CryptoComponent: um Componente de Criptografia para Sistemas de Software IoT de Baixo Custo

Bruno D. de Paiva, Bruno P. de Souza, Guilherme H. Travassos

Universidade Federal do Rio de Janeiro (UFRJ) - Rio de Janeiro – RJ – Brasil

{bdantas, bpsouza, ght}@cos.ufrj.br

Abstract. *The large number of IoT software systems devices motivated by the great advance of Industry 4.0 led organizations to list the most recurring security risks. Secure data transfer and information storage are among the top ten problems that most occur in these IoT devices. Therefore, this work presents a cryptographic component that allows devices with low computational power to transmit data securely, contributing to mitigating the lack of security in low-cost IoT software systems devices. Two in silico studies evaluated the cryptographic component to verify its feasibility and efficiency. The results suggest that the component adequately performs when encapsulated into a low cost IoT device.*

Resumo. *O grande número de dispositivos de sistemas de software IoT motivados pelo avanço da Indústria 4.0 tem levado organizações a listar os riscos de segurança mais comuns. A transferência segura de dados e o armazenamento de informações estão entre os dez problemas mais frequentes nesses dispositivos. Assim, este trabalho apresenta um componente de criptografia que permite que dispositivos IoT com baixo poder computacional transmitam dados seguramente, contribuindo para mitigar a falta de segurança nestes sistemas. Esse componente foi avaliado em dois estudos in silico para verificar sua viabilidade e eficiência. Os resultados sugerem que o componente apresenta desempenho adequado quando encapsulado em um dispositivo IoT de baixo custo.*

1. Contexto e Problemática

A Internet das Coisas (*Internet of Things* - IoT) representa um paradigma computacional no qual uma infraestrutura de rede global e dinâmica permite com que *coisas* virtuais e físicas tenham identidades, atributos físicos e utilizem interfaces inteligentes, facilitando sua integração e apresentação como uma rede de informações. Esta infraestrutura permite que objetos cotidianos usualmente não conectáveis a internet se tornem dispositivos IoT por meio da integração de hardware e embarcação de software [Othman et al. 2020]. Assim, sistemas de software construídos no paradigma da IoT promovem a interação com estes dispositivos para obter capacidade de identificação, sensoriamento e atuação nos ambientes em que estão inseridos [Motta *et al.,* 2018].

De acordo com o relatório anual da Cisco de 2020, os sistemas de software IoT promoveriam conexões máquina-máquina correspondentes a cerca de 15 bilhões de dispositivos IoT conectados no ano de 2023 [Cisco 2020]. Estes dispositivos IoT, por sua vez, não oferecem alta capacidade de proteção, pois usualmente possuem capacidade limitada de recursos computacionais, tais como limitações naturais de memória e processamento. Desta forma, dispositivos IoT são mais suscetíveis a ataques externos,

uma vez que a disponibilidade de aplicações de métodos de segurança é restrita e insuficiente [Ogonji et al. 2020]. Portanto, a vulnerabilidade de dispositivos IoT aliada a previsão de sua utilização em larga escala fomentou a busca por soluções para garantir sua segurança, principalmente para a conexão e transferência de dados dos usuários, tendo em vista a possibilidade de vazamento de informações [Bradley et al. 2018]. Adicionalmente, a falta de segurança adequada nestes dispositivos aumenta a necessidade da proteção das *coisas* para impedir falhas na proteção de privacidade. Por exemplo, cidades e carros inteligentes possuem canais de comunicação direto entre os dispositivos IoT e os usuários permitindo revelar informações sobre sua localização e dados pessoais mais sensíveis [Rana et al. 2022].

Na linha de falta de segurança em sistemas de software IoT, a OWASP (*Open Web Application Security Project*), uma organização sem fins lucrativos com o objetivo de melhorar a segurança do software, listou os dez maiores problemas de segurança específicos para estes sistemas, incluindo desde a falta de mecanismos para atualização de segurança até a transferência e armazenamento inseguro de dados [OWASP 2019].

