Desenvolvimento de Token de identificação criptografado para acesso residencial

1st Lincoln Wallace Veloso Almeida
IESTI
Universidade Federal de Itajubá
Itajubá, Minas Gerais
lincoln.wallace@unifei.edu.br

2nd Rodrigo Maximiano Antunes de Almeida
IESTI - Orientador
Universidade Federal de Itajubá
Itajubá, Minas Gerais
rodrigomax@unifei.edu.br

***Abstract*—This paper is about the development of a residential token access that guarantees security against the main radio frequency embedded devices cyber attacks. For this purpose, it's used open-source libraries of encryption and the development of a owner protocol to ensure confidentiality, integrity, and availability (the CIA triad) and also authenticity.**

***Index Terms*—security, cybersecurity, encryption, residences**

***Abstract*—Este trabalho visa o desenvolvimento de um token de acesso residencial que garanta a segurança contra ataques tradicionais à dispositivos embarcados de radiofrequência. Para isso, utiliza-se de bibliotecas Open-Source de criptografia e do desenvolvimento de um protocolo próprio para garantir Confidencialidade, integridade e disponibilidade (tríade CID), além da autenticidade.**

***Index Terms*—segurança, cibersegurança, criptografia, residências**

I. Introdução

O desenvolvimento de sistemas embarcados de uso cotidiano nem sempre leva em consideração requisitos de segurança no que diz respeito à segurança da informação, cyber segurança e proteção de dados. De modo que a integridade destes dispositivos pode ser comprometida a qualquer momento por meio de ataques.

Apesar de grande parte dos dispositivos só serem frágeis a invasões onde se possui o acesso físico ao hardware, uma grande leva de equipamentos IoT (Internet of Things) e de radiofrequência possuem uma grande vulnerabilidade à esses tipos de ameaças. Dentre os desprotegidos, estão aqueles que fornecem acesso às residências como controles de portão de garagem e controles de fechaduras. Uma vez que estes apenas enviam um sinal de radiofrequência com um significado de trigger (um comando simples de abertura). Onde através de subterfúgios, que serão discutidos posteriormente, qualquer infrator pode conseguir replicar tal comando e por consequência ter acesso à residência.

O órgão americano National Institute of Standards and Technology (NIST) define a segurança de dados como sendo a proteção atribuída a um sistema da informação automatizado para atingir objetivos aplicáveis de preservação da integridade, disponibilidade e confidencialidade dos recursos do sistema de informação (incluindo hardware, software, firmware, dados e telecomunicações) [1]. Com base em técnicas já utilizadas nos campos de tecnologia da informação como a criptografia [2] e protocolos de autenticação, é possível garantir a segurança de dados de sistemas de acesso residenciais.

II. Revisão Bibliográfica

A. Criptografia e segurança

A escrita é considerada uma das mais importantes invenções da história humana. Porém, ao mesmo tempo em que as pessoas tem a necessidade de compartilhar informações, há também a necessidade de ocultar informações. Esta necessidade levou a invenção da **criptografia**. Tal artificio está ligado não em esconder a existência de uma mensagem, mas sim o seu significado.

A seguir serão discutidas terminologias importantes para o desenvolvimento deste projeto.

1) Tríade CIA

Sigla em inglês representando o conceito definido pela NIST [1] que vem da junção dos conceitos de **confidencialidade**, **integridade** e **disponibilidade**.

Onde na **Confidencialidade** há a garantia de que os dados são protegidos contra o acesso indevido; em **Integridade** a garantia de que estes não serão modificados ou destruídos e manterão suas características originais; por fim em **Disponibilidade** há a garantia de acessibilidade para os usuários autorizados quando necessário.

2) Mensagem Cifrada

A terminologia refere-se ao texto que teve o seu conteúdo original escondido através de um algoritmo matemático de criptografia. Que funciona como uma função alterando a mensagem escondendo o seu significado através de uma **chave**, podendo esta ser gerada aleatoriamente ou escolhida manualmente, como mostra a Fig. 1

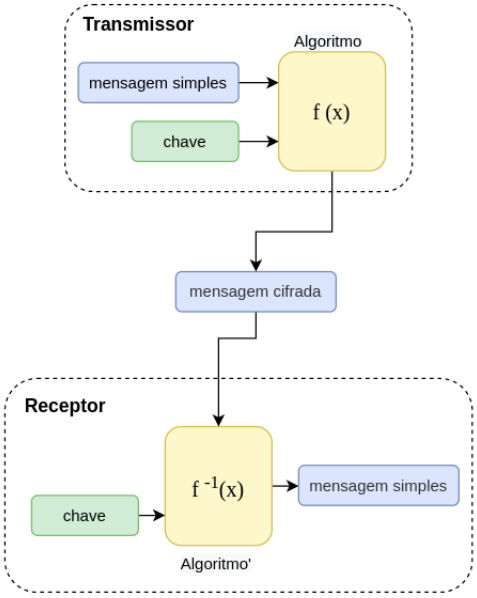

Fig. 1. Mensagem cifrada

O processo inverso ocorre para que a **mensagem cifrada** seja revertida novamente para a **mensagem simples**, ou seja, uma chave e a mensagem cifrada são utilizados como inputs em um algoritmo que realizará a decifragem da mensagem, transformando-a novamente na mensagem simples.

