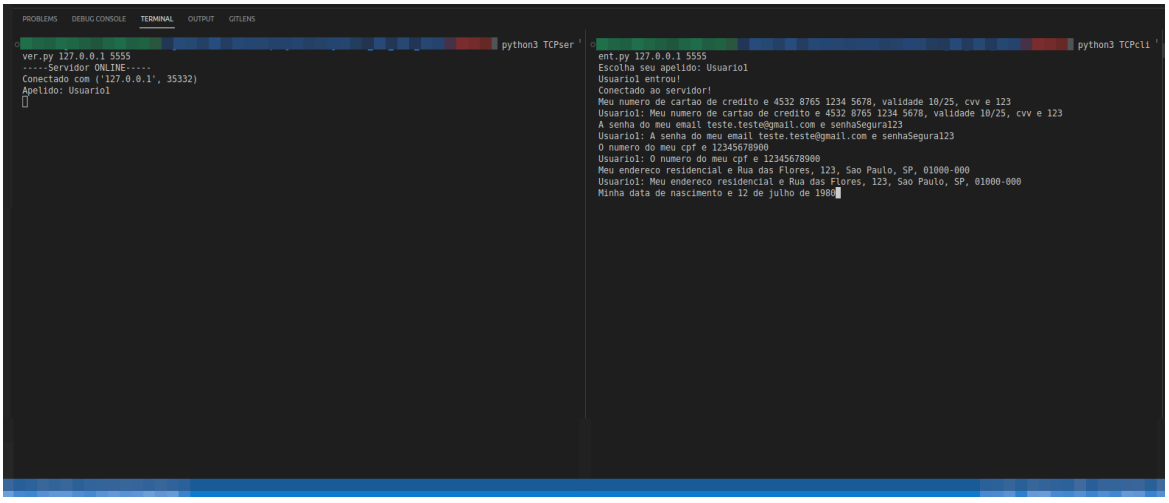
APÊNDICE A - Coleta dos dados do usuário sem o uso do middleware.

Figura 5 - Log do servidor do usuário sem o middleware



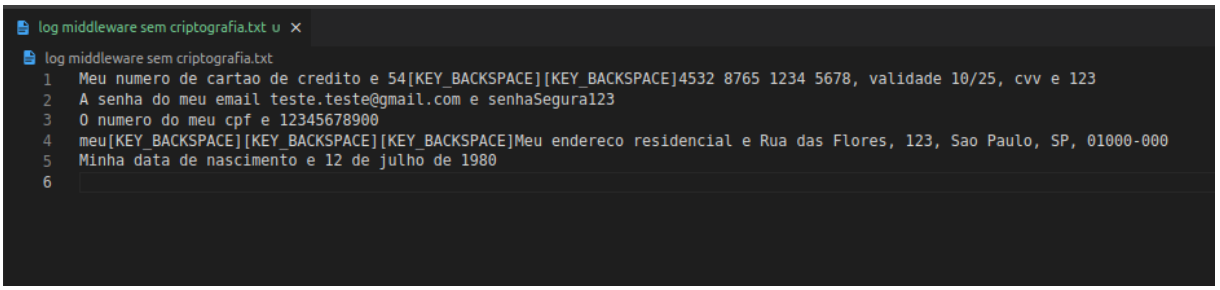
Fonte: Autores (2023).

Figura 6 - Última captura de tela do usuário sem middleware do servidor



Fonte: Autores (2023).

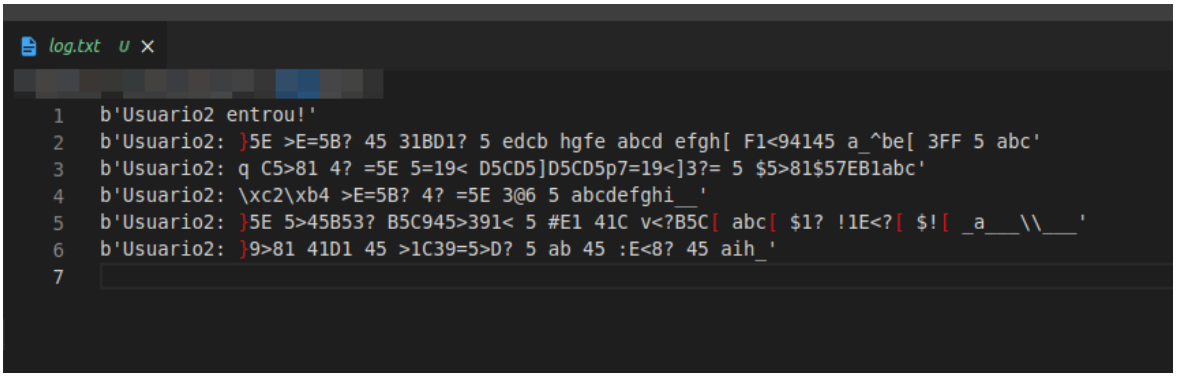
Figura 7 - Log do keylogger do usuário sem a utilização do middleware



Fonte: Autores (2023).