A literatura técnica apresenta alternativas para lidar com problemas de segurança de dados em sistemas de software IoT. Por exemplo, aplicar métodos de criptografia, limitar o acesso aos dispositivos por meio de autenticação, dentre outros [Rachit et al. 2021]. Neste trabalho, seguimos a opção de aplicar métodos de criptografia. Segundo Mousavi et al. (2021), algoritmos de criptografia leve podem ser adequados para resolver o problema de segurança de dados em dispositivos IoT. Neste sentido, a ausência de recursos computacionais nos dispositivos IoT de baixo custo e a necessidade de segurança na comunicação entre dispositivos e seus sistemas de software tornam os algoritmos de *Lightweight Cryptography* adequados por reduzir o uso da unidade de processamento central (CPU) e memória.

Neste sentido, este trabalho, um projeto final do curso de graduação da Engenharia de Computação e Informação da Escola Politécnica da UFRJ na temática de engenharia de sistemas de software contemporâneos, descreve e apresenta a avaliação experimental de um componente que encapsula um algoritmo de criptografia leve e uma de suas operações para manter um canal seguro de comunicação entre dispositivos IoT de baixo custo (utilizando processadores ESP32 e NodeMCU) e sistemas de software IoT [De Paiva et al. 2023]. Seu objetivo é criptografar os dados gerados pelo dispositivo IoT e trafegado no contexto sistêmico, permitindo a transferência de dados encriptados e reduzindo, ao máximo, a possibilidade de vazamento ou manipulação indevida destes dados por outros sistemas ou pessoas. Como contribuição principal, este trabalho oferece a descrição do componente de criptografia para dispositivos IoT de baixo custo. Este trabalho também apresenta as observações experimentais (dois estudos baseados em simulação) sobre sua viabilidade e desempenho e, com isso, oferece uma opção para a mitigação de um dos problemas listados pela OWASP [OWASP 2019].

Este trabalho está organizado da seguinte forma. Os trabalhos relacionados são apresentados na seção 2; a seção 3 aborda o desenvolvimento e requisitos do *CryptoComponent*. A seção 4 descreve como foi feita a avaliação da segurança. Em seguida, a seção 5 apresenta a avaliação de eficiência do *CryptoComponent*. A seção 6 apresenta as limitações do componente desenvolvido e, finalmente, a seção 7 apresenta as considerações finais e trabalhos futuros.

2. Trabalhos Relacionados

O trabalho de Albalawi et al. (2018) apresenta a criação de uma arquitetura para assegurar a telemedicina em dispositivos IoT. Especificamente, os autores estudaram a comunicação segura garantindo a confidencialidade das mensagens entre os dispositivos e as aplicações. Entretanto, diferente de nossa proposta, eles usam um conjunto de serviços paralelos, como por exemplo para garantir a autenticação e autorização na troca da informação, utilização de encriptação assimétrica e simétrica para garantir a encriptação da mensagem, além de não possuir um componente único no dispositivo IoT que garanta a criptografia da mensagem.

Chanal et al. (2019) propuseram um algoritmo híbrido que utiliza a combinação do algoritmo AES (*Advanced Encryption Standard*), ECC (*Elliptic Curve Crypytography*) e MD5 (*Message-Digest Algoritm*), além de incluir uma criptografia baseada na localização do dispositivo IoT para garantir a confidencialidade dos dados. Porém, no artigo não é possível identificar que tipos de dispositivos IoT foram testados, se o software pode ser reutilizado em diferentes dispositivos, ou se as chaves de criptografia são específicas. Em contraste, nosso foco está em dispositivos IoT de baixo poder computacional, com componente reutilizável e permitindo a encriptação da mensagem por meio de uma chave específica.

Garg et al. (2022) estudaram uma arquitetura distribuída que utiliza múltiplos algoritmos de criptografia para classificar os dados e, consequentemente, identificar o algoritmo mais otimizado conforme a sensibilidade do dado. Contudo, diferente da solução oferecida em nosso trabalho, eles utilizam de serviços externos para classificação e necessitam de recurso computacional externo ao dispositivo IoT para encriptar a mensagem. Similar ao trabalho de Chanal et al. (2019), não oferecem um componente único para garantir a encriptação da mensagem.

Logo, dentre os trabalhos apresentados, não é possível observar uma solução que ofereça diferenciais tais como um componente de software para encriptação de dados em dispositivos IoT de baixo custo que não necessite de uma infraestrutura dedicada para seu funcionamento. Com base nisso, a utilização das soluções apresentadas na disponibilidade de recursos computacionais adicionais, o componente foi construído.