3) Hashing

Este termo refere-se a técnica de colocar uma espécie de assinatura na mensagem para que o receptor possa aferir sobre sua autenticidade, ou seja, se a mensagem original não sofreu alguma alteração. Para realizar tal procedimento, primeiramente, com a mensagem a ser transmitida estruturada, realiza-se um cálculo com um algoritmo pré-determinado em todos os bytes e o resultado do mesmo (hash) é inserido na informação que será transmitida, geralmente ao final.

Dentre os algoritmos conhecidos, os mais utilizados são o checksum, que consiste em realizar a operação XOR (ou exclusivo), há também o Cyclic Redundancy Check (CRC), podendo este ser de 8, 16 ou 32 bits. Dentre as principais características do CRC e checksum está no fato de que ambos são **funções hash não seguras**. O que faz uma função ser criptograficamente segura é a dificuldade de se determinar que input causou determinado output. Para isto, existem os algoritmos da família Secure Hash Algorithm (SHA), padrão definido pela NIST [3]. Dentre as características deste algoritmo temos a sua **irreversibilidade**, em outras palavras, uma vez que a mensagem passa pela função de hash não é possível reverter o resultado para obter a mensagem original novamente, o que garante a segurança desta informação. Assim, os meios para autenticar a integridade da informação tratam-se da realização da comparação entre o hash gerado pela informação recebida e o hash armazenado.

Uma aplicação comum do uso de funções hash está no armazenamento de senhas em banco de dados. Pois, caso haja um vazamento ou alguma invasão o conteúdo original das senhas dos usuários contidos naquele servidor não é revelado.

4) criptografia simétrica e assimétrica

Criptografia simétrica consiste no uso de um algoritmo baseado em operações matemáticas e uma **chave secreta** ou **chave criptográfica** para gerar uma mensagem cifrada a partir de um texto simples como mostra a Fig. 2. A partir do texto cifrado, o receptor da mensagem deverá utilizar a mesma chave utilizada para cifrar o texto, no processo de descriptografar a mensagem de volta em texto simples. Assim, utiliza-se a mesma chave tanto para criptografar quanto para descriptografar. Logo, ambos as partes envolvidas no processo de transmissão de dados devem conhecer previamente a chave simétrica.

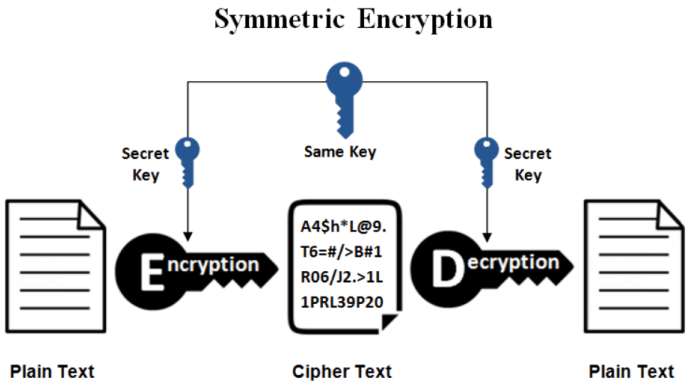

Fig. 2. Criptografia Simétrica [4]

Já na **Criptografia Assimétrica**, receptor e transmissor possuem pares de chaves **pública** e **privada**. Antes de iniciar a transmissão de dados, os dispositivos que desejam se comunicar geram tais pares de chaves e trocam as chaves públicas entre si. Quando um dos dispositivos deseja transmitir uma informação, esta é criptografada através de um algoritmo matemático e da **chave pública** do dispositivo destinatário, então ao ser enviada, só será possível descriptografar essa mensagem utilizando a **chave privada** do destinatário, que por sua vez está em posse apenas do dispositivo destinatário, como mostra a Fig. 3.

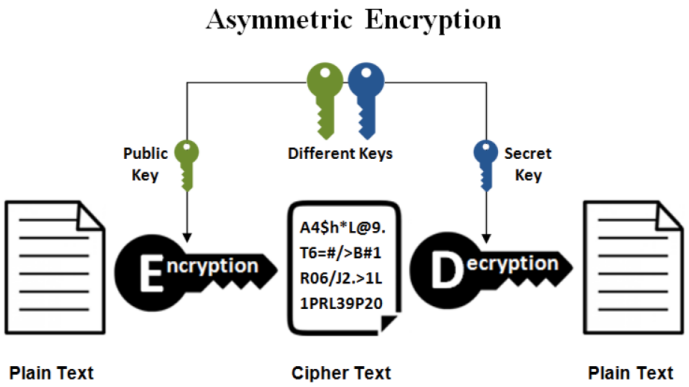

Fig. 3. Criptografia assimétrica [4]

B. Ataques

Exite uma diversa gama de ataques que podem ocorrer em todo e qualquer sistema da informação, dentre estes, será contextualizado aqueles mais ocorrentes em sistemas embarcados *bare-metal*.

1) Eavesdropping

Também conhecido pelos nomes sniffing e snooping, este tipo de ataque acontece quando um hacker intercepta, deleta ou modifica as informações transmitidas entre dispositivos [5]. O ataque depende de comunicações de rede não seguras para acessar dados em trânsito entre dispositivos. Tipicamente ocorre quando um usuário conecta-se a uma rede com tráfico de dados inseguros/não criptografados. Tais métodos podem ser difíceis de se identificar, visto que diferente das outras formas de *cyber ataques*, a presença de um dispositivo escutando a transmissão pode não causar uma lentidão notável ou afetar a performance dos dispositivos ou rede.

