PROTOCOLO SEGURO PARA INJEÇÃO DE CÓDIGO EM CUBESATS

ALEXANDRE HENRIQUE RADIS

DISSERTAÇÃO DE MESTRADO PROFISSIONAL EM ENGENHARIA ELÉTRICA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SECURE PROTOCOL FOR CODE INJECTION INTO CUBESATS

PROTOCOLO SEGURO PARA INJEÇÃO DE CÓDIGO EM CUBESATS

ALEXANDRE HENRIQUE RADIS

ORIENTADOR: DANIEL CHAVES CAFÉ, DR.
COORIENTADOR: JOÃO JOSÉ COSTA GONDIM, DR.

DISSERTAÇÃO DE MESTRADO PROFISSIONAL
EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPEE.MP-049

BRASÍLIA/DF: JUNHO - 2023

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROTOCOLO SEGURO PARA INJEÇÃO DE CÓDIGO EM CUBESATS

ALEXANDRE HENRIQUE RADIS

DISSERTAÇÃO DE MESTRADO PROFISSIONAL SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

_____

Prof. Dr. Daniel Chaves Café – ENE/Universidade de Brasília
Orientador

_____

Prof. Dr. Daniel Alves da Silva – LATITUDE/Universidade de Brasília
Membro Interno

_____

Profª. Drª. Janaina Goncalves Guimarães – Engenharia de Controle e Automação/Universidade Federal de Santa Catarina - Blumenau
Membro Externo

BRASÍLIA, 30 DE JUNHO DE 2023.

FICHA CATALOGRÁFICA

RADIS, ALEXANDRE HENRIQUE
Protocolo seguro para injeção de código em cubesats [Distrito Federal] 2023.
xii, 48p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2023).
Dissertação de Mestrado Profissional – Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Injeção de Código	2. CubeSat
3. Segurança das Comunicações	4. FreeRTOS
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

RADIS, A. H. (2023). Protocolo seguro para injeção de código em cubesats . Dissertação de Mestrado Profissional em Engenharia Elétrica, Publicação PPEE.MP-049, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 48p.

CESSÃO DE DIREITOS

AUTOR: Alexandre Henrique Radis
TÍTULO: Protocolo seguro para injeção de código em cubesats .
GRAU: Mestre ANO: 2023

Alexandre Henrique Radis

Departamento de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Dedicatória escrita por ALEXANDRE HENRIQUE RADIS*: Dedico esse trabalho a toda minha família.*

ACKNOWLEDGMENTS

A necessidade de evolução nos estudos foi um ensinamento que tive desde pequeno, seja por meus pais ou por meus avós. E foram encorajados e apoiados por eles, assim como por tios e primos. Tive também bons exemplos de professores em vários níveis escolares, desde o ensino básico ao ensino superior.

Mas a realização desse sonho de fazer um mestrado só foi possível graças à oportunidade oferecida pela ABIN e pela UnB para um mestrado profissional em segurança das comunicações. Portanto, gostaria de agradecer aos colegas do CEPESC pelo incentivo e apoio. Agradeço também aos professores da UnB pelo conhecimento apresentado e pelas orientações e incentivos.

E por último e mais importante, gostaria de agradecer a minha esposa e filhos pela compreensão e apoio pelas horas que perdi de estar com eles no dia-a-dia para a realização deste sonho.

RESUMO

Título: Protocolo seguro para injeção de código em cubesats
Autor: Alexandre Henrique Radis
Orientador: Daniel Chaves Café, Dr.
Coorientador: João José Costa Gondim, Dr.
Programa de Pós-Graduação em Engenharia Elétrica
Brasília, 30 de junho de 2023

Um projeto de satélite do tipo CubeSat começa com a identificação de suas necessidades e continua com seu desenvolvimento, montagem, lançamento, operação e obsolescência. No entanto, as necessidades podem mudar ao longo do ciclo de vida do satélite, como em qualquer projeto. Nos satélites do tipo CubeSat, a inclusão de novos serviços se torna um grande desafio, devido à impossibilidade de acesso físico aos equipamentos.

Injeção de código é uma solução que permite a inclusão de novos serviços em um satélite após seu lançamento. A inclusão de novos serviços em equipamentos microcontrolados apresenta diversos desafios de segurança, principalmente em satélites do tipo CubeSat, que possuem restrições de energia, comunicação, processamento, memória, entre outros. É necessário proteger o sistema microcontrolado contra ataques de negação de serviço, violação de dados, desativação de equipamentos e sequestro. Não é possível usar técnicas como firewall, antivírus ou inteligência artificial.

Como a inclusão de novos serviços nos microcontroladores significa a inclusão de novos códigos, e isso significa abrir uma grande oportunidade para ataques. É necessário mitigar esses ataques. Assim, o trabalho apresenta uma proposta de inclusão de novos códigos mitigando a possibilidade de ataques efetivos. Esta proposta é composta por medidas de segurança, protocolos de comunicação, utilização de HMAC para garantir o cumprimento e integridade dos novos códigos, e um sistema operativo em tempo real preparado para este desafio.

O trabalho apresenta um estudo do estado da arte e um referencial bibliográfico sobre o assunto. Segue-se a proposta conceptual, a metodologia de implementação e teste dos conceitos, resultados e conclusões obtidos. Entre os resultados obtidos, foi possível observar a viabilidade das medidas proposta, a defesa de ataques de tentativa de injeção de códigos mal formados ou não autênticos e a melhoria na execução do SHA3 para o MSP430FR5994, onde foi possível concluir a eficácia das medidas adotadas.

Palavras-chave: Injeção de Código, CubeSat, Segurança das Comunicações, FreeRTOS.

ABSTRACT

Title: Secure protocol for code injection into cubesats
Author: Alexandre Henrique Radis
Supervisor: Daniel Chaves Café, Dr.
Co-Supervisor: João José Costa Gondim, Dr.
Graduate Program in Electronic and Automation Systems Engineering
Brasília, Jun 30th, 2023

A CubeSat-type satellite project starts with identifying your needs and continues with its development, assembly, launch, operation and obsolescence. However, needs can change over the lifecycle of the satellite, as with any project. In CubeSat-type satellites, the inclusion of new services becomes a major access challenge, due to the physical impossibility of the equipment.

Code injection is a solution that allows the inclusion of new services in a satellite after its launch. The inclusion of new services in microcontrolled equipment presents several security challenges, mainly in CubeSat-type satellites, which have restrictions on energy, communication, processing, memory, among others. It is necessary to protect the microcontrolled system against denial of service attacks, data breach, equipment deactivation and hijacking. It is not possible to use techniques such as firewall, antivirus or artificial intelligence.

As the inclusion of new services in the microcontrollers means the inclusion of new codes, and this means opening up a great opportunity for attacks. It is necessary to mitigate these attacks. Thus, the work presents a proposal for the inclusion of new codes mitigating the possibility of effective attacks. This proposal comprises security measures, communication protocols, the use of HMAC to ensure compliance and integrity of the new codes, and a real-time operating system prepared for this challenge.

The work presents a study of the state of the art and a bibliographic reference on the subject. The conceptual proposal follows, the methodology for implementing and testing the concepts, results and conclusions obtained. Among the results obtained, it was possible to observe the viability of the proposed measures, the defense of attempted injection attacks of malformed or non-authentic codes and improvement in the execution of SHA3 for the MSP430FR5994. Where it was possible to conclude the effectiveness of the adopted measures.

Keywords: Code Injection, CubeSat, Communications Security, FreeRTOS.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS E ABREVIA CÕES

DoS *Denial of Service* (Negação de serviços). 2, 16

HMAC *Hash-based Message Authentication Code*. 5, 11, 15, 25, 27, 35, 42, 44

IoT *Internet of Things*. 2, 6, 7

MPU módulo de proteção de memória. 8, 25, 44

NIST *National Institute of Standards and Technology*. 11, 27

NUPITEC Núcleo de Propriedade Intelectual. 4

PAD complemento de zeros até concluir o tamanho do bloco. 13

PGE Protocolo de Garantia de Entrega. 17, 18, 21, 33, 44

PGIA Protocolo de Garantia de Integridade e Autenticidade. 17, 18, 25, 33, 36, 38, 39, 41, 44

SAA *South Atlantic Anomaly*. 2, 8, 15, 32, 44

SHA3 *Secure Hash Algorithm version 3*. 4, 11, 15, 27, 35, 42, 44

1 INTRODUÇÃO

Em um projeto de desenvolvimento e lançamento de um satélite, são elaboradas várias premissas que irão nortear o seu desenvolvimento para a definição de objetivos, custo, tempo e demais características. Essas premissas são elaboradas na fase inicial, sendo adequadas às novas condições durante o seu período de desenvolvimento até o lançamento. Entretanto, as necessidades não param de mudar, seja pelo surgimento de novas ameaças ao projeto, seja pelo surgimento de novas oportunidades de negócio ou outras possibilidades. Assim, é interessante que o projeto de um satélite permita acompanhar essa evolução e possa ser reconfigurado para novas operações durante a sua vida útil.

A reconfiguração de um satélite pode ser algo simples como o redirecionamento das câmeras de captura de imagens ou a mudança de uma constante de cálculo de um sensor que pode ser feita por uma função que receba os parâmetros indicados. Outra possibilidade pode ser a inclusão de um novo serviço, mais complexo por necessitar da injeção de novos códigos na programação do sistema computacional do satélite. Essa mudança na programação do satélite é um processo delicado por envolver questões de segurança, principalmente quanto à integridade e autenticidade do novo código a ser injetado no sistema computacional.

A integridade do código é necessária para impedir que um código incompleto seja executado, podendo prejudicar o funcionamento do satélite, ou até mesmo a sua saída de operação. A autenticidade é necessária para impedir que códigos maliciosos enviados por terceiros sejam executados, podendo acarretar o mau funcionamento do satélite, espionagem e outros usos ou modificações indevidas das informações contidas no satélite, inclusive o sequestro e controle do satélite por terceiros. Esta garantia da integridade e autenticidade dos novos códigos são tratadas por processos cada vez mais complexos e de alto custo computacional. Entretanto, a não utilização dessas técnicas de segurança pode levar à perda do satélite e consequentemente, de todo o investimento no desenvolvimento, lançamento e operação do mesmo. Como o risco é muito elevado, não é possível pensar em injeção de código sem mecanismos de segurança.

Hoje em dia, os satélites chamados CubeSat são considerados de baixo custo (até 2 milhões) e orbitam o planeta em altitudes entre 500 e 1.800 km de altitude da crosta terrestre, fornecendo uma vasta gama de serviços, conforme (TOKUMITSU; TSUJI; NAKAYA, 2023). Por serem satélites de pequeno porte, possuem restrições de poder computacional, capacidade de comunicação e disponibilidade de energia, entre outras, precisando do uso racional dos seus recursos.

O sistema de comunicação entre a base e o satélite do tipo CubeSat tem algumas peculia-

ridades normalmente não encontradas em sistemas de *Internet of Things* (IoT) IoT - *Internet of Things* e outros sistemas de telemetria ou computacionais. Entre essas peculiaridades está a baixa taxa de transmissão/recepção, a alta perda de pacotes e curtos períodos de janelas de comunicação. Parte dessas peculiaridades aqui no Brasil pode ser explicado pela anomalia magnética de baixa intensidade, conhecida como Anomalia do Atlântico Sul - em inglês *South Atlantic Anomaly* (SAA) que dificulta essas comunicações.

No caso específico do projeto Alpha Crux (REIS et al., 2020), foi considerado para o projeto uma taxa de transmissão de 9.600 bits por segundos e uma taxa de perda de pacotes nas transmissões até 40% da base para o satélite e 20% do satélite para a base. Esses foram os valores considerados no projeto desenvolvido. E devido a sua órbita heliossíncrona, com altitude de 500 km para o caso estudado, a comunicação só é feita em janelas curtas de até 10 minutos devido ao período total nessa órbita ser de aproximadamente 100 minutos.

Um agravante é a órbita heliossíncrona tornar a comunicação entre a base e o satélite eficaz em poucas passagens. Causando uma ausência de comunicação que pode chegar a até 12 horas.

1.1 HISTÓRICO DE ATAQUES EM “CUBESATS”

O uso dos CubeSats possuem um histórico de ataques como os demonstrados no artigo "CubeSat Security Attack Tree Analysis"(FALCO; VISWANATHAN; SANTANGELO, 2021). Esse histórico inicia avaliando o caso da intrusão no ROSAT X-Ray Satellite ocorrido em 1998, passando pelos casos do Landsat-7 em 2008, RQ-170 Sentinel em 2011 e os casos do NASA Goddard Space Flight Center e NASA Johnson Space Center ocorridos em 2018. Pode ser observado no artigo que nem todos os ataques são diretamente feitos aos satélite, sendo que em alguns casos esses ataques são feitos às estações-base ou ao sistema de comunicação.

Os casos relatados de ataques e expostos em árvores de ataque são dos tipos:

1. *Denial of Service* (Negação de serviços) (DoS);
2. Adulteração de dados; e
3. Desabilitar o uso do satélite.

O ataque de negação de serviço visa a não disponibilização dos serviços fornecidos pelo equipamento e pode ser feito de várias formas para o presente caso, com o uso de equipamentos de telecomunicação mais potentes e/ou melhor localizados que a base, com o preenchimento dos canais de comunicação com “lixo”, ou solicitando a execução de serviços até a

exaustão do sistema alvo. Devemos observar que nesse caso, quando o ataque é finalizado, os serviços retornam a atividade normalmente, pois o seu objetivo é apenas a indisponibilidade do serviço, e não a sua desabilitação ou destruição.

O ataque de adulteração de dados tem o objetivo de obter as informações disponibilizadas pelo CubeSat ou alterá-las. Esse objetivo é muito interessante para o caso de obtenção de alguma vantagem sobre os demais usuários da informação, seja ela por saber primeiro, saber a informação correta, ou fazer essa informação ser melhor aos seus negócios. Um ponto a destacar é que a identificação desses ataques é demorada e difícil, pois não há nenhum indício de que o fluxo de informações foi interrompido.

No terceiro, ocorre disponibilidade dos serviços fornecidos pelo equipamento, o que traz prejuízo na indisponibilidade dos serviços fornecidos pelo satélite de forma permanente. Devemos observar que esses são apenas casos relatados em um único artigo referente a segurança de satélites de apoio aos sistemas da NASA, não explorando o uso de satélites comerciais ou mesmo satélites críticos da própria NASA. Esses relatos demonstram o interesse dos atacantes nesse tipo de equipamento. Dentro do contexto apresentado, o objetivo do presente trabalho é permitir a reconfiguração do satélite em órbita via injeção de código, protegendo esse mecanismo com técnicas de segurança adaptadas ao baixo poder computacional dos CubeSats.

Entretanto, à lista anterior, deve ser acrescida o relato publicado em 30 de abril de 2023 (BRANDAO, 2023) do 1º sequestro de controle de satélite por hackers. Nesse último caso, o atacante pode simplesmente passar a utilizar os serviços fornecidos do satélite, como também cobrar um valor para a devolução do controle do mesmo, e também em caso de equipamentos com propulsão, utilizá-lo como arma de ataque a outros equipamentos.