3. *CryptoComponent*

A construção do *CryptoComponent* envolveu a elicitação de requisitos para seu funcionamento. Considerando a natureza restritiva dos dispositivos IoT, foi necessário identificar os tipos de algoritmos de criptografia passíveis de serem aplicados para mitigar o problema de segurança durante o tráfego de dados no sistema. Nesse contexto, dentre diferentes algoritmos de criptografia, tais como o *Elephant*, *Pyjamask*, e *SPECK*, têm sido identificados na literatura técnica. O *SPECK* foi escolhido, tendo em vista ter sido criado para ser utilizado em dispositivos de baixo poder computacional, como por exemplo rede de sensores, incluindo dispositivos IoT [Beaulieu et al. 2015]. Sua escolha foi também influenciada pela simplicidade de utilização, na qual é necessário fornecer apenas uma chave única e uma mensagem de tamanho arbitrário para que o algoritmo produza uma mensagem encriptada. Além disso, o *SPECK* apresenta segurança avaliada e, por ser uma criptografia de bloco, apresenta baixo consumo de recurso computacional. Adicionalmente, para fins de avaliação do componente, foi implementado o algoritmo de XOR, escolhido por ser uma das etapas para construção do *SPECK* [Beaulieu et al. 2015].

Em seguida, os requisitos funcionais e não funcionais do *CryptoComponent* foram especificados. Dentre estes, destacam-se a capacidade do componente de encriptar o dado passado como parâmetro imediatamente antes de ser enviado pela rede, garantindo assim que tráfego de dados encriptados até seu recebimento na outra ponta e a capacidade do componente de decriptar o dado antes de ser consumido. Essa característica configura uma Criptografia *End to End*. A Tabela 1 mostra um extrato dos requisitos funcionais e não funcionais especificados para criação do *CryptoComponent*. Os demais requisitos e informações sobre escolha dos algoritmos estão descritos na sessão 2.5 de [De Paiva et al. 2023].

Após a definição dos requisitos, as escolhas para a implementação do projeto foram dadas visando garantir maior reutilização de código. Além disso, algumas decisões do projeto variaram entre a otimização do uso de recursos computacionais, tendo em vista a natureza do dispositivo e a facilidade da utilização por parte dos desenvolvedores, privilegiando a ampla utilização do componente. Sendo assim, por ser um componente para ser utilizado em dispositivos IoT, o *CryptoComponent* foi codificado na linguagem de programação C++. O componente pode ser utilizado em um dispositivo IoT por meio da instanciação de um objeto da classe *CryptoComponent* informando o método de encriptação de interesse (operação XOR ou algoritmo *SPECK*) e informando se a demanda é por Encriptar ou Descriptar a mensagem considerando uma Chave específica.

Tabela 1. Extrato dos requisitos funcionais e não funcionais do CryptoComponent

Código	**Descrição**
RF01	O componente deve ser capaz de criptografar e descriptografar mensagens de texto utilizando o *SPECK* e *XOR*
RNF01	O componente deve ser capaz de lidar com mensagens de texto em diferentes codificações de caracteres
RNF12	O componente deve ser capaz de executar em dispositivos de software IoT de classe zero e um

O *CryptoComponent* possibilita sua utilização pelos sistemas de software por meio de uma API (*Application Programming Interface*) que oferece acesso para as facilidades de encriptar e descriptar a informação independente da linguagem de programação. Neste caso, a comunicação entre os dispositivos IoT e o sistema de software deve ser realizada por meio de um *broker,* com protocolo MQTT, conforme apresentado na Figura 1.

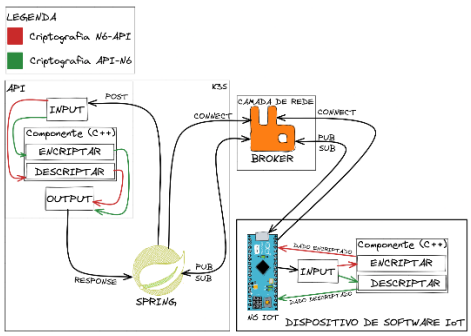


Figura 1. Fluxo de Utilização do CryptoComponent

A disponibilidade da API do *CryptoComponent* garante a possibilidade da encriptação/descriptação dos dados em todo o fluxo de recepção e envio da mensagem. O envio dos dados ao *broker* é feito por meio de rede pública utilizando o protocolo MQTT, logo, a utilização do componente nos dispositivos IoT garante a segurança do envio da mensagem. Da mesma forma, a segurança dos dados consumidos do *broker* é

garantida porque as aplicações que estão consumindo esses dados, em nosso caso, são executadas em K3S, um *kubernetes* de baixo custo, que garante que cada uma delas utilize uma rede privada, fazendo com que seu tráfego de rede não seja visto externamente.