No contexto deste artigo, um exemplo de ataque de *eavesdropping* seria a captação do sinal transmitido pelo token de acesso residencial através de uma analisador de espectro eletromagnético e a sua posterior decodificação.

2) Análise eletromagnética

Similar ao ataque de *eavesdropping* este ataque fornece aos hackers outra maneira de olhar dentro do software incorporado sem invadi-lo. Os hackers podem usar a análise eletromagnética para registrar e analisar as emissões de um dispositivo, descobrir suas operações criptográficas e até extrair chaves secretas. Este é o tipo de ataque mais demorado e caro, pois requer:

- Proximidade física do sistema embarcado alvo;
- Informações sobre o layout do dispositivo no qual o sistema está incorporado;
- Isolamento de outros dispositivos para proteger o sistema contra interferência eletromagnética.

3) Replay Attack

Este ataque pode ser complementar aos dois anteriores. Trata-se basicamente de capturar uma transmissão de dados e replicá-la, fazendo com que o sistema entenda essa transmissão como sendo legítima do dispositivo que a enviou originalmente na primeira vez. Como por exemplo, um sinal de um controle de portão comandando a abertura de um portão eletrônico pode ser capturado utilizando técnicas de *eavesdropping* e análise eletromagnética e então replicado para que o portão seja aberto/fechado de modo indevido. A condição para que tal ataque ocorra está na ausência de uma aferição da autenticidade do comando.

4) Notação Big-O

Em ciência da computação, para se classificar a eficiência de um algoritmo, utiliza-se a ferramenta de notação Big-O [6], onde se relaciona o tamanho da entrada do algoritmo (elementos) com os ciclos de operação de CPU necessários para a realização do mesmo (operações). Deste modo, quanto mais instruções são necessárias para executar um algoritmo para uma dada entrada, menor é a sua eficiência e pior é a curva do algoritmo, o oposto também é valido como mostra a figura 4.

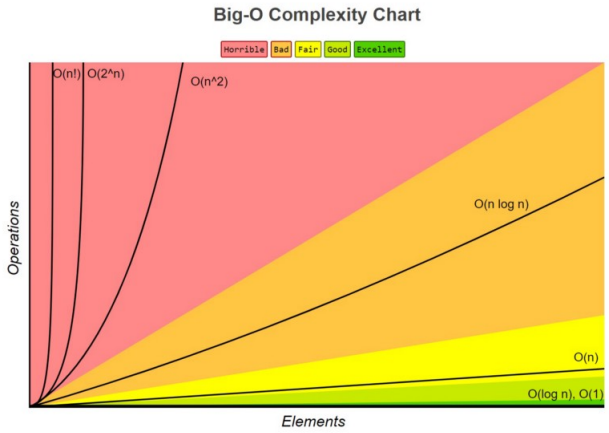

Fig. 4. Notação Big-O [6].

Logo, quanto menor é o número de clocks necessários para a operação, mais rápida é a execução do algoritmo. Além disso, quanto mais ineficiente computacionalmente é o algoritmo, maior é o número de ciclos de clock consumidos para a realização do mesmo e, portanto, maior é o gasto energético.

C. Instrumentação e Hardware

A instrumentação é a ciência que estuda, desenvolve e aplica instrumentos de medição e controle de processos [7]. Os elementos responsáveis pela aquisição do ambiente a sua volta são os chamados sensores. Os componentes físicos que realizam o processamento e as operações necessárias ao funcionamento do sistema são chamamos de *hardware*.

1) Microcontrolador RP2040

O microcontrolador RP2040 [8] trata-se de um componente desenvolvido recentemente pela fundação *Raspberry Pi* e que possui um processador dual core ARM Cortex M0+. Demais recursos presentes no chip são mostrados na fig.5

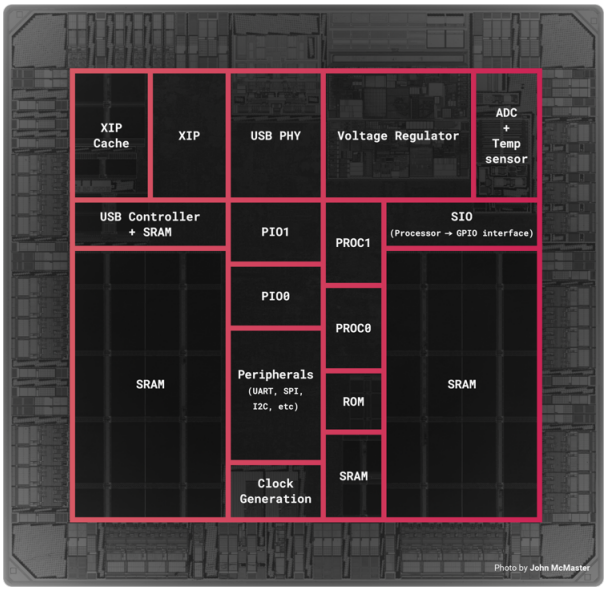

Fig. 5. RP2040.