1.2 MOTIVAÇÃO

Embora existam vários métodos que visam garantir a integridade e autenticidade das transmissões de informação, as restrições de memória, processamento, disponibilidade de energia, baixas taxas de comunicação e pequenas janelas de comunicação, impedem a direta aplicação de técnicas muito elaboradas de segurança das comunicações. Dentre as técnicas de implementação proibitivas estão a implantação de barreira de segurança (*firewall*), antivírus e inteligência artificial para proteger o satélite de possíveis ataques. Portanto, é necessário o desenvolvimento de uma solução que atenda às necessidades e restrições impostas.

Assim, na expectativa de atacar o problema definido no subitem 1.1, este trabalho tem por principal motivação a adaptação de um sistema operacional de tempo real para o microcontrolador MSP430FR5994 da Texas Instruments com segurança suficiente para permitir

a injeção de novos códigos, mitigando a possibilidade dos ataques indicados nos subitens anteriores. Esta segurança prevê um protocolo para garantir a entrega dos pacotes entre o satélite e a base, um protocolo para garantir a integridade e autenticidade do código a ser injetado utilizando hash de autenticação baseado em *Secure Hash Algorithm version 3* (SHA3) adaptado para uso do módulo multiplicador de 32 bits do microcontrolador. Também é utilizado o módulo de proteção de memória do microcontrolador para eliminar a possibilidade de códigos dos novos serviço alterem o núcleo do sistema operacional.

1.3 DEFINIÇÃO DO PROBLEMA

Baseados nos históricos de ataques e os possíveis danos apresentados nos subitens anteriores, é temerária a possibilidade de injeção de código sem nenhum tipo de proteção ou segurança em algum tipo de satélite. Portanto, é necessário encontrar alternativas para que essa injeção de código seja feita apenas por quem está autorizado a fazê-lo, e que os códigos a serem injetados sejam íntegros e não danifiquem sem intenção o funcionamento do equipamento. Essa solução deve conseguir operar com as restrições comuns a CubeSats: baixa taxa de transmissão, baixo consumo de energia e curtas janelas de comunicação, e propiciar a garantia de integridade e autenticidade do código a ser injetado no satélite.

1.4 PUBLICAÇÕES

Na busca de encontrar solução para o problema, já foi publicado um trabalho referente a criação de dois protocolos de comunicação entre a base e o satélite para garantir a entrega dos pacotes durante as curtas janelas de comunicação, e que os novos serviços a serem injetados no satélite sejam autênticos e íntegros (RADIS; GONDIM; CAFÉ, 2022).

E também está em curso o processo de registro de software, junto ao Núcleo de Propriedade Intelectual (NUPITEC) da Universidade de Brasília, de um método para execução do SHA3-256 para o microcontrolador MSP430FR5994 que utiliza o módulo de multiplicação dedicados deste microcontrolador para rotação de bits, conseguindo reduzir em 35% o tempo de execução deste processo utilizado na autenticação dos novos serviços a serem injetados.

1.5 ORGANIZAÇÃO DO TRABALHO

Este trabalho começa com esta introdução para situar o leitor sobre as peculiaridades da injeção de novos serviços em satélites em operação. O gráfico abstrato da Figura 1.1 mostra

um resumo da solução, com a possibilidade legítima de incluir novas ideias. É possível visualizar a existência de comunicação entre a base e o satélite e uma sistemática de autenticação HMAC para validar a injeção de códigos legítimos. Mas caso ocorra uma injeção de código ilegítimo, o sistema de proteção de memória irá impedir alteração do sistema operacional e manter as funções básicas operacionais.

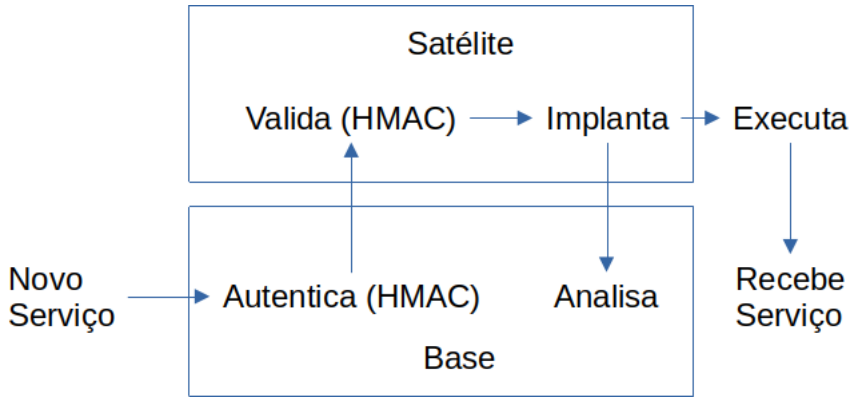

Figura 1.1 – Gráfico abstrato da solução

O trabalho segue com uma revisão bibliográfica sobre o tema e sobre os mecanismos utilizados na proposta de solução do problema. Tendo em sequência as proposições das medidas elaboradas para a injeção segura de novos serviços no satélite em órbita.

A implementação das medidas propostas e a infraestrutura experimental de testes está descrita no capítulo 4. Seguido dos resultados obtidos durante a execução dos testes. No último capítulo é dedicado um espaço para as conclusões e trabalhos futuros, seguido do referencial bibliográfico.

2 REVISÃO BIBLIOGRÁFICA

Os satélites são categorizados pela massa, como grandes satélites, microssatélites e nanossatélites (TOKUMITSU; TSUJI; NAKAYA, 2023). A categoria dos nanossatélites, que engloba os CubeSats, foi escolhida para o presente trabalho por fornecer serviços baseados em satélite e possuir custos menores de desenvolvimento, construção e de lançamento para órbitas. Sendo que os CubeSats podem proporcionar avanços e configurações inovadoras para a observação aeroespacial. Entretanto, essa configuração apresenta restrições de recursos e uso de energia, além de possuir um ciclo de desenvolvimento restrito e com baixo orçamento.

Para otimizar o uso dos recursos disponíveis, o atual trabalho propõe uma metodologia capaz de alterar os serviços fornecidos pelo CubeSat quando ele já estiver em órbita, permitindo a continuação do desenvolvimento do equipamento, mesmo após o seu lançamento. Esta perspectiva permite a injeção de código durante o período de operação do equipamento, em contrapartida, pode causar uma sensível diminuição na segurança do equipamento e sua comunicação. Para mitigar essa diminuição da segurança, a metodologia traz várias proposições para melhorar a segurança do processo, principalmente quanto às possibilidades de ataques para a obtenção do controle ou informações do satélite, levando em considerações as características de IoT do sistema e condicionantes específicas.

P. Yue et al. fizeram um estudo sobre as diferentes formas de ataque em satélites em geral (YUE et al., 2022). Segundo os autores, esses ataques podem ser do tipo: homem-no-meio (*Man-in-the-middle*), repetição (*replay*), obscurecimento, entre outros, além de citarem contra-medidas para proteger as informações e manter o satélite operacional. Outra preocupação no desenvolvimento de CubeSats é a economia de energia. I. Bisio et al. produziram um artigo comparando métodos de alocação de recursos em sistemas que possuem elevada perda de pacotes com foco em métricas de consumo de energia para satélites geoestacionários (BISIO et al., 2013). Segurança e consumo de energia são duas preocupações frequentemente compartilhadas com o universo de dispositivos IoT.

Em IoT existem várias preocupações com as vulnerabilidades como citados nos trabalhos de R. Ramadan (RAMADAN, 2022), P. Rahimi et al. (RAHIMI et al., 2021) e R. P. Lee et al. (LEE; MARKANTONAKIS; AKRAM, 2019), sendo nesses a preocupação maior com a confidencialidade. Os trabalhos de T. Hiscock et al. (HISCOCK; SAVRY; GOUBIN, 2019) e M. A. Philip et al. (PHILIP et al., 2017) demonstram a preocupação com a possibilidade de acesso físico às informações internas do sistema e o trabalho de S. Aruna et al. (ARUNA et al., 2020) trata dos ataques quanto a integridade de códigos. O trabalho de M. Hwang et al. (HWANG; YANG; SHIU, 2003) tem como propósito o uso de satélites para intercomunicação baseado na confidencialidade, bem como o trabalho de T. Ferrer et al. (FERRER;

CÉSPEDES; BECERRA, 2019) propõe o uso de nanosatélites em rede para comunicação com áreas remotas.

Encontramos uma orientação das tecnologias utilizadas em IoT no trabalho de M. Sinha e S Dutta (SINHA; DUTTA, 2021), onde são demonstrados vários comparativos entre técnicas de criptografia leve (*Ligthweight Cryptography*), apesar de apresentar técnicas de confidencialidade e não de integridade e autenticidade como estamos propondo. Outras orientações importantes estão nos tópicos que se sequem e foram utilizadas no desenvolvimento da metodologia.

2.1 TRABALHOS CORRELATOS

O trabalho (FEREIDOONI et al., 2017) faz uma análise na segurança dos dados de equipamentos médicos e demonstra que nenhum dos equipamentos testados possuem segurança para garantir que os dados não sejam alterados por injeção de códigos. Mantendo na área biomédica, o trabalho (KWARTENG; CEBE, 2022) faz uma análise comparativa da segurança entre equipamentos médicos implantados e demais sistemas embarcados.

Na área de IoT podemos ver no trabalho (TSAUR; CHANG; CHEN, 2022) a implementação segura de uma sistemática de substituição de todo o firmware de um sistema embarcado utilizando blockchain para múltiplos dispositivos. Em outro trabalho (WANG et al., 2022), podemos ver uma implementação de novo firmware baseada em autenticação com MD5. Podemos observar que ambos trabalhos utilizam a rede ethernet para a substituição integral do firmware dos dispositivos.

Os trabalhos da área biomédica estão mais preocupados com a autenticidade dos dados e não com a inclusão de novos serviços. Os trabalhos da área de IoT estão preocupados com a substituição integral do *firmware* do microcontrolador dos equipamentos de IoT. Sendo que no primeiro caso utiliza uma estrutura de blockchain que demanda comutação distribuída e não pode ser aliado ao caso dos CubeSats. E no segundo caso utiliza MD5 que possui fragilidades como demonstrada no trabalho (GUZMAN; SISON; MEDINA, 2018).

Assim, nenhum dos trabalhos está exposto às condições de comunicação e energia dos CubeSats, bem como não estão preparados para a inclusão de novos serviços sem a substituição completa do firmware do microcontrolador. Portanto, o presente trabalho se mostra atual e aplicável ao seu objetivo de injeção segura de novos serviços aos satélites do tipo CubeSats.

2.2 SAA - SOUTH ATLANTIC ANOMALY

A *South Atlantic Anomaly* é uma anomalia que ocorre em altitudes entre 200 e 800Km acima da crosta terrestre e se estende entre as latitudes 0º e 50ºS, e 90ºW e 40ºE (NASUDDIN; ABDULLAH; HAMID, 2019), interferindo nas comunicações via radiofrequência e causando danos aos equipamentos (WIKMAN; SJÖBLOM, 2017) (TAXONERA et al., 2019). O mais importante dessas interferências causadas por essa anomalia é o entendimento que ela causa perda de toda a transmissão em alguns períodos e depois as comunicações voltam a normalidade, chamadas "Perdas em Rajadas".

O artigo "The future of the South Atlantic anomaly and implications for radiation damage in space"(HEIRTZLER, 2002) traz uma perspectiva histórica e futura de como essa anomalia deve se comportar até 2100. O impacto desta anomalia nas comunicações dos satélites causa um incremento das perdas. Assim, as perdas nas transmissões de dados podem chegar a até 20% para o caso de download das informações e 40% para upload.

2.3 MÓDULO DE PROTEÇÃO DE MEMÓRIA

O microcontrolador MSP430FR5994 (TEXAS INSTRUMENTS, 2016) possui um módulo de proteção de memória (MPU) dedicado, este módulo tem duas funcionalidades. A primeira é proteger a propriedade intelectual, impedindo que uma possível intrusão no microcontrolador recupere os binários para ser feita engenharia reversa e descobrir os mecanismos programados no mesmo. A segunda é um módulo que define as operações que podem ser feitas em cada região de memória do microcontrolador, sendo que as três operações possíveis são as de leitura, escrita e execução. No presente trabalho utilizaremos apenas a última opção.

Conforme o *User Guide* (TEXAS INSTRUMENTS, 2012) e o *ApplicationReport* (TEXAS INSTRUMENTS, 2014), existem duas áreas de memória que o módulo de proteção de memória não atua. A primeira que inicia no endereço 0x00000 e termina no endereço 0x03FFF é a faixa de memória RAM. A segunda que inicia no endereço de memória 0x0FF80 e termina em 0x0FFFF que correspondem aos endereços das rotinas de tratamento de interrupção. Embora o mapeamento de memória do MSP430 seja de 20 bits, os endereços de memória para indicação das bordas são de apenas 16 bits, por esse motivo a indicação das bordas só podem ser feitos em passos mínimos de 16 unidades, por desconsiderar os 4 bits menos significativos do endereço de memória de 20 bits.

Este sistema de configuração nos permite a criação de três regiões de controle independente, conforme segue e demonstrado na figura 2.1:

1. começando no endereço 0x04000 e terminando no valor do registrador (MPUSEGB1-1);
2. começando valor do registrador MPUSEGB1 e terminando em (MPUSEGB2-1);
3. começando valor do registrador MPUSEGB2 e terminando no endereço 0xFFFFF;

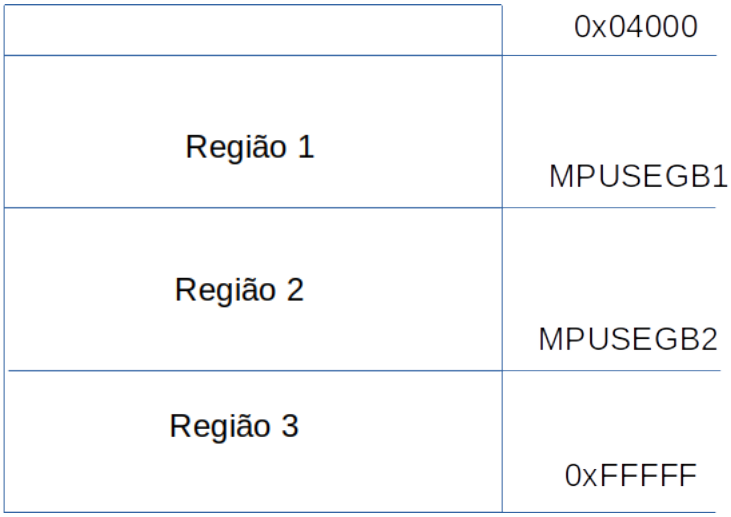

Figura 2.1 – Definição das regiões do Módulo de Proteção de Memória

Com as regiões definidas, o próximo passo é a definição das permissões para cada região, para isso o controlador possui o registrador MPUSAM com os bits específicos para cada permissão, onde x é a região de memória definida e pode assumir os valores 1, 2 e 3:

1. MPUSEGxXE - Permite execução e leitura independentemente do valor de xRE quando em 1 e não permite execução quando em 0;
2. MPUSEGxWE - Permite escrita e leitura independentemente do valor de xRE quando em 1 e não permite escrita quando em 0;
3. MPUSEGxRE - Permite leitura quando em 1 e não permite quando em 0, deve-se observar que caso alguma das opções anteriores for 1, esse bit não tem função, pois nesse caso é necessário a permissão de leitura.