Desta forma, pode-se dizer que os dados são transmitidos de forma segura e encriptada em cada um dos momentos da arquitetura apresentada na Figura 1. É importante ressaltar que a construção da API representa uma alternativa para garantir que o componente seja utilizado nos sistemas desenvolvidos no Laboratório de Engenharia de Software, não sendo necessário que outro usuário, por exemplo, tenha que utilizar o sistema K3S ou mesmo hospedar o serviço da API dentro de uma rede privada. Ver Capítulo 3 de [De Paiva et al. 2023] para informações adicionais.

4. Avaliação da Segurança do CryptoComponent

Para avaliar a viabilidade do *CryptoComponent* como instrumento de segurança em sistemas de software IoT, verificamos se ele realiza a criptografia dos dados enviados dos dispositivos IoT para o *broker*. Nesse contexto, a questão de pesquisa deste estudo é: "O *CryptoComponent* consegue manter um canal de comunicação de troca de mensagens encriptado entre as 'coisas' e o *broker*?" Desta forma, com base nesta questão de pesquisa e no uso do GQM (*Goal, Question, Metric*) [Basili et al., 1994], descrevemos o estudo como sendo **analisar** o *CryptoComponent* **com o objetivo de** *caracterizar* **com respeito à** sua capacidade de criptografar e descriptografar os dados **do ponto de vista dos** pesquisadores **no contexto de** dispositivos IoT de baixo custo enviando dados para um sistema de software IoT.

4.1 Contexto do Projeto do Sistema de Software IoT

O *CryptoComponent* foi aplicado inicialmente no sistema SAFE, um sistema de software IoT para realizar o monitoramento das condições de biossegurança e utilização das instalações da Universidade Federal do Rio de Janeiro. A construção deste sistema foi motivada no período da pandemia de Covid-19, momento em que diferentes preocupações de biossegurança foram trazidas à tona, tais como a necessidade de distanciamento pessoal, maiores cuidados com limpeza e higiene, e garantia de manter os ambientes ventilados.

O SAFE é composto por quatro subsistemas: dispositivos B(iosecurity)IoT, *broker*, *manager* e *dashboard*. O dispositivo BIoT é responsável por coletar as medidas dos sensores de CO_2, temperatura, umidade e número de pessoas em uma determinada instalação da universidade e enviar estas informações para o subsistema *broker*, que garante o tráfego de dados entre os dispositivos BIoT e o subsistema *manager*, desenvolvido em *Spring*, possibilitando o recebimento das medidas coletadas pelos sensores. Adicionalmente, o *manager* permite que os usuários gerenciem seus perfis, incluindo papéis institucionais, tendo em vista que cada papel possui um cargo associado com responsabilidades específicas, como por exemplo membro de equipe de limpeza ou de manutenção, o que faz com que receba uma notificação de limpeza ou manutenção toda vez que uma instalação é desocupada ou necessita reparo, preservando estas informações em seu banco de dados. O *dashboard* permite que os usuários observem as informações a respeito de cada instalação, obtendo informações como número de ocupantes em uma instalação, concentração de CO_2, temperatura e umidade em um determinado momento, além de informação sobre disponibilidade ou não da instalação devido a necessidade de limpeza ou manutenção.

4.2 Planejamento do Estudo

O estudo foi realizado para avaliar o uso do componente no dispositivo BIoT. Este trabalho é baseado em um estudo de simulação (*in silico*). Estes estudos são importantes como uma etapa inicial de avaliação de um produto ou processo, sistema de software, comportamento de equipes de desenvolvimento, tecnologias de software, dentre outros [França e Travassos 2016]. Seu objetivo foi analisar o fluxo de dados obtido pelo *manager* ao se conectar com o dispositvo BIoT por meio de um *broker* e, com isso, avaliar se o *CryptoComponent* está ou não criptografando as mensagens enviadas. Este estudo não envolveu a participação de humanos e nem provocou situações de risco nas instalações. Desta forma, não foi necessário coletar o consentimento ou realizar a caracterização dos participantes. O estudo é definido como um fator (software do dispositivo BIoT) e dois tratamentos (sem e com o *CryptoComponent)*. A H_0 para este estudo é: Não há diferença na segurança do sistema de software com ou sem o *CryptoComponent*.