2) Quad-SPI

Baseado no barramento de comunicação de Serial Peripheral Interface (SPI), *Quad-SPI* ou QSPI [9] trata-se de uma interface serial onde 4 terminais de transferência de dados são utilizados para operações de leitura, escrita e apagar a memória flash de chips. É especialmente útil em aplicações que envolvem uma alta transferência de dados e um uso intensivo de memória, como multimídia e em casos onde a memória interna do chip não é suficiente. Este tipo de barramento foi especificamente desenvolvido para se comunicar com chips de memória flash que suportam esta interface.

3) Radio nrf24l01

O módulo NRF24l01 [10], trata-se de um transceptor, logo, possui a capacidade de realizar a transmissão e recepção de dados. Cada módulo tem uma faixa de 125 endereços podendo ser utilizados para se comunicar com os outros 6 módulos e também permite que várias unidades sem fio se comuniquem entre si em um local especificado. Operando na faixa de rádio ISM (Industrial, Cientifica e médica) de 2.400 - 2.4835GHz, utilizando modulação GFSK, possui CI de SPI e alguns outros componentes passivos necessários para a operação de um sistema de rádio. A interface de operação do rádio se dá através do periférico de SPI. Além disso o módulo possui um alcance de até 100 metros e uma taxa de transmissão de até no máximo 2Mbps.

III. Desenvolvimento

Para o desenvolvimento do *token* criptografado de acesso residencial, dividiram-se os esforços em 3 etapas, sendo elas: Os hardwares (do *token* e da tranca), protocolos de segurança (de cadastro e utilização diária) e Firmware.

A. Hardware

As tomadas de decisão para o desenvolvimento do hardware do *Token* e da Tranca foram realizadas de forma a prevalecer um custo acessível e de fácil implementação. As figuras 8 e 9 representam um sistema macro dos hardwares do Token e da tranca que serão discutidos nas sessões posteriores.

Para realizar a prototipagem do projeto, fez-se o uso da placa de desenvolvimento **Raspberry Pi Pico** [11] e o módulo de rádio **NRF24L01** como mostram as figuras 6 e 7 respectivamente, uma vez que possuem todos os recursos de hardware representados nas figuras 8 e 9. Atributos dos quais serão discutidos posteriormente.

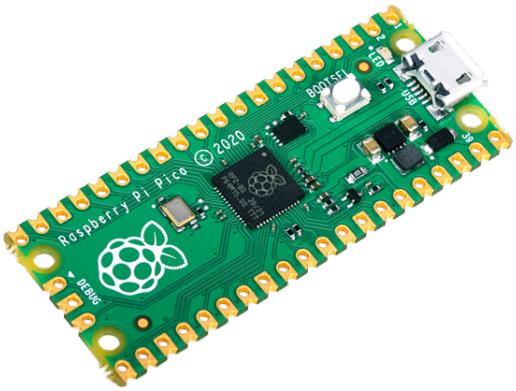

Fig. 6. Placa de Desenvolvimento ”Raspberry Pi Pico”.

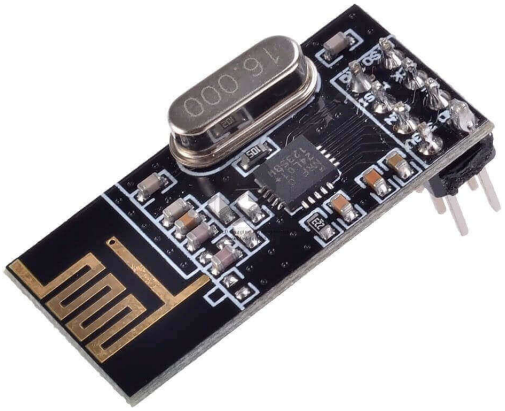

Fig. 7. Módulo de rádio NRF24L01.

1) Microcontrolador

Em primeiro lugar, foi pensado o microcontrolador para o hardware do Token e tranca, para estes foi escolhido o MCU RP2040. A justificativa pela escolha de tal microcontrolador está no fato de que este por ser um modelo recém lançado

ainda não sofreu com os efeitos da atual **crise de semicondutores** [12].

O microcontrolador possui uma alta disponibilidade no mercado, isto se deve ao fato de que por se tratar de um lançamento consideravelmente recente [13], o chip ainda está passando por uma fase de **prova de conceito (POC)** [14] em grande parte dos setores da industria de semicondutores e embarcados.

Outra vantagem está no fato de que o modelo de microcontrolador escolhido encontra-se no preço de $\$1,00$ USD.

A desvantagem no uso deste MCU está no fato de que é necessário o uso de uma memória externa quad-spi para a gravação do programa, como mostra as figuras 9 e 8, porém esta desvantagem pode ser vista também como uma vantagem, pois possibilita a flexibilização do tamanho da memória de programa. Onde o mesmo microcontrolador pode executar firmware's que tem requerimentos diferentes de armazenamento para memória flash (ou memória de programa).

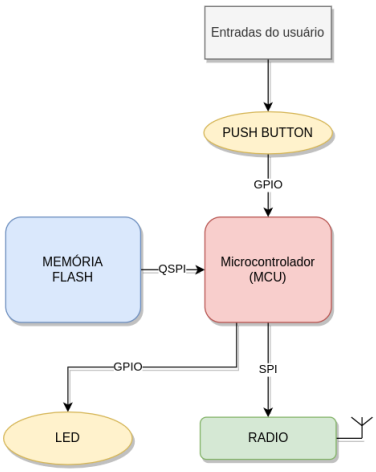

Fig. 8. Diagrama macro do sistema do Token.