Para concluir a configuração temos que a habilitar as interrupções para o caso de violação. Caso essas interrupções não sejam habilitadas as configurações anteriores são inócuas. Caso as interrupções sejam habilitadas, ocorre a interrupção de reinicialização do sistema. Essas interrupções são habilitadas no bit MPUSEGIE localizado no registrador MPUCTL0, sendo que as indicações de interrupções ficam no registrador MPUCTL1. O registrador MPUCTL0 possui dois bits que permitem a configuração do módulo de proteção de memória, o bit MPUENA habilita a configuração e o bit MPULOCK bloqueia a configuração, observar que após esse último passar para bloqueado, não pode mais ser alterado.

2.4 MÓDULO DE MULTIPLICAÇÃO DE 32 BITS

Outro importante módulo do MSP430FR5994 (TEXAS INSTRUMENTS, 2016) para a realização do atual projeto, é o módulo de multiplicação de até 32 bits. Este módulo é capaz de fazer a multiplicação de dois números de 32 bits formando um número de 64 bits. Para um processador de 16 bits, ele pode agilizar muitos cálculos e facilitar outras operações que possam se utilizar da multiplicação, como, por exemplo, a rotação de bits, sendo que a funcionalidade nativa de rotação de bits do microcontrolador rotaciona apenas um bit por ciclo de clock.

Tendo em vistas, que o módulo de multiplicação opera com multiplicações atômicas de 16 bits, formando números de 32 bits e fazendo a soma desses resultados no resultado final, podemos utilizar o seguinte modelo matemático:

$$A_{32} * B_{32} = (AH_{16} * 2^{16} + AL_{16}) * (BH_{16} * 2^{16} + BL_{16}) \tag{2.1}$$

Resultando em 4 operações de multiplicação, 3 operações de soma e 4 operações de atribuição, conforme demonstrado na extrapolação abaixo, condizente com a indicação de 11 ciclos de clock informados no User Guide (TEXAS INSTRUMENTS, 2012):

$$(AH_{16} * BH_{16}) * 2^{32} + (AL_{16} * BH_{16}) * 2^{16} + (AH_{16} * BL_{16}) * 2^{16} + AL_{16} * BL_{16} \tag{2.2}$$

Entretanto, como a operação de multiplicação inicia ao ser gravado o valor de BL, e o resultado de RES0 é a parte menos significativa do valor de AL*BL podemos utilizar os seguintes passo:

1. Gravamos AL em MPY32L;
2. Gravamos AH em MPY32H;
3. Gravamos BL em OP2L;
4. Gravamos BH em OP2H;
5. Espera-se 5 ciclos de clock;
6. Lemos o valor de RES0, menos significativo;
7. Lemos o valor de RES1;
8. Lemos o valor de RES2;
9. Lemos o valor de RES3, mais significativo;

2.5 HMAC COM SHA3

Os códigos de autenticação de mensagens baseado em hash - *Hash-based Message Authentication Code* (HMAC) são um subgrupo de funções de códigos de autenticação de mensagens, o livro "Criptografia e Segurança de Redes"(STALLINGS, 2015) traz uma boa explanação sobre o assunto e outras opções para autenticação de mensagens. A construção do HMAC padronizada pelo *National Institute of Standards and Technology* (NIST) com base nos trabalhos de Bellare (BELLARE; CANETTI; KRAWCZYK, 1996a) e (BELLARE; CANETTI; KRAWCZYK, 1996b), e opera conforme a fórmula 2.3.

$$HMAC(K, M) = H[(K^{+} \oplus opad)||H[(K^{+} \oplus ipad)||M]] \tag{2.3}$$

Onde:

- $\oplus$ é um XOR bit-a-bit;
- || é a concatenação de valores;
- K^{+} é a senha completada por zeros até o tamanho de bits do hash;
- ipad é o número 0x5C repetido o tamanho de bits do hash dividido por 8;
- opad é o número 0x36 repetido o tamanho de bits do hash dividido por 8;
- M é a mensagem a ser autenticada; e
- H é a função hash a ser utilizada.

No diagrama 2.2, podemos visualizar uma representação simplificada do processo de geração de um hash HMAC utilizando o SHA3, explicado na sequência, como o processo de geração do hash, onde podemos observar a indicação de valores específicos para a operação XOR com a senha e a concatenação deste resultado com a mensagem e passagem dessa cadeia de caracteres pelo primeiro processo de hash. O resultado do primeiro hash é concatenado com o resultado do XOR da senha com outro número específico e executado um segundo processo de hash, resultando no hash de assinatura.

O objetivo é um processo que não possa ser invertido, e por esse motivo é importante que o processo de geração de hash seja confiável e não possa ser feita engenharia reversa. No atual projeto foi escolhido o SHA3-256 que foi aprovado pelo NIST como padrão de hash confiável e é indicado pelo mesmo instituto como o padrão para geração de hash confiável.

A primitiva de hash SHA3 foi padronizada pelo NIST para uso compatível com o SHA2 e foi objeto de concurso conforme explanado no trabalho de Christof Paar e Jan Pelzl (PAAR;

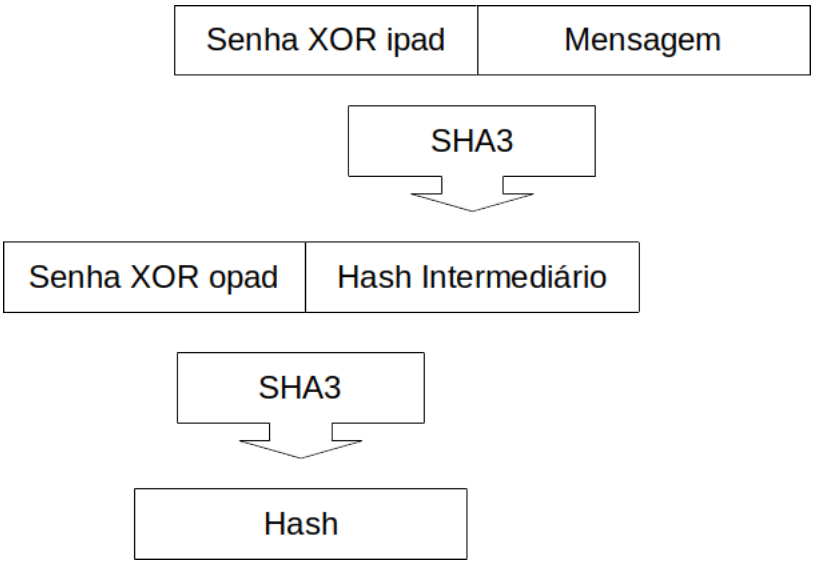

Figura 2.2 – Diagrama de processamento do HMAC com o SHA3

PELZL, 2010), sendo que o método vencedor foi o Keccak de Guido Bertoni, Joan Daemen, Michaël Peetersand Gilles Van Assche.

O funcionamento do Keccak baseia-se em uma estratégia de absorção e liberação das informações em rounds, onde primeiro se absorve todas as informações e depois se liberam as informações desejadas nas quantidades de rounds necessárias, conforme a figura 2.3, onde 'x' corresponde a entrada e 'y' corresponde ao hash de saída.

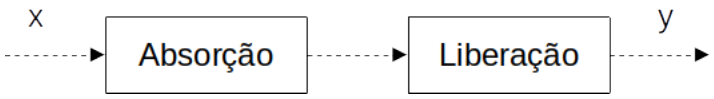

Figura 2.3 – SHA3 Diagrama de Alto Nível

Tanto a absorção como a liberação são feitas em vários passos internos, dependendo da quantidade de bits de entrada e de saída. A figura 2.4 demonstra o processo de absorção e liberação para o caso de n bits de entrada e m bits de saída.

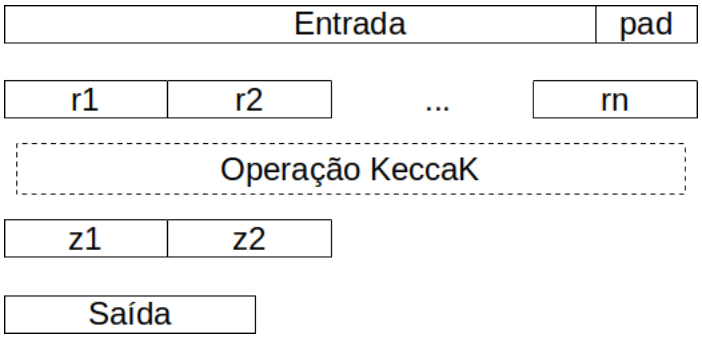

Figura 2.4 – SHA3 Diagrama de Entrada e Saída

Onde:

- Entrada - itens de entrada;
- PAD - complemento de zeros até concluir o tamanho do bloco (PAD);
- r - bits do bloco de entrada acrescidos do PAD e divididos em porções do tamanho do hash, r1, r2, ..., rn;
- z - bits do bloco de saída, divididos em porções do tamanho do hash, z1 e z2;
- n - quantidade de blocos necessários para cobrir toda a entrada;
- pad - bits a serem complementados para satisfazer r * n;
- Saída - hash definido para saída.

O Keccak se baseia em um cubo de 5 x 5 x Palavra que faz o embaralhamento dos bits, é conhecido como Keccak-f e é composto de 24 repetições dos 5 processos a seguir, com a nomenclatura padrão para o Keccak na literatura:

θ - Substituição de valores dos bits com os da coluna anterior, atual e posterior;

ρ - Permutação circular da palavra;

π - Permutação na matriz 5x5;

χ - Substituição entre os bits de linhas diferentes;

ι - Substituição dos bits iniciais por constante da rodada.

Os passos θ e ι, devem ser o primeiro e último passo a serem feitos respectivamente. e os passos ρ, π e χ não precisam respeitar ordem e são independentes, entretanto esses passos não podem ser feitos em paralelo por interferirem entre si.

2.6 FREERTOS

O FreeRTOS(FREERTOS..., 2023) é um sistema operacional de tempo real desenvolvido para o gerenciamento de microcontroladores com o objetivo de gerenciar multitarefas em tempo real. Para o gerenciamento das tarefas, o FreeRTOS utiliza o modelo preemptivo aliado a priorização das tarefas, onde os programas devem permitir que sejam interrompidas e retomadas a qualquer tempo (preemptivo) e são executadas conforme a sua prioridade. Se as tarefas forem chaveadas rápidas o suficiente, há uma impressão que as tarefas são executadas em paralelo.

Este modelo, visa garantir que as tarefas de maior prioridade sempre sejam executadas primeiro, com o objetivo de impedir que as tarefas mais prioritárias não sejam "atrapalhadas"pelas tarefas menos prioritárias. Por outro lado, caso as tarefas mais prioritárias tenham

um uso muito intenso do processador, as tarefas menos prioritárias não terão oportunidade de serem executadas. Existem alguns mecanismos que podem permitir que tarefas menos prioritárias possam ser executadas, mesmo com uso intenso do processador. Dentre esses mecanismos, está a escalação de prioridade de tarefas que não foram executadas nos prazos devidos.

Para o entendimento desse modelo, podemos utilizar como base o livro "Sistemas operacionais de tempo real e sua aplicação em sistemas embarcados"(DENARDIN; BARRIQUELLO, 2019). Entre os conceitos principais estão os de interrupção, preempção, "ticks"de temporização e prioridades, como segue:

- Interrupção: capacidade do microcontrolador interromper a execução de uma atividade em virtude da ocorrência de algum evento, seja um evento externo ao microcontrolador, como a mudança de estado de um pino, seja um evento interno como a conclusão de uma temporização;
- Preempção: capacidade de uma função poder ser interrompida durante a sua execução e concluída posteriormente, sem sofrer alteração do seu resultado. Por exemplo, não ter seus contadores alterados;
- Ticks: Períodos de tempos em que o sistema operacional libera a execução da tarefa;
- Prioridade: A prioridade determina a importância de uma tarefa para o atendimento dos serviços fornecidos pelo sistema.

Com esses 4 conceitos, é possível entender o funcionamento de uma maneira resumida de como e quando o sistema operacional - SO, decide qual a tarefa que irá ser executada em cada momento seguindo os seguintes passos:

1. Um temporizador é configurado para o período de um tick;
2. Ao final de cada período é gerada uma interrupção, o SO suspende a execução da tarefa em execução e escolhe a tarefa com maior grau de prioridade para ser executada, que pode ou não ser a mesma;
3. Caso a tarefa escolhida não seja a mesma que já estava em execução, a que estava em execução ficará paralisada até ser liberada para execução pelo SO;
4. No caso de ocorrência de uma interrupção, a execução da tarefa será paralisada, será executada a tarefa definida para a interrupção e depois a tarefa que foi paralisada voltará para a sua execução normalmente.

2.7 RESUMO

Este capítulo começou com a análise do uso dos CubeSats e a necessidade de inclusão de novos serviços, depois analisou trabalhos correlatos indicando a atualidade e oportunidade do assunto, e também a condicionante relativa a *South Atlantic Anomaly* (SAA). Na sequência foram analisados os módulos de proteção de memória e de multiplicação de 32 bits do MSP430FR5994, o funcionamento do HMAC com SHA3 e do FreeRTOS que serão utilizados para algumas soluções deste projeto.

No próximo capítulo será proposto uma solução para o problema foco do trabalho. O capítulo 4 irá apresentar as soluções técnicas para a implantação da solução e a infraestrutura de testes. Os resultados encontrados estão no capítulo subsequente, deixando as conclusões para o capítulo 6.

3 PROPOSIÇÃO

A injeção de código é o processo de modificação da funcionalidade do satélite após o lançamento do mesmo, isso se faz necessário, pois projetos de CubeSats possuem um ciclo de desenvolvimento menor que dos satélites tradicionais e devido as novas ideias que surgem enquanto o equipamento está em órbita, é comum haver modificações de requisitos após o lançamento do satélite. Entretanto, essa injeção de código traz várias implicações de segurança, desde a simples injeção de um código mal formado que pode inviabilizar a operação do CubeSat até o uso indevido e malicioso. Por esse motivo é necessário proteger a operação de injeção de código, pois apesar dela oferecer uma versatilidade maior ao ciclo de vida do equipamento e das funcionalidades do satélite, ela também traz consigo os perigo mencionados. Assim, para mitigar essas ameaças, primeiro precisamos conhecê-las e depois criarmos medidas para mitigar essas ameaças como segue.

3.1 MODELO DE AMEAÇA

Na introdução, além das ameaças citadas na lista 1.1, consta menção ao trabalho (BRANDAO, 2023) formando a seguinte lista de ameaças:

1. *Denial of Service* (Negação de serviços) (DoS);
2. Adulteração de dados;
3. Desabilitar o uso do satélite; e
4. Sequestro do satélite.

Os ataques do tipo Negação de Serviço podem ser feitos de duas formas para a situação dos CubeSats. A primeira, simplesmente com a inclusão de outros rádios com potências maiores que inviabilizem as comunicações, esse tipo de ataque é chamado de ‘jamming” e suas contra medidas estão fora do alcance deste projeto. A segunda opção de ataque é simplesmente outro rádio ficar enviando códigos ao satélite que faça ele ficar fazendo processamentos previstos, mas que sobrecarreguem o atacado e impossibilite a execução para obtenção dos serviços esperados, um exemplo simples para isso, seria o caso de enviar muitos e-mails para um servidor a ponto de ele começar a perder esses e-mails.