4.3 Execução do Estudo

O estudo foi executado em quatro etapas. Etapa de avaliação analisou o tráfego de dados do dispositivo BIoT e verificou sua segurança. Etapa de implantação modificou o dispositivo BIoT e fez com que enviasse dados encriptados para o *broker*. Etapa de observação criou um simulador para se conectar ao mesmo *broker* do dispositivo BIoT e receber as informações. Etapa de análise analisou os dados relacionados ao tráfego de rede e informações recebidas utilizando instrumentos específicos.

O tráfego de mensagens do BIoT foi observado por meio do software *Wireshark*, com foco no tráfego da rede de um dispositivo para o *broker* seguindo as mesmas configurações utilizadas pelo dispositivo BIoT. Deste modo, foi possível reproduzir a requisição e verificar se existia algum tipo de criptografia prévia envolvida no sistema, como é possível observar na Figura 2.

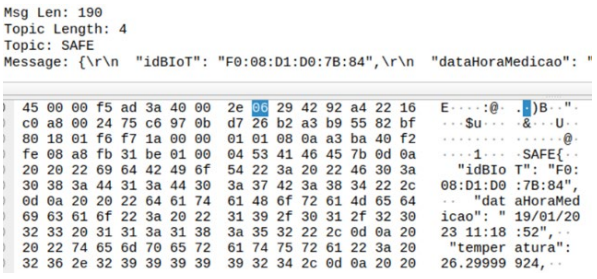


Figura 2. Análise do tráfego de subscribe no tópico SAFE

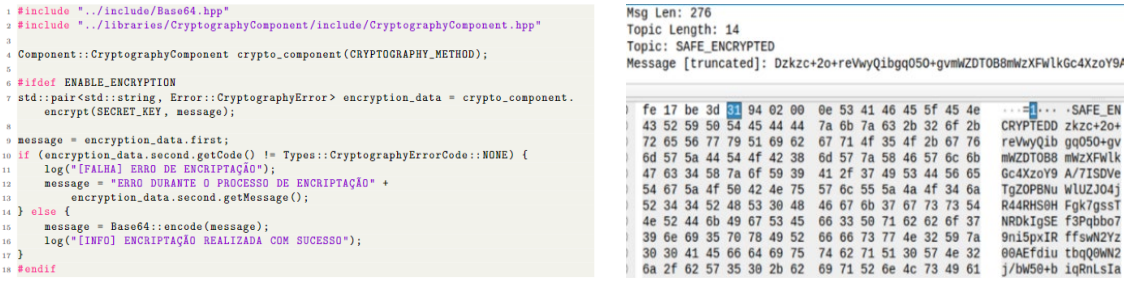
Para a reprodução da requisição, foi necessário simular o comportamento da API em *Spring* e realizar as requisições para o *broker* em MQTT, desta forma, utilizou-se a linguagem de programação Python e a biblioteca paho.mqtt para conexão com o tópico de interesse (i.e., SAFE) e realizar a conexão no mesmo *client* que é acessado pelo sistema SAFE. Para isso, foram informados o *host*, porta, e credenciais de acesso, tal como o tópico para que esta conexão fosse efetiva.

Após observar as requisições, foi possível perceber que os dados trafegaram de forma insegura, no qual estes estão em texto não encriptado, além de não se adequar aos requisitos do sistema SAFE e se tornando uma oportunidade de aplicação do componente de criptografia.

4.4 Avaliação do Componente

Para realizar a avaliação da utilização do componente, é necessário alterar o código que está executando no dispositivo BIoT, a fim de habilitar a utilização do componente e fazer com que os dados enviados ao *broker* sejam agora encriptados como é possível observar na linha 7 do trecho de código na Figura 3a. Uma vez habilitada, o componente irá retornar ao usuário final a mensagem encriptada, caso a criptografia ocorra com sucesso ou uma mensagem de erro caso contrário.