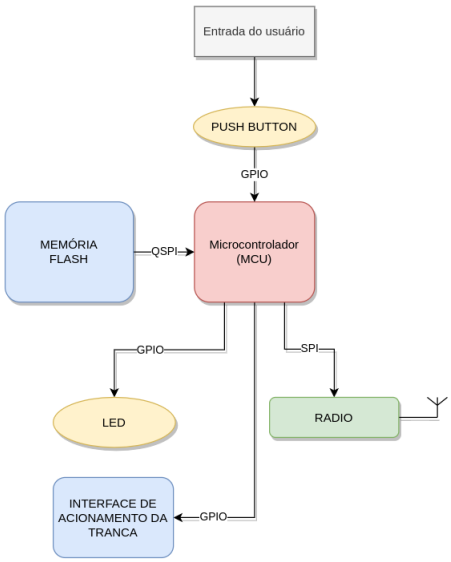

Fig. 9. Diagrama macro do sistema da tranca.

2) Memória flash

Para o desenvolvimento utilizou-se uma memória flash de $16Mbit$ que é maior ou do mesmo tamanho que a média das memórias disponíveis de cerca de no máximo $16Mbit$ ($2MBytes$) ou em grande parte dos microcontroladores atuais.

3) Rádio

A interface de transferência de dados wireless escolhida foi a utilização de um rádio que opera na faixa de 2.4GHz, possuindo uma modulação GFSK como já descrito anteriormente. A justificativa para a escolha de tal componente está no fato de que por se tratar de um rádio digital simples, operando na faixa ISM, a solução dispensa o uso de infraestrutura de telecomunicações de terceiros. Nas redes mobile por exemplo LTE [15] e/ou GSM existe a necessidade de contratação de serviços para a utilização do recurso, como também acontece no caso da rede Narrowband IoT (NbIoT). Dentre as alternativas, existe a opção do uso de modem's Lora utilizando uma configuração de ponto a ponto e que portanto dispensaria o uso de um gateway LoraWAN ou rede Lora por exemplo existe a necessidade de

4) Interface com o usuário

Para a interação homem-máquina fez-se o uso de push button's como interface padrão de entrada do usuário e 2 led's como saídas, onde cada possível estado destes representa uma condição dos sistemas do token e da tranca. Seja estes estados acesso, apagado ou piscando.

B. Protocolo de Segurança

Grande parte do desenvolvimento do projeto do Token se deu em torno da criação de um protocolo de segurança proprietário (PSP), tal qual garantisse os princípios da tríade CIA sem permitir que a complexidade do sistema escalasse um nível em que não fosse mais possível e/ou viável a utilização de um sistema micro-controlado simples sem a utilização

de uma camada de sistema operacional, o chamado ”bare-metal” permitindo assim um custo acessível. O protocolo de segurança foi divido em duas categorias que serão discutidas posteriormente referentes a 2 momentos distintos do uso do sistema. A primeira sendo o **cadastro do Token** e posteriormente o **uso diário** do mesmo. Com relação ao padrão de criptografia, optou-se por abordar um **algoritmo com padrão simétrico** devido ao fato de que os algoritmos de criptografia assimétricos terem um custo computacional maior, o que leva um maior gasto energético e maior tempo de processamento, como discutido anteriormente.

1) Cadastro do Token

1 Uma medida tomada para garantir a autenticidade e disponibilidade: Ambos os sistemas da tranca e token possuem a chave simétrica padrão (**CSP**) gravadas em firmware. A primeira etapa do cadastro é descrita pela figura 10. Esta etapa é responsável pela geração de um ID para o **SToken** e também a identificação do deste pela **STranca**.

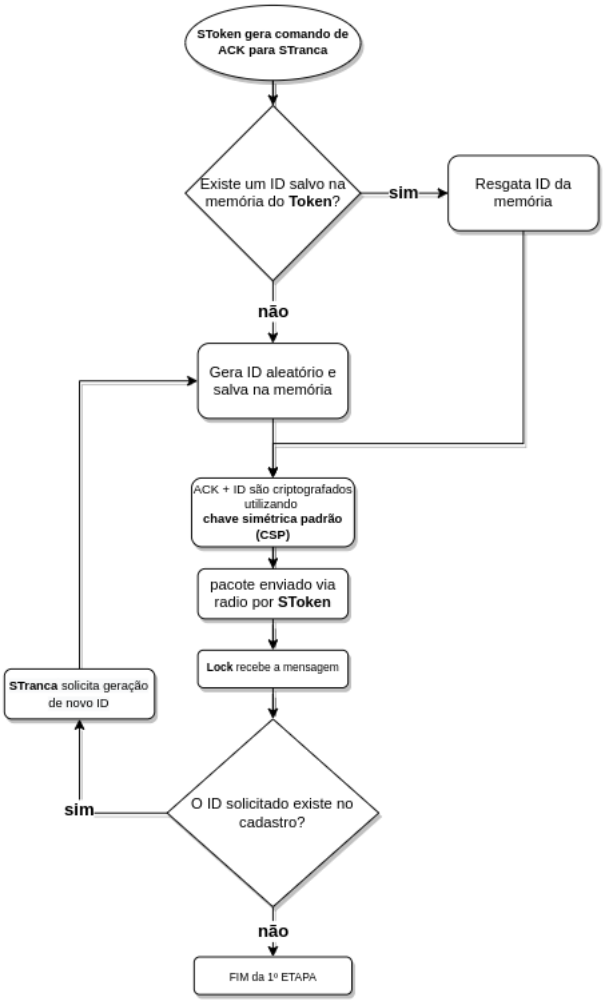

Fig. 10. Algoritmo primeira etapa de cadastro.