A adulteração de dados pode ser feita com a inclusão de códigos maliciosos no equipamento atacado que altere as informações enviadas a base. Já a desabilitação do uso do

satélite pode ser feito com a inclusão de códigos maliciosos que promovam, por exemplo, reinícios sequenciais do sistema para não permitir que o sistema faça o que está programado a fazer. Entretanto, o caso de desabilitação do uso do satélite pode ocorrer apenas por problemas de programação, ou a implantação de códigos em que a integridade foi comprometida, intencionalmente ou não. O último caso, sequestro do satélite, é um pouco mais elaborado, pois além da inclusão de códigos maliciosos, esses códigos precisam roubar o controle do satélite, passando o controle efetivamente para o atacante.

O presente projeto pretende propor medidas e protocolos para mitigar a possibilidade de ataques do tipo, adulteração de dados, desabilitação do satélite e sequestro do mesmo. Sendo que a defesa quanto aos ataques de negação de serviço não estão no escopo do presente trabalho.

3.2 MEDIDAS PROPOSTAS

Para atender as necessidades da injeção de novos serviços apresentados e fazer a defesa das ameaças estudadas, o presente trabalho propõe as medidas a seguir para efetivamente injetar novos serviços no microcontrolador do satélite mitigando as ameaças consideradas.

1. Determinação das comunicações apenas nas janelas com maior período;
2. Garantir que as comunicações seja efetivas durante o período de comunicação determinado;
3. Garantir que os novos serviços a serem incluídos sejam íntegros e autênticos; e
4. Não permitir que o núcleo do sistema operacional do microcontrolador possa ser alterado pelos novos serviços.

O item 1, tem o objetivo de impedir que rádios localizados em locais fora do perímetro definido, possam efetivar comunicações indevidas com o satélite, mitigando assim, a possibilidade de ataques do tipo DoS e tentativas de injetar códigos maliciosos no satélite. Para atender aos itens 2 e 3 são propostos dois protocolos: Protocolo de Garantia de Entrega (PGE) e Protocolo de Garantia de Integridade e Autenticidade (PGIA). O item 4, objetiva o que pode ser considerado o pior cenário, a quebra da senha de comunicação entre a base e o satélite, pois caso isso ocorra o atacante poderia incluir qualquer código no equipamento. Entretanto, mesmo que essa quebra de senha ocorra, o atacante não poderá alterar o núcleo do sistema operacional, abrindo possibilidade de retomada do satélite por parte da base.

O PGIA tem por finalidade garantir que o código a ser implementado seja exatamente o mesmo código que foi desenvolvido e testado na base. Em outras palavras, deve garantir que

não ocorreram alterações durante a transmissão, chamada de integridade, e que o código a ser injetado foi enviado pela base, chamado de autenticidade. Para esse propósito é utilizado um hash HMAC explanados em 2.5 com senha para validação do código recebido quanto a sua integridade e autenticidades.

Para a implementação do PGIA temos restrições de consumo de energia e do link de comunicação entre a base e o satélite (9600bps), além do processamento do HMAC ser dispendioso computacionalmente. Assim, nós optamos por separar as fases de processamento do PGIA das fases de transmissão dos dados. Pois como os períodos das janelas de comunicações são curtas e o CubeSat normalmente está sobrecarregado, deixamos esses períodos para garantir a entrega dos pacotes durante as janelas de comunicação, criando o Protocolo de Garantia de Entrega (PGE), conforme 3.4.

O PGE foi criado com a incumbência de garantir que as informações sejam efetivamente transferidas entre a base e o satélite, esse procedimento inicia na abertura da janela de comunicações e busca transmitir todos os dados necessários entre os dois elementos do sistema. O seu funcionamento é muito simples. O protocolo exige um retorno do receptor para toda mensagem enviada. Se o receptor não enviar uma confirmação, o transmissor repete o envio até que a confirmação do recebimento pelo receptor dos dados enquanto estiver na janela de comunicação.

Com o PGE garantindo que os dados são transmitidos, cabe ao PGIA o tratamento desses dados com o objetivo de garantir a integridade e autenticidade do código recebido pelo CubeSat da base e disponibilizá-lo para o processo de injeção. Com novo código disponibilizado para injeção, cabe ao satélite parar e destruir possíveis serviços que estejam alocados na região de memória destinada a novos serviços e injetá-los nesta região de memória, além de inicializar o serviço.

3.3 PGIA - PROTOCOLO DE GARANTIA DE INTEGRIDADE E AUTENTICIDADE

Para identificar a função de cada pacote recebido, utilizaremos 1 byte inicial conforme indicado na tabela 3.1. Onde a parte mais significativa do byte corresponde ao código do sistema (sistema atual, código 1) e o menos significativo relativo à função do pacote no sistema.

Deve-se observar que essa codificação permite a aplicação da ideia a outros processos, podendo ter, a princípio, 16 sistemas diferentes, com até 16 códigos em cada sistema, a serem alterados conforme as necessidades de cada projeto. Entretanto, visando criar códigos genéricos e agregadores, o sistema 0 será utilizado para esses códigos. Assim, o código

00 deve ser utilizado como um agregador de vários códigos simples de validações que não tenham argumentos, como, por exemplo: 00132C.

- 00 - Significa agregador, cada item posterior deve ser avaliado separadamente;
- 13 - Validação OK, conforme tabela 3.1;
- 2C - Significa alguma validação com base em lista de códigos a ser criada para atender algum futuro sistema.

O funcionamento do protocolo de integridade e de autenticidade é formado conforme a figura 3.1 da seguinte forma:

1. A base envia o binário (10) e o hash (11) para o satélite;
2. No período entre janelas o satélite valida o binário com o hash, o satélite tem 12 horas para essas validações;
3. A base envia confirmação de envio do hash, códigos 12, essa confirmação é importante por permitir uma janela de 12 horas para identificar possíveis erros ou desistência no código enviado;
4. O satélite envia o resultado das validações, códigos 13 ou 14;
5. No período entre janelas, no caso do satélite ter validações positivas (Códigos 12 e 13) implementa o código, caso contrário prepara envio de erro para a base;
6. Satélite envia para base a operação de sucesso na implementação (15), código de falha na implementação (16), ou código de não implementado (17). No caso de código 17, envia o binário e o hash para possível análise;
7. Passo existente apenas no caso de não implementação (16): A base analisa o binário e o hash recebido na busca de possível ataque.

Tabela 3.1 – Lista de Códigos do Protocolo de Garantia de Integridade e Autenticidade

Código (hex)	Função
10	Binário a ser injetado
11	Hash
12	Confirma o envio
13	Validação OK
14	Validação Falhou
15	Implementação Ok
16	Falha ao implementar
17	Não implementado

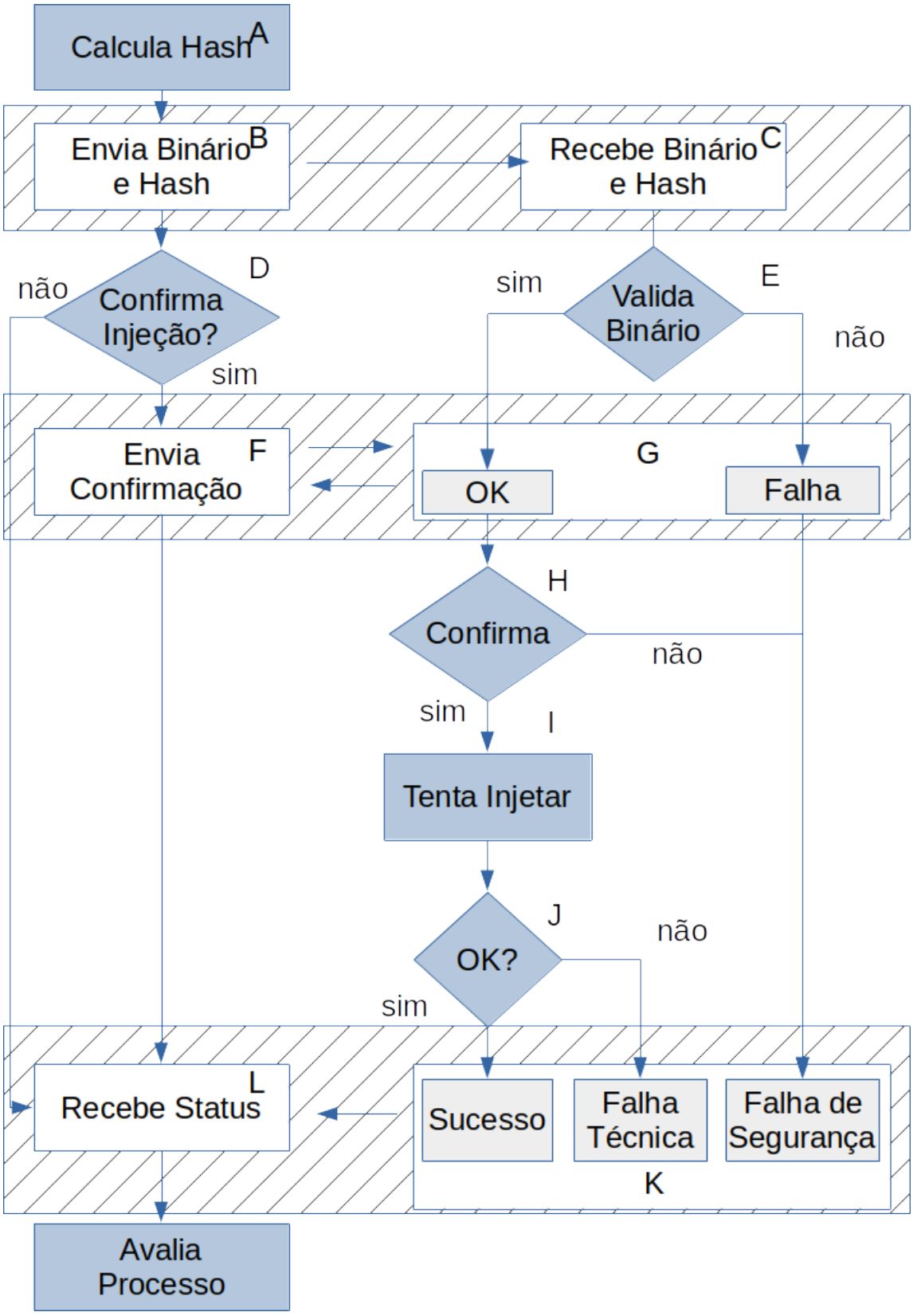

Figura 3.1 – Fluxo do Protocolo de Garantia de Integridade e Autenticidade

3.4 PGE - PROTOCOLO DE GARANTIA DE ENTREGA

Devido às peculiaridades do contexto de operação, é interessante estabelecer um protocolo de comunicação que garanta a entrega dos pacotes e depois um protocolo que garanta que o código enviado pela base seja íntegro, bem como a autenticação da base como emissora do código a ser utilizado.

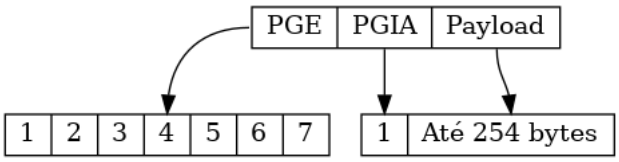

Figura 3.2 – Bytes do Protocolo de Garantia de Entrega

Para a elaboração do PGE, fizemos o estudo matemático da Tabela 3.2, englobando os 7 bytes de cabeçalho, como proposto na figura 3.2. Consideramos um bloco de 2560 bytes de informação a serem transmitidos num cenário de 40% de perdas na transmissão da base para o satélite e 20% na transmissão do satélite para a base. As perdas não foram consideradas em situações de reenvio. Foi considerado também um tempo entre os pacotes de 1 milissegundo para evitar colisão e sobreposição. Também consideramos 20 milissegundos como tempo entre o satélite receber o bloco completo, analisar e responder com uma confirmação (acknowledge) ou início de retransmissão, se necessário.

$$T_0 = (C/p) * 7 + C \tag{3.1}$$

O total de bits iniciais (T_0) a ser transmitido é igual ao bloco de bytes (C) a serem transmitidos acrecido da quantidade de cabeçalhos (C/p) multiplicado pelos bytes de cabeçalho (7), onde (p) corresponde ao tamanho máximo do pacote.

$$Q_0 = T_0/p \tag{3.2}$$

A quantidade de blocos de envio inicial (Q_0) a ser transmitido ao Total de bytes inicial (T_0) a serem transmitidos dividido pelo tamanho do pacote (p);

$$t_1 = (T_0 * 8)/9800 + Q_0/1000 + 0,02 \tag{3.3}$$

O tempo total da primeira transmissão (t_1) corresponde ao total de bytes iniciais (T_0) multiplicado pelo número de bits da palavra (8), acrescido da quantidade de blocos de envio inicial (Q_0) multiplicado por 1 milissegundo e os 20 milissegundos reservados para a resposta.

$$Q_1 = Q_0 * 0.4 \tag{3.4}$$

As falhas no caso de envio da base para o satélite correspondem a 40% dos pacotes.

$$t_2 = (p * Q_1 * 8)/9800 + p/1000 \tag{3.5}$$

O tempo total da primeira transmissão (t_2) corresponde ao total de bytes restantes ($Q_1 * p$) multiplicado pelo número de bits da palavra (8), acrescido da quantidade de blocos de envio inicial (Q_0) multiplicado por 1 milissegundo.

Na tabela 3.2, a coluna "Tamanho" representa o tamanho do pacote e varia de 512 bytes a 32 bytes. Já a coluna "Quantidade" ilustra a quantidade de pacotes resultantes da divisão entre o tamanho total do bloco pelo tamanho do pacote. Aqui consideramos os bytes do pacote acrescidos dos bytes de cabeçalho. A coluna "Tempo (sec)" mostra o tempo total de transmissão considerando os pacotes, o cabeçalho e o tempo de intervalo como citado no parágrafo anterior.

Tabela 3.2 – Estudo de transmissão de 2560 bytes em pacotes de diferentes tamanhos num cenário com perdas

Tamanho	**Quantidade**	**Tempo (sec)**
512	6	3,44
256	11	3,24
128	22	3,35
64	45	3,44
32	98	3,86

Nota-se, pela tabela, que o pacote de 256 bytes é o mais eficiente. Uma vantagem secundária de utilizar esse tamanho é que ele pode ser representado em 1 byte.

O motivo de considerarmos um cabeçalho de 7 bytes para cada pacote é a configuração de um cabeçalho útil, inclusive para uma possível internacionalização do protocolo. Assim, o cabeçalho foi idealizado como segue:

1º byte: Código identificador do país nas ligações internacionais;

2º e 3º bytes: Código identificador do equipamento do país, até 65535;

4º byte: Número do pacote na janela;

5º byte: CheckSum do pacote;

6º byte: Tamanho do pacote.

7º byte: Agregação de pacotes: A parte mais significativa representa o número do sub-pacote e a parte menos significativa denota o número total de sub-pacotes;

Com essas configurações, o satélite pode identificar nos três primeiros bytes se a mensagem é para ele, podendo economizar energia enquanto espera uma mensagem que lhe seja direcionada, algo próximo ao que existe hoje no protocolo IP - Internet Protocol. Observar que diferente do protocolo IP, o objetivo do protocolo proposto é a comunicação base-satélite, e não a criação de uma rede de satélites. Portanto, uma base pode se comunicar com vários satélites, mas não está prevista comunicação entre os satélites.