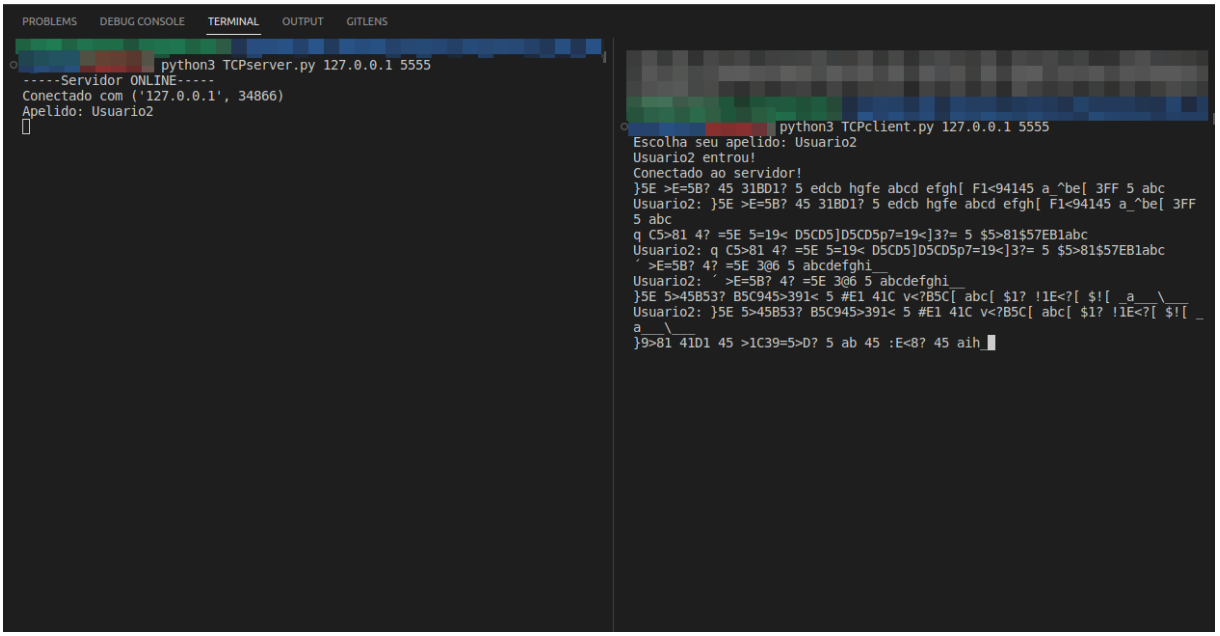
APÊNDICE B - Coleta dos dados do usuário com o uso do middleware.

Figura 8 - Log do servidor do usuário com o middleware



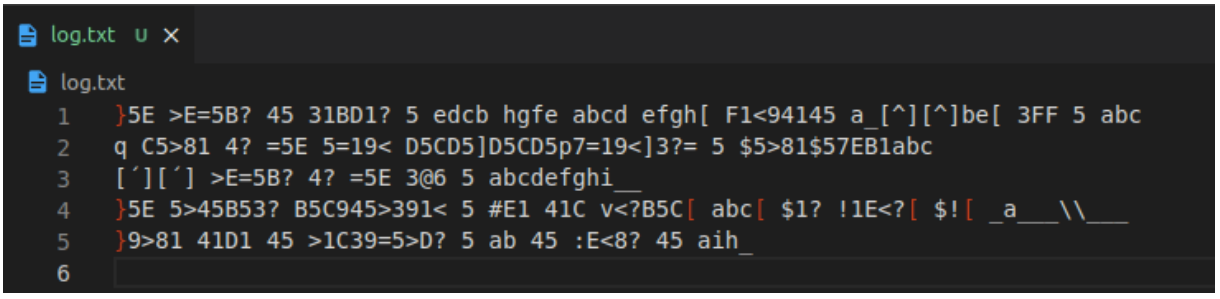
Fonte: Autores (2023).

Figura 9 - última imagem coletada pelo screenlogger do usuário com o middleware



Fonte: Autores (2023).

Figura 10 - Log do keylogger do usuário com o middleware



Fonte: Autores (2023).

APÊNDICE C - Mensagens descriptografadas

Figura 11 - Exibição da mensagem 1 descriptografada é apresentada no middleware (parte 1)

Fonte: Autores (2023)

Figura 12 - Exibição da mensagem 1 descriptografada é apresentada no middleware (parte 2).

Fonte: Autores (2023).

Figura 13 - Exibição da mensagem 1 descriptografada é apresentada no middleware (parte 3)

Fonte: Autores (2023).

Figura 14 - Exibição da mensagem 2 descriptografada é apresentada no middleware (parte 1)

Fonte: Autores (2023).

Figura 15 - Exibição da mensagem 2 descriptografada é apresentada no middleware (parte 2)

Fonte: Autores (2023).

APÊNDICE C - Softwares e bibliotecas utilizadas - Versões

- Visual Studio Code 1.78.2
- Ubuntu 20.04.6 LTS
- pySerial 0.18.0
- Tkinter (versão não especificada)
- PyAutoGUI 0.9.53
- pystray 0.18.0
- Pillow 9.5.0

- multiprocessing 3.8.10
- Python 3.8.10
- Arduino IDE 1.8.19
- USB Host Shield Library 2.0
- LiquidCrystal Library 1.0.7