2 A partir da identificação inicia-se o então o cadastramento com uma autenticação do **SToken** que solicitou o cadastro. Verificando se o ID contido na resposta é o mesmo que solicitou o cadastramento na primeira etapa. Como mostra a figura 11 a seguir:

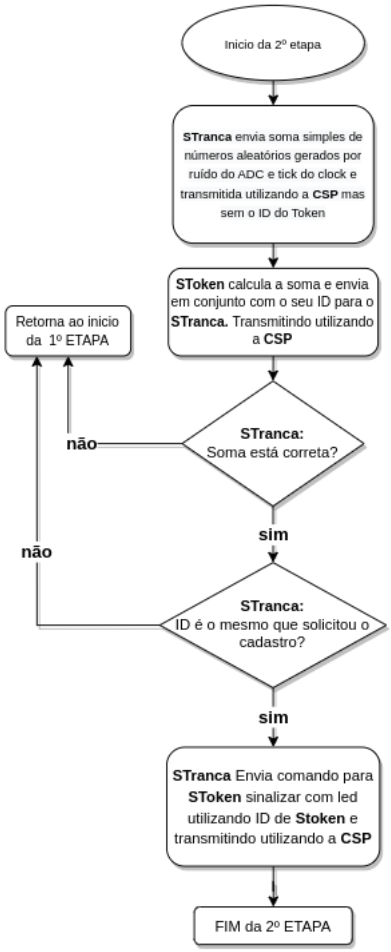

Fig. 11. Algoritmo segunda etapa de cadastro.

3 A partir do momento em que a autenticação é concedida por parte do **STranca** é feito o envio do comando para o LED piscar. Cabe então ao usuário confirmar o sinal que garante a autenticidade do **SToken** a ser cadastrado ou constatar um timeout e reiniciar o procedimento, como mostra a figura 12

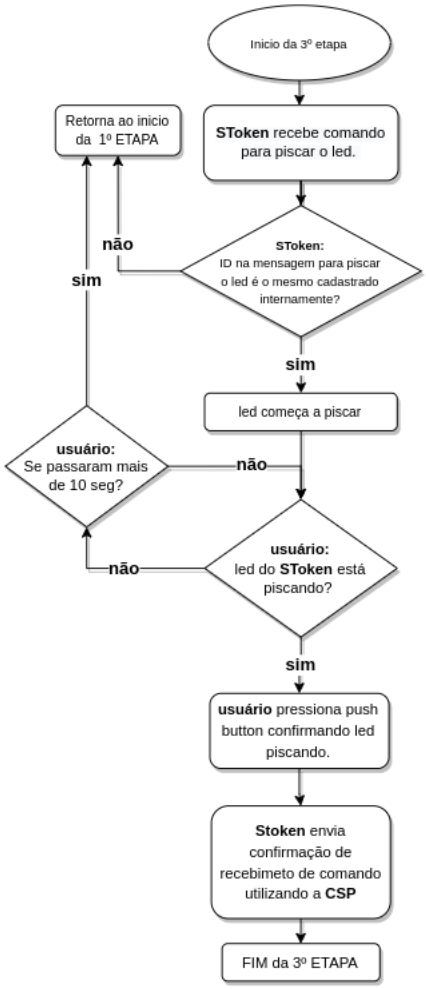

Fig. 12. Algoritmo terceira etapa de cadastro.

4 Assim que o **STranca** recebe a confirmação do usuário de que o **SToken** em questão foi identificado corretamente, é gerado uma nova chave simétrica. A nova chave simétrica é então enviada para o Token e também registrada na memória e associada ao ID do **SToken** cadastrado. A **CSP** deixa então de ser utilizada.

Fig. 13. Última etapa de cadastro.

Assim, finalize-se o procedimento de cadastro de um novo **SToken**.

2) Uso diário do Token

Com o cadastro realizado, o uso diário do token nos comandos de abrir/fechamento se darão através de um outro padrão.

1 Para o protocolo de uso cotidiano, a primeira etapa trata-se apenas da requisição feita pela **SToken**, utilizando como mecanismo de criptografia a chave simétrica gerada no momento do cadastro.

Fig. 14. Protocolo de uso cotidiano primeira parte.

2 Assim que o **STranca** recebe a solicitação uma série de checagens é feita para validar a autenticidade e a integridade da requisição. Tal como o timestamp enviado e o número de solicitações já feitas.