Os pacotes 4 a 6 tem a finalidade de gerenciamento do pacote na janela e garantir a integridade dos pacotes recebidos, com o prévio descarte do mesmo caso o tamanho ou o checksum não sejam atendidos, evitando o recebimento de pacotes quebrados e minimizando o uso de processamento desnecessário. O 7º byte tem a função de indicar a necessidade de concatenação de pacotes para a formação de sequências de bytes maiores que 256 bytes. Observar que esse valor é independente do número do pacote indicado no byte 4, entretanto esses pacotes devem ter numeração sequencial para impedir confusões, seja por problemas de transmissão, seja por interferência de um atacante.

3.5 RESUMO

No presente capítulo foram descritas 4 medidas que visam garantir que seja possível a inclusão de novos serviço no satélite mitigando a possibilidade de injeção de códigos maliciosos ou mal formados que possa causar danos ao funcionamento do satélite. No próximo capítulo do trabalho são apresentados os aspectos técnicos para a sua implementação e a infraestrutura de testes montada para a evolução do trabalho. Na sequência, são apresentados os resultados obtidos, seguido do capítulo com as conclusões, terminando com o referencial bibliográfico.

4 METODOLOGIA

A injeção de novos serviços apresenta vários desafios como apresentados nos capítulos anteriores, principalmente por abrir portas para a injeção de códigos maliciosos por terceiros. O caso de injeção de código tem também a complexidade de integração do novo serviço a ser fornecido pelo CubeSat com os demais serviços já disponíveis no equipamento, como o sistema de comunicação com a base ou a telemetria do próprio satélite. Isso se deve ao fato de que sistemas embarcados não possuem a facilidade de sistemas operacionais, como acesso a um sistema de arquivos. Se fosse o caso, bastaria carregar um arquivo no satélite com um script ou um código compilado para executar a nova funcionalidade. Sistemas embarcados mais enxutos não possuem essas facilidades. Todo o código deve ser compilado e carregado antes do lançamento do satélite. Dessa forma, os endereços das funções são fixos na memória do microcontrolador. Assim, o código injetado precisa se adaptar à situação em que o microcontrolador foi gravado antes do lançamento.

Resumindo, temos como requisito principal para o sistema a proteção quanto a inclusão de códigos não autorizados ou mal formada e como requisito secundário a unificação do processo de comunicação visando/diminuir a superfície de ataque. A esses requisitos, somam-se as seguintes métricas para atender as necessidades do CubeSat:

- Eficácia na transmissão dos dados da base para o satélite dentro da janela;
- Eficácia na transmissão de dados do satélite para a base dentro da janela; e
- Eficiência na execução dos hashs de garantia de integridade e autenticidade.

O cuidado para a injeção de código deve iniciar já na geração do binário principal do satélite: ele será a base para a elaboração e testes dos novos serviços. Assim, no desenvolvimento do binário principal é interessante, que seja prevista uma área para os novos serviços, onde possam ser gravados e executados os novos códigos. E para atender essa necessidade foi utilizado o módulo de proteção de memória citados nos capítulos 2.3, como segue:

1. Região de memória para a leitura e escrita dos dados, memória RAM;
2. Região de memória para leitura, escrita e execução dos novos códigos; e
3. Região de memória para leitura e execução dos códigos do sistema operacional do satélite que não podem ser alterados.

O item 3 é uma barreira contra interferência indevida dos novos serviços no sistema operacional do satélite. E após a geração do binário principal e sua injeção no controlador do satélite, é necessário guardar uma cópia para o desenvolvimento futuro de novas ferramentas.

Quando for definido um novo serviço a ser executado pelo satélite, o desenvolvedor deve gerar uma nova cópia para o desenvolvimento e testes das novas funcionalidades observando os cuidados necessários citados no capítulo 4.1.2. Após a fase de desenvolvimento e testes das novas funcionalidades o desenvolvedor deve gerar um novo binário e extrair apenas os códigos da região de memória destinadas aos novos serviços a serem incluídos, esse código extraído será o binário enviado pela base para ser transferido pelo Protocolo de Garantia de Integridade e Autenticidade (PGIA) explanado no capítulo 3.3 e injetado no satélite.

Para a análise do sistema proposto foi montada uma infraestrutura de testes que será descrito na secção 4.3, onde foi possível incluir o monitoramento dos pacotes transmitidos entre os elementos do sistema e as falhas inerentes aos processos de transmissão de rádio comunicação descritas em 4.3.1, inclusive referente a SAA.

4.1 CASO DE USO

O caso de uso deste sistema operacional é de um satélite do tipo CubeSat de baixa órbita, com pequenas janelas de comunicação para a transferência de dados, baixa taxa de transmissão, necessidade de inclusão de novos serviços durante a sua vida útil e baixo consumo de energia. Para isso foi selecionado um microcontrolador MSP430FR5994 que trabalha com baixo consumo de energia e possui alguns itens importantes de segurança como o módulo de proteção de memória (MPU) onde é possível definir os níveis de acesso a regiões de memórias, e o Modulo de Multiplicação de 32 bits - MPY para agilizar o processamento do Hash HMAC.

4.1.1 Proteção de memória

O MSP430FR5994 possui um módulo de proteção de memória. Este módulo permite a definição das possibilidades de operações (leitura, escrita e execução) em regiões de memória superiores a 0x03FFF. A primeira parte do processo de definição da proteção de memória é definir três regiões: inferior, central e superior, com base em dois delimitadores. A segunda parte é a simples definição das possibilidades de operação para cada região.

Para o presente projeto, foram definidos os delimitadores em 0x0F000 e 0x10000, a parte inferior foi definida como leitura e escrita, a parte central foi definida como leitura, escrita e execução e a parte superior foi definida como leitura e execução, conforme figura 4.1.

NA	TINYRAM e INFOs	
	Constantes e Variáveis	0x01C00
LE	Constantes e Variáveis	0x04000
LEX	Códigos de interrupções	0x0F000
	Novos Serviços	0x0F088
NA	Marcadores de interrupção	0x0FF80
LX	Sistema Operacional	0x10000

Figura 4.1 – Mapa do módulo de proteção de memória implantado

Onde:

- NA - Não aplicável, conforme 2.3;
- LE - Leitura e escrita;
- LEX - Leitura, escrita e execução; e
- LX - Leitura e execução.

Essa decisão foi tomada para garantir que o código a ser injetado tenha uma área a ser escrita (parte central), mas não possa sobrescrever o código principal que fica protegido de escrita na área superior, aumentando assim, a proteção do sistema quanto a injeção de códigos maliciosos ou mal formados.

4.1.2 Cuidados dos novos serviços no MSP430

A inclusão de novos serviços no MSP430 precisa de muitos cuidados, isso ocorre em virtude da organização de memória do microcontrolador, pois o armazenamento dos códigos, variáveis e constantes não são sequenciais, não permitindo, por exemplo, que os novos códigos utilizem constantes que não estavam previstas no código inicial, principalmente as sequências de constantes (array de char, por exemplo). Outro ponto que deve ser cuidadosamente garantido é o endereçamento das funções, pois os novos serviços precisam saber exatamente os endereços de memória das demais funções que irão utilizar.

A ferramenta de desenvolvimento da Texas Instruments possui um mapeamento de memória que permite a indicação dos endereços de memória onde serão gravados cada tipo de informação e subdivisões: constantes, executável, variáveis, etc. Assim, foi possível a definição de constantes e variáveis na região da memória RAM, os novos serviços em uma região

que permite escrita e execução e o sistema operacional em uma região que não permite a escrita, conforme a figura 4.1

Com essas definições, foi possível atribuir aos novos serviços a serem inseridos o endereço de memórias 0x0F000 com espaçamento para desenvolvimento de 0x00F80 bytes a serem inseridos, isso ocorre pelo fato dos espaços de memória entre 0x0FF80 e 0x0FFFF serem destinados a variáveis do sistema e não estarem sujeitas as restrições do módulo de proteção de memória.

4.2 EFICIÊNCIA NA EXECUÇÃO DO SHA3

O HMAC é a base da segurança do sistema de garantia de integridade e autenticidade. O hash HMAC enviado pela base deve ser comparado com o gerado pelo satélite. Caso essas duas informações sejam idênticas, o código binário é considerado integro e autêntico. No atual projeto, o compartilhamento de senha de autenticação entre a base e o satélite deve ser feito com o satélite ainda em solo, permanecendo fixa durante toda a vida útil do satélite, este compartilhamento é feito com a inserção da senha no próprio código gerado para o satélite e armazenado na base.

Para o funcionamento do HMAC foi escolhida a função de geração de hash chamada SHA3-256, sendo esse um dos padrões indicados pelo NIST como seguro para a geração de hash e por permitir uma chave de 256 bits. Entretanto, essa função exige um alto consumo de processamento para a sua execução. A título de medida de tempos, ao trazermos a biblioteca padrão do Arduíno para o MSP430 e fazermos as devidas adaptações para ele rodar, obtivemos que cada hash gerado pelo SHA3-256 demorava em torno de 45 milissegundos em sequências de 256 bytes.

Entretanto, para a geração de um único hash HMAC é necessário rodar duas vezes o SHA3 conforme descrito em 2.5, e somando-se a isso outros procedimentos de preparação e execução. Chega-se a 0,1 segundo para a geração de um hash. Isto sem nenhum processamento paralelo que possa estar compartilhando recursos e atrasar os procedimentos, o que não ocorre na janela de comunicações entre o satélite e a base.

Essa demora na execução do SHA3 é principalmente em decorrência da grande quantidade de rotação de palavras de 64 bits que no MSP430 é um procedimento bastante lento, demorando em torno de 5 ciclos de clock para cada bit a ser rotacionado, pois é necessário utilizar o carry para a transferência de informações entre cada palavra de 16 bits. Para minimizar essa demora, propomos a utilização do módulo de hardware multiplicador de 32 bits do MSP430.

4.2.1 Uso do MPY32 - 32-Bit Hardware Multiplier

O módulo de hardware multiplicador de 32 bits foi explicado no Capítulo 2 e em 4.2 e pode ser utilizado para agilizar o processo de rotação de bits em palavras maiores que o número de bits padrão do controlador, que no caso do MSP430 é de 16 bits. Pois assim como podemos substituir matematicamente uma divisão por 2 pela rotação à direita de um bit, podemos substituir a rotação de alguns no sentido contrário bits pela sua multiplicação deste valor por uma potência de 2.

Para utilizarmos esse recurso, desenvolvemos a metodologia descrita passo-a-passo na lista a seguir, considerando (E) a entrada da palavra de 64 bits e (roda) a quantidade de bits a serem rotacionados a esquerda.

1. Calculamos o número de palavras de 16 bits que será necessário rodar (numPalavra=roda/16);
2. Calculamos o número de bits para rodar na palavra de 16 bits (n = mod16(roda));
3. Definimos a potência de 2 necessária para a rotação dos bits na palavra de 16 bits (ROT=2^n));
4. Separamos a palavra de 64 bits em duas palavras de 32 bits;
5. Escrevemos a primeira palavra de 32 bits no MPY32(MPYS_H e MPYS_L);
6. Escrevemos o valor de ROT no MPY(OP2);
7. Lemos o valor de RES2 e gravamos em S[mod(numPalavra)];
8. Lemos o valor de RES1 e gravamos em S[mod(numPalavra+3)];
9. Lemos o valor de RES0 e gravamos em S[mod(numPalavra+2)];
10. Escrevemos a segunda palavra de 32 bits no MPY32(MPYS_H e MPYS_L);
11. Escrevemos o valor de ROT no MPY(OP2);
12. Lemos o valor de RES2 e fazemos um ”OU” com a S[mod(numPalavra+2)];
13. Lemos o valor de RES1 e gravamos em S[mod(numPalavra+1)];
14. Lemos o valor de RES0 e fazemos um ”OU” com a S[mod(numPalavra)];

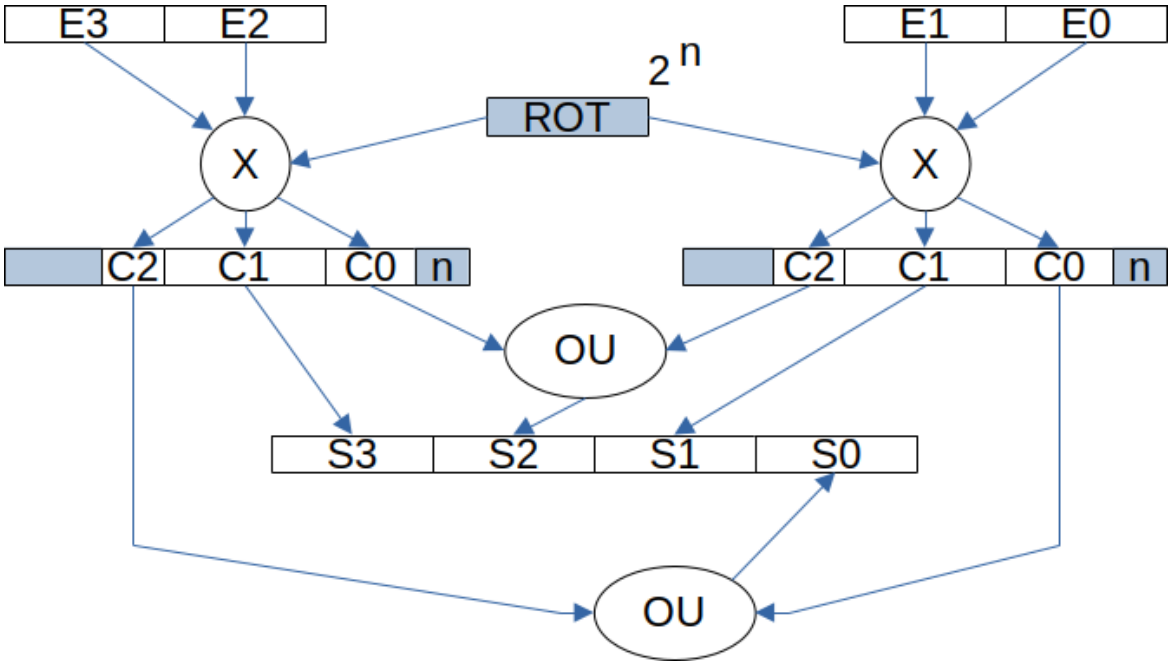

Figura 4.2 – Rotação alguns bits utilizando o MPY de 32 bits, processador de 16 bits em palavras a ser rotacionada de 64 bits

Na figura 4.2 demonstramos graficamente o descrito acima, para facilitar o diagrama, estamos utilizando uma rotação inferior a 15 bits.

E tendo em vista que precisamos de agilidade, escolhemos eliminar os passos de cálculos dos expoentes e do número de palavras a serem rotacionadas, sendo feitas macros específicas para cada quantidade de bits a serem rotacionados.

4.3 INFRAESTRUTURA EXPERIMENTAL

Para os testes das medidas propostas foi montada uma infraestrutura em uma rede WiFi composta por um celular como ponto de acesso e um notebook com duas máquinas virtuais, sendo uma simulando a base e a outra um possível atacante visando injetar código malicioso. Foi conectada a rede WiFi, um kit de desenvolvimento ESP-32 para simular o rádio de comunicação do satélite. O ESP32 foi conectado via UART a um kit de desenvolvimento MSP430 normalmente utilizados em "CubeSats".

A escolha do ESP-32 para a interface de rádio foi pelo seu alto poder de processamento utilizado para os vários serviços necessários à simulação. Entre esses serviços está a comunicação via UDP entre o ESP-32 e as máquinas virtuais, a simulação das falhas de comunicação entre a base e o satélite, a geração de relatórios de tráfego e o detalhamento dos pacotes. A partir desse ponto chamaremos o MSP430 de satélite, o ESP-32 de rádio e as maquinas virtuais por suas funções, base e atacante, conforme figura 4.3.

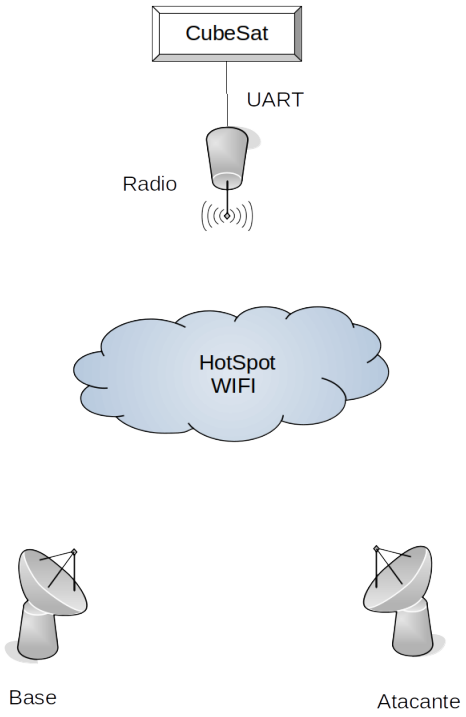

Figura 4.3 – Infraestrutura Experimental

O rádio nessa arquitetura tem a função de simular o ambiente adverso e dar uma leve vantagem ao atacante, para isso ele foi programado para:

1. Transmitir os sinais da base para o satélite com falha de 40%, recebendo via UDP na

porta 1818 e transferindo para o satélite via UART;

2. Transmitir os sinais do satélite para a base com falha de 20%, recebendo via UART e transferindo para a base via UDP na porta 1818;

3. Retransmitir todas as comunicações entre base e satélite para o atacante via UDP na porta 1881;

4. Transmitir os sinais do atacante para o satélite sem falhas, recebendo via UDP na porta 1881 e transferindo para o satélite via UART;

5. Gerar página web com todas as transmissões, inclusive as bloqueadas, conforme 4.4.

Sentido	País	Satélite	ID	Tamanho	Comp	Mensagem
B->S	x37	x00x01	x00	x00	x00x7F	
S->B	x37	x00x01	x00	x00	x00x7F	
BLOK	x37	x00x01	x01	xC6	x09x11	0x10tesuando
BLOK	x37	x00x01	x02	x9B	x21x11	0x11$ú¿vHZÝÅ#ß, »´T²ãXaôçw5Î—mdÆ
BLOK	x37	x00x01	x01	xC6	x09x11	0x10tesuando
BLOK	x37	x00x01	x02	x9B	x21x11	0x11$ú¿vHZÝÅ#ß, »´T²ãXaôçw5Î—mdÆ
BLOK	x37	x00x01	x01	xC6	x09x11	0x10tesuando
B->S	x37	x00x01	x02	x9B	x21x11	0x11$ú¿vHZÝÅ#ß, »´T²ãXaôçw5Î—mdÆ
S->B	x37	x00x01	x00	x3B	x01x00	0x02
B->S	x37	x00x01	x01	xC6	x09x11	0x10tesuando
S->B	x37	x00x01	x00	x3D	x02x00	0x020x01
B->S	x37	x00x01	x00	x00	x00x7E	
S->B	x37	x00x01	x00	x00	x00x7E	
B->S	x37	x00x01	x00	x00	x00x7F	
S->B	x37	x00x01	x00	x00	x00x7F	
B->S	x37	x00x01	x01	x5D	x01x11	0x12
S->B	x37	x00x01	x01	x5F	x01x11	0x14
BLOK	x37	x00x01	x00	x3A	x01x00	0x01
S->B	x37	x00x01	x01	x5F	x01x11	0x14
B->S	x37	x00x01	x00	x3A	x01x00	0x01
B->S	x37	x00x01	x01	x5D	x01x11	0x12
S->B	x37	x00x01	x01	x5F	x01x11	0x14
B->S	x37	x00x01	x00	x3A	x01x00	0x01
BLOK	x37	x00x01	x01	x5F	x01x11	0x14
BLOK	x37	x00x01	x01	x5F	x01x11	0x14

Figura 4.4 – Relatório de Tráfego

O satélite e a base foram programados para atender aos requisitos do protocolo apresentado, sendo que a base foi programada para indicar os inícios e fim das janelas de comunicação, em um caso real isso seria definido por temporizadores e/ou geolocalizações. O atacante foi programado para receber as transmissões e simular injeção de código, tentando senhas aleatórias.

4.3.1 Falhas de transmissão

A transmissão de dados sempre está sujeita a falhas e para o presente caso, temos a interferência da SAA citada em 2.2. Essa anomalia imputa falhas em períodos de tempos contínuos e liberações em tempos contínuos, o que nesse trabalho é interpretado como maior probabilidade de ocorrência de falha quando as falhas estão ocorrendo e menor probabilidade de falha quando as falhas não estão ocorrendo, conforme segue:

1. Para falhas de 20%
    (a) Quando a transmissão do bit anterior foi efetivo, a probabilidade de falha é de 1,25%
    (b) Quando a transmissão do bit anterior tiver falhado, a probabilidade de nova falha é de 95%
2. Para falhas de 40%
    (a) Quando a transmissão do bit anterior foi efetivo, a probabilidade de falha é de 3,33%
    (b) Quando a transmissão do bit anterior tiver falhado, a probabilidade de nova falha é de 95%

Considerando as informações disponíveis que demonstram uma taxa de perdas da base para o satélite em torno de 40% e do satélite para a base na ordem de 20%. Assim, para a simulação das perdas com o efeito avalanche, optamos por utilizarmos fatores diferentes para os casos em que esta ocorrendo falha ou não ocorrendo falhas. O método é muito simples, aumentamos a probabilidade de ocorrer falhas quando as falhas estão ocorrendo e diminuímos a probabilidade de ocorrência de falhas quando a falha não está ocorrendo.

Para a efetiva determinação de falha, o sistema verifica se o último byte testado passou ou não. Caso não tenha passado, calcula o módulo de 20 de um número gerado pelo gerador aleatório do ESP32 (esp_rand), o que corresponde a uma probabilidade do próximo byte passar de 5%. Para o caso do byte anterior tenha passado, utiliza o Fator de Passagem indicado na tabela 4.1 para calcular se o byte vai passar ou não.

Tabela 4.1 – Fatores de Passagem obtidos empiricamente

Percentual de Falha	Fator de Passagem
20%	30
40%	80

4.3.2 Visualização Base e Ataque

Para a verificar as respostas de comunicação que a base e o ataque possuem com o satélite, foram elaborados algumas rotinas que utilizam do "Dialog"do Linux conforme (JARGAS, 1999) para facilitar a visualização. A primeira tela 4.5 e mostra apenas as opções disponíveis de operação.

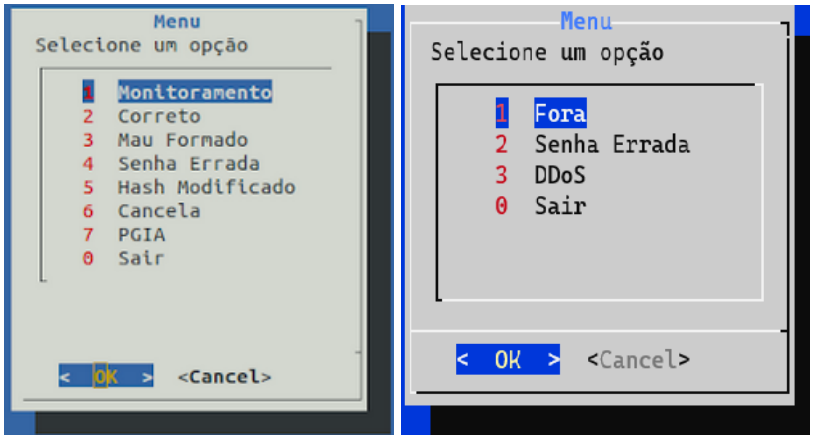

Figura 4.5 – Primeira Tela - Base e Atacante

A segunda tela 4.6 mostra o processamento feito pela base conforme a solicitação feita na primeira tela e as suas comunicações com a base.

Na segunda tela os seguintes itens:

1. UDP - Indicando a porta que está trabalhando e o início do processo de recepção de dados;
2. PGE - Indicando o início e fim de cada janela de conexão
3. PGIA - Indicando as informações recebidas na janela anterior e a serem transmitidas na próxima janela de comunicação.

Figura 4.6 – Segunda Tela - Monitoramento

5 RESULTADOS

O presente trabalho obteve vários resultados. Os resultados principais estão alinhados ao objetivo principal permitir a inclusão de novos serviços protegendo o sistema da injeção de códigos maliciosos ou mal formados. Os resultados são todos os que auxiliaram a execução do projeto. Os resultados principais foram a efetiva implementação de novas funcionalidades pela base, a defesa quanto a alguns ataques e a impossibilidade de alteração do núcleo do sistema operacional por códigos maliciosos. Os resultados secundários foram a implantação de alguns mecanismos implantação de mecanismos como a melhoria na eficiência do uso do SHA3 para a implantação do HMAC, a separação do código do sistema em núcleo principal e serviços, e a implantação de uma metodologia de simulação de falhas na transmissão.

5.1 IMPLANTAÇÃO DE NOVOS SERVIÇOS

Conforme a Figura 5.1, o sistema permitiu a implantação do principal objetivo do trabalho, a injeção de novos serviços ao satélite em operação.

Figura 5.1 – Resultado da implantação de novo serviço

Onde podemos observar o processo de implantação de novos serviços, o processo ocorreu

conforme descritivo a seguir e atende o PGIA conforme a Figura 5.2.

1. O sistema está rodando e o operador define o envio de um novo serviço.
2. Na primeira janela fora de comunicação após a definição o sistema mostra as linhas de PGIA: "Registrou Binário Correto", "Enviando Código"e "Enviando Hash".
3. Na primeira janela de comunicação são enviados os códigos indicados na etapa anterior;
4. No próximo período sem comunicação o operador confirma a injeção do código: "Enviando Código de Confirmação";
5. Na próxima janela de comunicação são trocadas as informações de confirmação de injeção e de validação do código;
6. No próximo período sem comunicação a base identifica que o código foi validado: "Código Validado".
7. Na próxima janela de comunicação a base recebe o código de efetiva implantação;
8. No próximo período sem comunicação a base identifica que o código foi implantado: "Código Implantado".
9. Na próxima janela de comunicação a base recebe resposta do novo serviço;
10. No próximo período sem comunicação a base identifica a resposta do novo serviço: "novo";

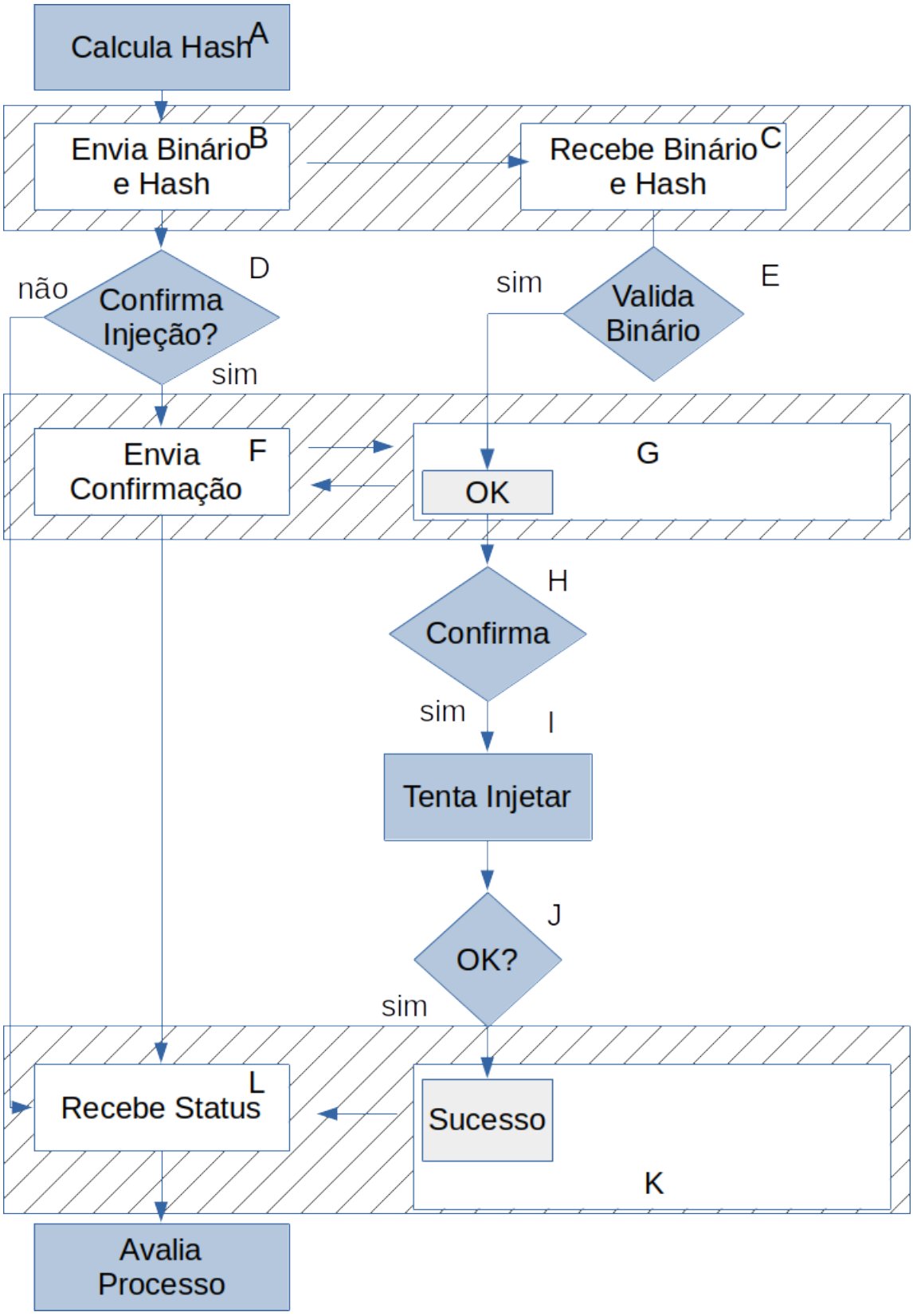

Figura 5.2 – Fluxo da implantação de novo serviço

5.2 PROTEÇÃO QUANTO AOS ATAQUES

Existem vários ataques e problemas que podem ser tentados nesse tipo de equipamentos, nas seguintes subseções descrevemos como o sistema se defende dos vários ataques, esse processo deve atender ao PGIA conforme a Figura 5.3:

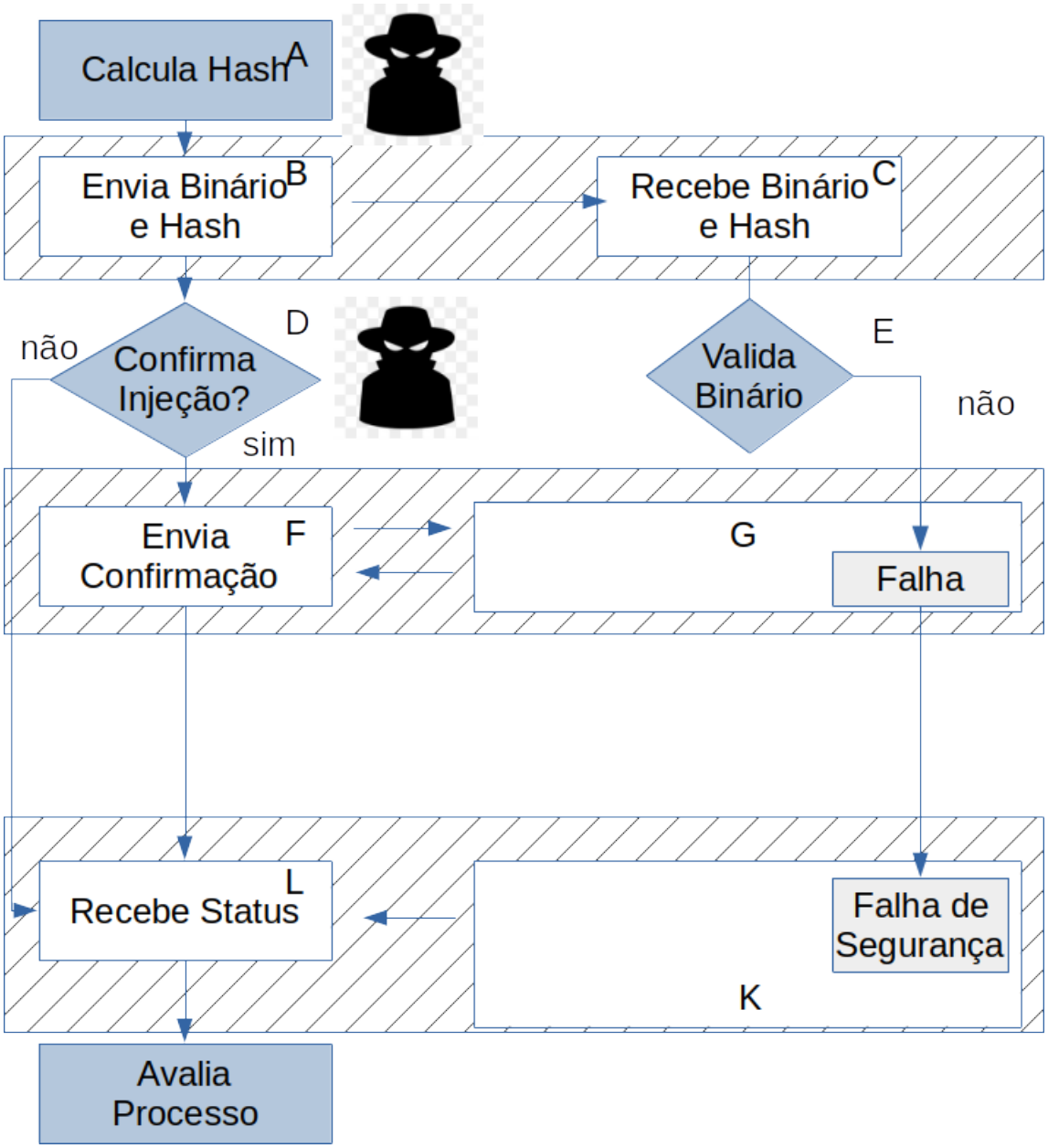

Figura 5.3 – Fluxo do sistema sob ataque

5.2.1 Ataque fora do período de comunicação

Uma possibilidade ao atacante é a tentativa de comunicação com o satélite fora da janela de comunicação entre o satélite e a base, o que foi sumariamente ignorada pelo satélite.

5.2.2 Uso de senha errada

Durante o período de comunicação o atacante conhecedor do sistema pode tentar enviar um novo serviço para ser executado no satélite, neste caso o sistema se comportou conforme a figura 5.4, impedindo a implantação do código pretendido pelo atacante, conforme o passo-a-passo a seguir.

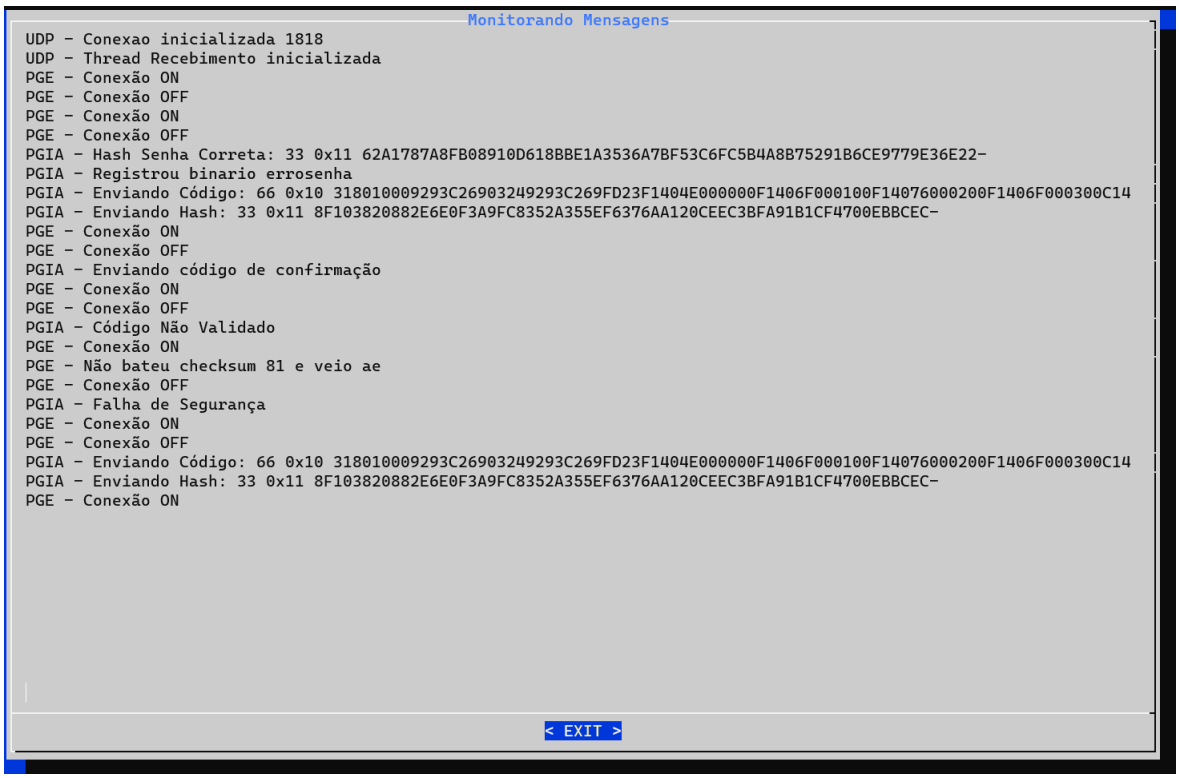

Figura 5.4 – Ataque sem o conhecimento da senha

1. O sistema está rodando e o atacante define o envio de um novo serviço utilizando a senha errada.

2. Na primeira janela fora de comunicação após a definição o sistema mostra as linhas de PGIA: "Hash Senha Correta", "Registrou Binário errosenha", "Enviando Código"e "Enviando Hash".

3. Na primeira janela de comunicação são enviados as linhas "Enviando Código"e "Enviando Hash";

4. No próximo período sem comunicação o operador confirma a injeção do código: "Enviando Código de Confirmação";

5. Na próxima janela de comunicação são trocadas as informações de confirmação de injeção e de validação do código, observar a ocorrência de um erro de checksum;

6. No próximo período sem comunicação a base identifica que o código não foi validado: "Código Não Validado".

7. Na próxima janela de comunicação a base recebe o código de falha de segurança;

8. No próximo período sem comunicação a base identifica que o código foi implantado: "Código Implantado".

9. Na próxima janela de comunicação a base recebe o Código binário e o hash que o satélite havia recebido;

10. No próximo período sem comunicação a base identifica os códigos Binários e os hash que o satélite recebeu para implantar, mas que não foram efetivos.

5.2.3 Envio de código alterado

Outro meio de ataque é a alteração proposital do código já validade na base, substituindo alguma cadeia de caracteres de forma intencional. A tentativa deste ataque é demonstrada na figura 5.5. Onde o sistema foi capaz de identificar esta alteração e não implantar o código malicioso, conforme o passo-a-passo a seguir.

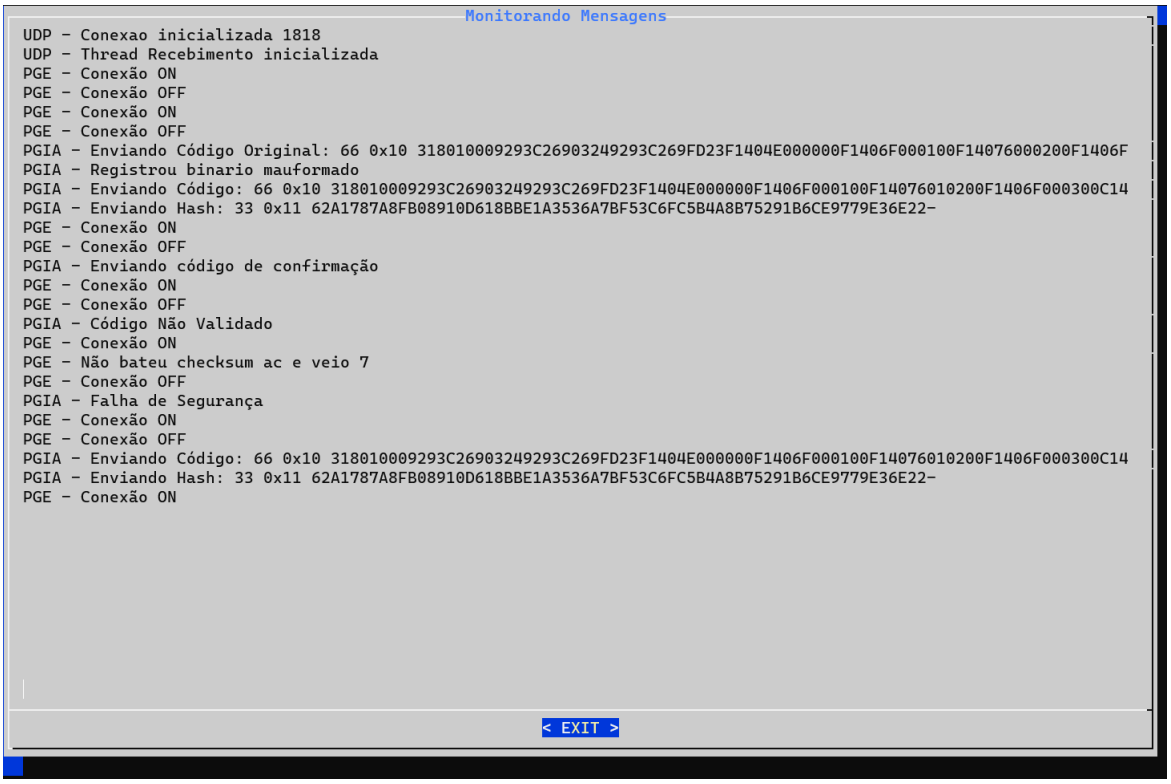

Figura 5.5 – Alteração de código já validado

1. O sistema está rodando e o atacante define o envio de um novo serviço utilizando uma alteração no código binário.

2. Na primeira janela fora de comunicação após a definição, o sistema mostra as linhas de PGIA: "Código original", "Registrou Binário malformado", "Enviando Código"e "Enviando Hash".

3. Na primeira janela de comunicação são enviados as linhas "Enviando Código"e "Enviando Hash";

4. No próximo período sem comunicação o operador confirma a injeção do código: "Enviando Código de Confirmação";

5. Na próxima janela de comunicação são trocadas as informações de confirmação de injeção e de validação do código, observar a ocorrência de um erro de checksum;

6. No próximo período sem comunicação a base identifica que o código não foi validado: "Código Não Validado".

7. Na próxima janela de comunicação a base recebe o código de falha de segurança;

8. No próximo período sem comunicação a base identifica que o código foi implantado: "Código Implantado".

9. Na próxima janela de comunicação a base recebe o Código binário e o hash que o satélite havia recebido;

10. No próximo período sem comunicação a base identifica os códigos Binários e os hash que o satélite recebeu para implantar, mas que não foram efetivos.

5.2.4 Ataque de negação de serviço

Em alguns casos o atacante quer apenas impedir a disponibilização dos serviços, os ataques conhecidos como DoS - Denial of Services. Neste caso o sistema foi incapaz de injetar novos código, impedindo a disponibilização de novos serviços pelo satélite. Entretanto, esse processo não afetou os serviços que já estavam sendo fornecidos pelo satélite, tornando este ataque parcialmente efetivo.

Para este ataque ser parcialmente efetivo, basta o atacante enviar no início de cada janela de comunicação, um sinal composto de um hash e um código binário qualquer. Neste caso o sistema do satélite irá entender que precisa processar os dados recebidos e irá ficar 4 ciclos de comunicação até negar essa tentativa, impedindo que a base injete códigos novos. Na figura 5.6 podemos observar a tentativa de implementação de novo serviço sob ataque de negação de serviço, devemos observar que o satélite fica confuso com as informações recebidas e fica enviando as informações de falha de validação e os códigos que recebe, mas não implanta nenhum código.