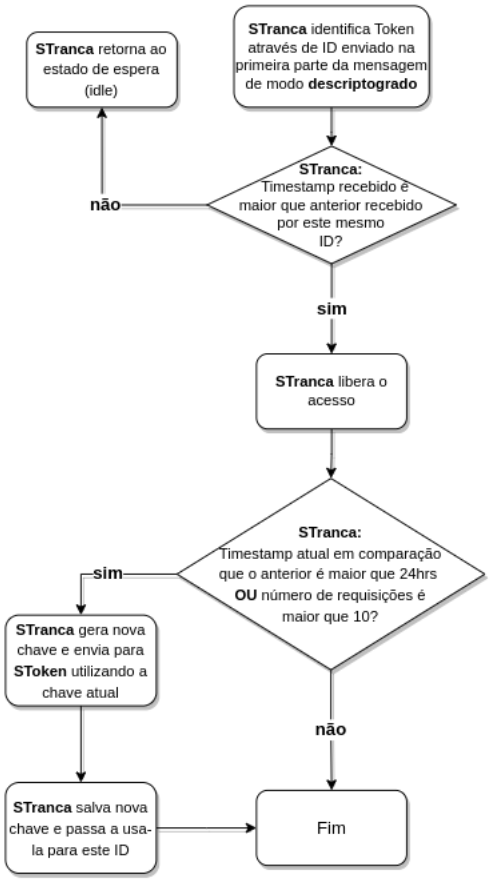

Fig. 15. Protocolo de uso cotidiano segunda parte.

Vale também notar que todas as comunicações entre os dois sistemas são precedidas de uma assinatura de hash.

C. Firmware

O firmware para o projeto foi escrito em **linguagem C++**, utilizando a toolchain GNU para ARM **arm-none-eabi**. Para acesso aos periféricos do chip utilizou-se da SDK do microcontrolador RP2040 [16]. Que utiliza a ferramenta de build automatizada **CMake**.

A implementação do firmware em grande parte baseia-se na implementação do protocolo de segurança em si. O hardware utilizado trata-se de um protótipo utilizando a placa de desenvolvimento **Raspberry Pi pico** e o módulo de rádio NRF24l01.

No desenvolvimento de softwares em C++, uma grande parcela do problema está na garantia de um bom gerenciamento de memória.

A garantia de um bom gerenciamento de memória está entre os maiores desafios, senão o maior no desenvolvimento de softwares em C/C++. O problema é tão grave, que a Microsoft relatou o gerenciamento de memória como sendo uma das principais causas de bugs de segurança, sendo a causa de 70% destes [17]. A Google também informou que 70% de suas falhas e bugs de segurança são causados por problemas com gerenciamento de memória [18].

Quando visto da perspectiva de Sistemas Embarcados a preocupação com o gerenciamento de memória é anida mais preocupante, visto que em sistemas bare-metal não há uma camada de sistema operacional para proteger eventuais falhas de software e fazer com que o hardware se comporte de uma forma segura.

Devido a estas preocupações já existem padrões na industria como por exemplo o **Misra C**, **Misra C++** e **AUTOSAR**, amplamente adotados no setor automotivo.

Para o desenvolvimento do Firmware dos sistemas do Token e Tranca foram utilizados algumas das diretivas presentes de versões de rascunho do MISRA C++ 2008 encontradas [19].

Também adotou-se técnicas de **programação defensiva** como por exemplo a ativação de flags de compilação:

- **-Wall**: Responsável por ativar todos os warnings do compilador.
- **-Wextra**: Ativa warnings extras além dos ativados por **-Wall**.
- **-Werror**: Transforma todos os warnings em erros obrigando o tratamento destes, adotando assim uma política ”zero warning”.

Dentre outras flags ativadas.

Para a arquitetura de código, utilizou-se da arquitetura de **Máquina de estados** por se tratar de uma estrutura mais apropriada para este tipo de problema. O código será disponibilizado no Github [20] para eventual consulta.

IV. Resultados

Analisando se profundamente o funcionamento do sistema do Token, foi possível concluir que durante o seu **uso diário** o sistema possui pouca vulnerabilidade. Foi possível construir portanto um protótipo de um dispositivo imune a ataques de replay attack, devido ao fato de que nenhuma solicitação de abertura/fechamento é igual. Logo mesmo se capturada através de um mecanismos de eavesdropping como um analisador de espectro não é possível reutilizar esta transmissão cifrada esperando que a mesma funcione novamente.

Relativo ao protocolo de uso cotidiano, pode-se perceber que a troca frequente das chaves não permite que haja uma coleção de dados, ou seja, mensagens cifradas para que possa ser possível reverte-las novamente em texto simples através de técnicas computacionais de engenharia reversa.

Em relação ao protocolo de cadastro pode-se afirmar que o resultado foi atingido com certa quantidade de Overengineering, pois resultou se em um protocolo demasiadamente complexo. Realizando a Análise qualitativa de Big-O pode-se afirmar que o procedimento resultou em um algoritmo da categoria **O(1)** visto que o número de interações é sempre o mesmo, uma vez que a quantidade de dados de entrada não se altera.

V. Conclusão

Dentro dos parâmetros estipulados, obteve-se êxito no desenvolvimento do protótipo, que mostrasse os conceitos de segurança com simplicidade em Hardware e robustez em desenvolvimento de firmware.