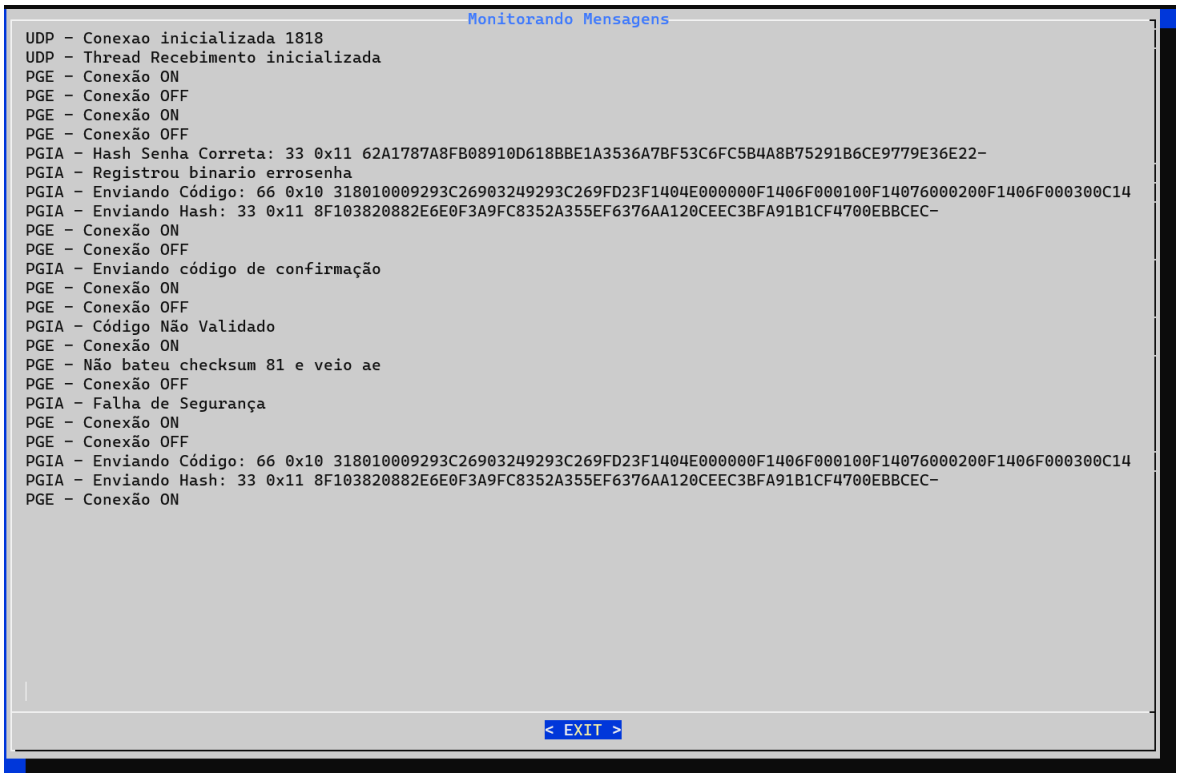

Figura 5.6 – Ataque DoS - Negação de serviço

5.3 EFICIÊNCIA DO SHA3

A solução para melhorar a execução do SHA3 foi explanada no tópico 4.2.1 onde obtivemos uma redução de 52 para 29 milissegundos o tempo de geração de um hash SHA3-256, indicando uma redução de tempo na ordem de 35%, e se considerarmos que para a geração de um hash HMAC são necessários a geração de 2 hashs SHA3-256, a redução de tempo chega a 46 milissegundos para cada geração de hash HMAC.

5.4 O USO DA MPU

A defesa aos ataques apresentados na seção anterior está baseada na impossibilidade do atacante saber a senha compartilhada entre a base e o satélite. Mas, o que ocorreria no satélite caso o atacante descobrisse a senha por algum método? Para respondermos a essa pergunta à luz do sistema proposto, temos que analisar os objetivos do atacante:

Para o caso da implantação de serviço malicioso ou alteração maliciosa de serviço implantado, o sistema permitiria a implantação, ou alteração do serviço, conforme apresentado no item 5.1. Neste caso, a observação das comunicações seria fundamental para a identificação da injeção e a escolha de como reestabelecer o sistema.

Outro objetivo, pode ser a alteração do núcleo do sistema operacional, neste caso o sis-

tema também permitiria a implantação do código malicioso. Entretanto, para este caso, a tentativa de execução do código malicioso iria incorrer em exceção do sistema e sua reinicialização. Podendo causar, no limite a perda do acesso e controle do satélite tanto pela base quanto pelo atacante.

6 CONCLUSÃO

O uso dos protocolos de garantia de entrega e de garantia de integridade e autenticidade, PGE e PGIA, demonstrados na seção anterior, nos permitem avaliar positivamente a sua implementação com o uso do FreeRTOS - Sistema Operacional de Tempo Real, para a injeção segura de códigos em satélites CubeSat. Esta implementação teve por objetivo propiciar uma forma segura de incluir novos serviços neste tipo de equipamento, tendo em vista as suas condicionantes de baixa taxa de comunicação, pequenas janelas de comunicação e restrições de hardware.

Para o desenvolvimento do sistema, foram criadas, melhoradas e implementadas várias ferramentas, tais como, uma infraestrutura para a simulação do ambiente de comunicação, otimização da execução do SHA3, adaptação do FreeRTOS com uso do módulo de proteção de memória (MPU), além do próprio desenvolvimento e implementação dos protocolos PGE e PGIA.

A infraestrutura demonstrou incluir os problemas de comunicação encontrados no Brasil, incluindo o fenômeno da Anomalia do Atlântico Sul - *South Atlantic Anomaly* (SAA), e permitir em rede IP, utilizando o protocolo UDP, o trafego dos dados conforme os modelos de rádio utilizado nas comunicações entre base e satélites do tipo CubeSat, tudo isso graças as intercomunicações desenvolvidas para o ESP32.

A otimização do SHA3 que foi desenvolvida para o MSP430, que é uma opção para uso nos CubeSats, utilizou o Módulo de Multiplicação de 32 bits para agilizar as rotações de palavras de 256 bits conseguindo uma redução de 35% no tempo necessário para os cálculos, o que para as condições apresentadas é muito interessante, tendo em vista que para garantir a integridade e autenticidade é necessário gerar o HMAC do código e esse processo é custoso em termos de hardware.

O uso da MPU para a separação do FreeRTOS em um núcleo que não pode ser modificado, intencionalmente ou não, e uma área de serviços que pode ser alterada pela sistemática desenvolvida, trouxe para o sistema garantia extra contra possíveis ataques. Mesmo que o ataque seja efetivo com a senha correta, não poderá alterar o núcleo no satélite, mantendo a possibilidade do reestabelecimento dos serviços pela base.

Todas essas contribuições se interligam para a formação de um sistema operacional capaz de disponibilizar uma forma segura de permitir a inclusão de novos serviços ao satélite após o seu lançamento. Esta possibilidade cumpriu com o objetivo principal deste trabalho de inclusão de novos serviços após o lançamento do satélite, otimizando assim, o ciclo de vida do satélite, tanto por permitir que seja lançado com menos funcionalidades, um reduzido

ciclo de desenvolvimento, como permitindo a implementação de novas ideias e necessidades que possam surgir após o lançamento.

6.1 TRABALHOS FUTUROS

Entretanto, essas necessidades não são exclusivas dos satélites, no universo de IoT - Internet das Coisas, existem muitos casos em que um sistema operacional capaz de permitir a inclusão segura de novos códigos em seus elementos torna-se útil, seja pela dificuldade de acesso ao equipamento, como em canais, tubulações, torres, ou localizações remotas, seja pela questão econômica de logística para a atualização de vários equipamentos em vários locais distintos.

Assim, a evolução dos conceitos, técnicas e ferramentas ora propostos e implementados é uma alternativa viável para trabalhos futuros, tais como, a implementação de novos protocolos para as condições de rede internet e a previsão de ofuscação das funcionalidades a serem implementadas, seja por questões estratégicas, seja por questões de propriedade intelectual.

Tratando especificamente do MSP430, o seu módulo de proteção de memória possui uma previsão para resguardo de propriedade intelectual, não utilizado neste trabalho pela natural dificuldade física de acesso ao hardware do equipamento após o seu lançamento, mas pode servir para proteger a propriedade intelectual em outras situações. Neste caso seria interessante um protocolo que garantisse que a transmissão dos mecanismos seja feita de forma confidencial.

Outras opções interessantes a serem trabalhadas é o uso de PUF - Physically Unclonable Function (função fisicamente não replicáveis) em microcontroladores que possuam essa função para as garantias de integridade e autenticidade.

Assim, o presente trabalho mostrou-se motivador. Primeiro pelo estudo, aprendizado, desenvolvimento e aplicação de técnicas necessárias à inclusão de novos serviços a um satélite já em órbita, mitigando a possibilidade de ataques ao mesmo. Pelo desenvolvimento de um sistema operacional de tempo real baseado no FreeRTOS capaz de manter as suas características básicas mesmo sob-ataque. Do uso de técnicas específicas para o microcontrolador MSP430FR5994 que podem ser adaptadas para outros microcontroladores. E pela geração de protocolos capazes de permitir e proteger o sistema de injeção de novos serviços em um equipamento já em operação.

Mas também, pelas possíveis evoluções, robustecimentos e novas possibilidades decorrentes de novos desenvolvimentos baseados nos conhecimentos ora desenvolvidos e aplicados ao presente trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

ARUNA, S. et al. Lightweight cryptography algorithms for iot resource-starving devices. *Role of Edge Analytics in Sustainable Smart City Development: Challenges and Solutions*, Wiley Online Library, p. 139–169, 2020.

BELLARE, M.; CANETTI, R.; KRAWCZYK, H. Keying hash functions for message authentication. In: SPRINGER. *Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16*. [S.l.], 1996. p. 1–15.

BELLARE, M.; CANETTI, R.; KRAWCZYK, H. Message authentication using hash functions: The hmac construction. *RSA Laboratories' CryptoBytes*, Citeseer, v. 2, n. 1, p. 12–15, 1996.

BISIO, I. et al. Comparison among resource allocation methods with packet loss and power metrics in geostationary satellite scenarios. In: IEEE. *2013 IEEE International Conference on Communications (ICC)*. [S.l.], 2013. p. 4271–4275.

BRANDAO, H. *1º caso na história: hackers assumem controle de satélite da ESA*. 2023. Disponível em: <https://gizmodo.uol.com.br/1o-caso-na-historia-hackers-assumem-controle-de-satelite-da-esa/>.

DENARDIN, G. W.; BARRIQUELLO, C. H. *Sistemas operacionais de tempo real e sua aplicaçao em sistemas embarcados*. [S.l.]: Editora Blucher, 2019.

FALCO, G.; VISWANATHAN, A.; SANTANGELO, A. Cubesat security attack tree analysis. In: IEEE. *2021 IEEE 8th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*. [S.l.], 2021. p. 68–76.

FEREIDOONI, H. et al. Fitness trackers: fit for health but unfit for security and privacy. In: IEEE. *2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*. [S.l.], 2017. p. 19–24.

FERRER, T.; CÉSPEDES, S.; BECERRA, A. Review and evaluation of mac protocols for satellite iot systems using nanosatellites. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 19, n. 8, p. 1947, 2019.

FREERTOS - Real-time operating system for microcontrollers. 2023. Disponível em: <https://www.freertos.org/>.

GUZMAN, L. B. de; SISON, A. M.; MEDINA, R. P. Md5 secured cryptographic hash value. In: *Proceedings of the 2018 International Conference on Machine Learning and Machine Intelligence*. [S.l.: s.n.], 2018. p. 54–59.

HEIRTZLER, J. The future of the south atlantic anomaly and implications for radiation damage in space. *Journal of Atmospheric and Solar-Terrestrial Physics*, v. 64, n. 16, p. 1701–1708, 2002. ISSN 1364-6826. Space Weather Effects on Technological Systems. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1364682602001207>.

HISCOCK, T.; SAVRY, O.; GOUBIN, L. Lightweight instruction-level encryption for embedded processors using stream ciphers. *Microprocessors and Microsystems*, Elsevier, v. 64, p. 43–52, 2019.

HWANG, M.-S.; YANG, C.-C.; SHIU, C.-Y. An authentication scheme for mobile satellite communication systems. *ACM SIGOPS Operating Systems Review*, ACM New York, NY, USA, v. 37, n. 4, p. 42–47, 2003.

JARGAS, A. *Dialog –tudo*. 1999. Disponível em: <https://aurelio.net/shell/dialog/>.

KWARTENG, E.; CEBE, M. A survey on security issues in modern implantable devices: Solutions and future issues. *Smart Health*, Elsevier, p. 100295, 2022.

LEE, R. P.; MARKANTONAKIS, K.; AKRAM, R. N. Ensuring secure application execution and platform-specific execution in embedded devices. *ACM Transactions on Embedded Computing Systems (TECS)*, ACM New York, NY, USA, v. 18, n. 3, p. 1–21, 2019.

NASUDDIN, K. A.; ABDULLAH, M.; HAMID, N. S. A. Characterization of the south atlantic anomaly. *Nonlinear Processes in Geophysics*, Copernicus Publications Göttingen, Germany, v. 26, n. 1, p. 25–35, 2019.

PAAR, C.; PELZL, J. Sha-3 and the hash function keccak. *Understanding Cryptography-A Textbook for Students and Practitioners*, 2010.

PHILIP, M. A. et al. A survey on lightweight ciphers for iot devices. In: IEEE. *2017 International Conference on Technological Advancements in Power and Energy (TAP Energy)*. [S.l.], 2017. p. 1–4.

RADIS, A. H.; GONDIM, J. J. C.; CAFÉ, D. C. Proposed security measures for code injection for cubesats. In: IEEE. *2022 Workshop on Communication Networks and Power Systems (WCNPS)*. [S.l.], 2022. p. 1–7.

RAHIMI, P. et al. Trends and challenges in ensuring security for low-power and high-performance embedded socs. In: IEEE. *2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*. [S.l.], 2021. p. 226–233.

RAMADAN, R. Internet of things (iot) security vulnerabilities: A review. *PLOMS AI*, v. 2, n. 1, 2022.

REIS, L. et al. An overview of the alfa crux cubesat mission for narrowband communication. *Advances in the Astronautical Sciences*, Univelt Incorporated, n. 173, p. 245–255, 2020.

SINHA, M.; DUTTA, S. Survey on lightweight cryptography algorithm for data privacy in internet of things. In: SPRINGER. *Proceedings of the Fourth International Conference on Microelectronics, Computing and Communication Systems*. [S.l.], 2021. p. 149–157.

STALLINGS, W. *Criptografia e segurança de redes. Princípios e práticas. 6ª ed*. 6ª. ed. [S.l.]: Pearson Prentice Hall, 2015.

TAXONERA, R. I. et al. *EDAC software implementation to protect small satellites memory*. Dissertação (Mestrado) — Universitat Politècnica de Catalunya, 2019.

TEXAS INSTRUMENTS. *User's Guide - MSP430FR58xx, MSP430FR59xx, and MSP430FR6xx Family*. [S.l.], 2012. Revised April 2020.

TEXAS INSTRUMENTS. *MSP430 FRAM Tecnology - How To and Best Pratices*. [S.l.], 2014. Revised August 2021.

TEXAS INSTRUMENTS. *Data Sheet - MSP430FR599x, MSP430FR596x Mixed-Signal Microcontrollers*. [S.l.], 2016. Revised January 2021.

TOKUMITSU, M.; TSUJI, M.; NAKAYA, J. Survey on harness design for cubesats: Understanding the constraints of cubesats design and toward an optical wireless bus for cubesats. 2023.

TSAUR, W.-J.; CHANG, J.-C.; CHEN, C.-L. A highly secure iot firmware update mechanism using blockchain. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 22, n. 2, p. 530, 2022.

WANG, N. et al. Firmware security verification method of distance learning terminal based on md5 algorithm. In: FU, W.; SUN, G. (Ed.). *e-Learning, e-Education, and Online Training*. Cham: Springer Nature Switzerland, 2022. p. 436–450. ISBN 978-3-031-21164-5.

WIKMAN, J.; SJÖBLOM, J. *Software based memory correction for a miniature satellite in low-Earth orbit*. 2017.

YUE, P. et al. On the security of leo satellite communication systems: Vulnerabilities, countermeasures, and future trends. *arXiv preprint arXiv:2201.03063*, 2022.