A. Atualizações futuras

Percebeu-se, durante o desenvolvimento do projeto, a necessidade de algumas atualizações para uma versão comercial do produto. Mesmo utilizando alguns dos princípios e guidelines do MISRA C++, boas práticas de código como Clean code, ativando todos os warnings e tratando-os como erros, não existe nenhuma garantia de que não ocorrerão problemas futuros com corrompimento de memória. Seja devido ao acesso ilegal de regiões de memória, memory leaks, stack overflow entre outros problemas causados pelo gerenciamento de memória indevido em C++. A raiz deste problema está no fato de que ambas linguagens C e C++ não são construídas e definidas baseadas em critérios de memory safety. Portanto uma solução para tal problema está na adoção de uma linguagem que por definição é memory safe como a Linguagem Rust. Logo, uma possível melhoria na segurança interna do sistema estaria na reescrita do firmware em Rust, visto que a linguagem vem sendo amplamente adotada no contexto de sistemas embarcados [21].

Uma das possíveis melhorias a ser aplicada de modo a se reduzir o custo final do sistema é a troca da memória flash do sistema do token por uma de menor tamanho, visando a redução de custo desta parte do hardware. Outra importante melhoria a ser feita com relação ao sistema da tranca está no projeto de uma interface de hardware para o devido acionamento da tranca. Uma possibilidade de melhoria que impactaria no aumento da segurança estaria na implementação de um chip de criptografia dedicado.

Outra melhoria interessante seria realizar o cadastramento de novos tokens utilizando canais seriais ao invés de rádios. Isto faria com que grande parte da complexidade trazida pelo protocolo de cadastramento seria eliminada.

VI. Agradecimentos

Ao meu orientador pela dedicação e empenho da realização deste trabalho. Ao meu coordenador pela elaboração de uma grade de curso tão rica. À universidade pelo incrível ambiente de fomento ao desenvolvimento. Ao meu primo Felipe Lucchini Almeida que foi uma das minhas grandes inspirações para se adentrar no mundo da engenharia. À Raspberry Pi Foundation pela iniciativa da democratização e flexibilização no ensino da computação.

References

[1] M. Swanson and B. Guttman, "Generally accepted principles and practices for securing information technology systems," 1996-09-03 1996. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=890092

[2] Q. Liu, Y. Li, L. Hao, and H. Peng, "Two efficient variants of the rsa cryptosystem," in *2010 International Conference On Computer Design and Applications*, vol. 5, 2010, pp. V5–550–V5–553.

[3] Q. Dang, "Secure hash standard," 2015-08-04 2015.

[4] "Symmetric vs. asymmetric encryption – what are differences?" [Online]. Available: https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences

[5] Fortinet, "Eavesdropping." [Online]. Available: https://www.fortinet.com/resources/cyberglossary/eavesdropping

[6] P. Rail, "All you need to know about "big o notation" to crack your next coding interview," 2018. [Online]. Available: https://www.freecodecamp.org/news/all-you-need-to-know-about-big-o-notation-to-crack-your-next-coding-interview-9d57

[7] M. de Roure, "Instrumentação industrial – entenda de uma vez por todas," 2017. [Online]. Available: https://instrumentacaoecontrole.com.br/instrumentacao-industrial-guia-completo/

[8] R. P. Foundation, "Documentação RP2040." [Online]. Available: https://www.raspberrypi.com/documentation/microcontrollers/rp2040.html

[9] B. Gunasekaran, "*Quad-SPI, Everything You Need To Know!*" 2020. [Online]. Available: https://embeddedinventor.com/quad-spi-everything-you-need-to-know/

[10] N. Semicondutor, "nrf24l01 single chip 2.4ghz transceiver product specification." [Online]. Available: https://br.mouser.com/datasheet/2/297/NRSAS00020_1-2559917.pdf

[11] "Raspberry pi pico datasheet an rp2040-based microcontroller board." [Online]. Available: https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf

[12] S. Frieske, Benjamin e Stieler, "The "semiconductor crisis" as a result of the covid-19 pandemic and impacts on the automotive industry and its supply chains," *World Electric Vehicle Journal*, vol. 13, no. 10, p. 189, 2022.

[13] "Raspberry pi rp2040." [Online]. Available: https://datasheets.raspberrypi.com/rp2040/rp2040-product-brief.pdf

[14] D. Goodwin, "Proof of concept, prototype, mvp, and mmp in embedded software development," 2019. [Online]. Available: https://www.bluefruit.co.uk/quality/poc-prototype-mvp-mmp-embedded/

[15] H. Remmert, "What is lte: How it works and why it matters," 2021. [Online]. Available: https://www.digi.com/blog/post/what-is-lte

[16] R. Pi, "Raspberry pi pico c/c++ sdk," 2022. [Online]. Available: https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-c-sdk.pdf

[17] C. Cimpanu, "Microsoft: 70 percent of all security bugs are memory safety issues." [Online]. Available: https://www.zdnet.com/article/microsoft-70-percent-of-all-security-bugs-are-memory-safety-issues/

[18] ——, "Chrome: 70% of all security bugs are memory safety issues." [Online]. Available: https://www.zdnet.com/article/chrome-70-of-all-security-bugs-are-memory-safety-issues/

[19] "Misra c++ 2008 standards." [Online]. Available: https://www.cppdepend.com/misra-cpp

[20] L. W. V. Almeida, "Tfg - final graduation work source code." [Online]. Available: https://github.com/LOCNNIL/TFG

[21] M. Ling, Y. Yu, H. Wu, Y. Wang, J. R. Cordy, and A. E. Hassan, "In rust we trust – a transpiler from unsafe c to safer rust," in *2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2022, pp. 354–355.