Assim, ao modificar o código do dispositivo BIoT, foi possível realizar o envio da informação encriptada para o *broker* e, por conta da geração de caracteres especiais, a fim de evitar problemas durante o envio e recebimento da mensagem por meio do *broker*, foi optado por converter o dado em base64 antes deste ser enviado a fim de evitar com que caracteres especiais impactassem a leitura do resultado por outros sistemas, mantendo a integridade do sistema. Após o processo de criptografia ter sido adequado ao dispositivo BIoT, o *payload* da requisição do *broker* MQTT foi modificado, como é possível observar na Figura 3b.



```
1  #include "../include/Base64.hpp"
2  #include "../libraries/CryptographyComponent/include/CryptographyComponent.hpp"
3
4  Component::CryptographyComponent crypto_component(CRYPTOGRAPHY_METHOD);
5
6  #ifdef ENABLE_ENCRYPTION
7  std::pair<std::string, Error::CryptographyError> encryption_data = crypto_component.
       encrypt(SECRET_KEY, message);
8
9  message = encryption_data.first;
10 if (encryption_data.second.getCode() != Types::CryptographyErrorCode::NONE) {
11     log("[FALHA] ERRO DE ENCRIPTAÇÃO");
12     message = "ERRO DURANTE O PROCESSO DE ENCRIPTAÇÃO" +
13         encryption_data.second.getMessage();
14 } else {
15     message = Base64::encode(message);
16     log("[INFO] ENCRIPTAÇÃO REALIZADA COM SUCESSO");
17 }
18 #endif
```

```
Msg Len: 276
Topic Length: 14
Topic: SAFE_ENCRYPTED
Message [truncated]: Dzkzc+2o+reVwyQibgqO5O+gvmWZDTOB8mWzXFWlkGc4XzoY9A

 fe 17 be 3d 31 94 02 00  0e 53 41 46 45 5f 45 4e   ···=1··· ·SAFE_EN
 43 52 59 50 54 45 44 44  7a 6b 7a 63 2b 32 6f 2b   CRYPTEDD zkzc+2o+
 72 65 56 77 79 51 69 62  67 71 4f 35 2b 67 76      reVwyQib gqO5+gv
 6d 57 5a 44 54 4f 42 38  6d 57 7a 58 46 57 6c 6b   mWZDTOB8 mWzXFWlk
 47 63 34 58 7a 6f 59 39  41 2f 37 49 53 44 56 65   Gc4XzoY9 A/7ISDVe
 54 67 5a 4f 50 42 4e 75  57 6c 55 5a 4a 4f 34 6a   TgZOPBNu WlUZJO4j
 52 34 34 52 48 53 30 48  46 67 6b 37 67 73 73 54   R44RHS0H Fgk7gssT
 4e 52 44 6b 49 67 53 45  66 33 50 71 62 62 6f 37   NRDkIgSE f3Pqbbo7
 39 6e 69 35 70 78 49 52  66 66 73 77 4e 32 59 7a   9ni5pxIR ffswN2Yz
 30 30 41 45 66 64 69 75  74 62 71 51 30 57 4e 32   00AEfdiu tbqQ0WN2
 6a 2f 62 57 35 30 2b 62  69 71 52 6e 4c 73 49 61   j/bW50+b iqRnLsIa
```

(a) *(b)*

Figura 3. Trecho de código para permitir a Encriptação da informação (a) e Análise do tráfego de subscribe no tópico SAFE_ENCRYPT (b)

Como meios de teste, em vez de utilizar o mesmo tópico em que as mensagens descriptadas estão sendo enviadas ao SAFE, foi criado um tópico, SAFE_ENCRYPTED, para o envio de mensagens encriptadas a fim de não impactar as aplicações que já consomem os dados do tópico.

No final da encriptação, a mensagem deixa de permitir observar o tráfego de forma simples, sendo necessário decodificar a mensagem para base64. Mesmo que este processo seja feito, obter a mensagem inicial sem a chave de descriptação e sem o algoritmo utilizado será bem difícil.

Desta forma, respondendo a questão de pesquisa, é possível inferir que o *CryptoComponent* tem a possibilidade de execução em dispositivos IoT de baixo custo no contexto de um projeto simulado, no qual tal fato pôde ser avaliado ao modificar o código do dispositivo IoT que envia os dados para o *broker*. Além disso, o fato do dado ser encriptado durante o seu tráfego pelo *broker* e ao ser obtido por meio do simulador ainda estar encriptado, também fortalece a proteção oferecida pela funcionalidade da criptografia. Mais informações sobre a simulação e avaliação do componente se encontram no Capítulo 4 de [De Paiva et al. 2023].

5. Avaliação da Eficiência do *CryptoComponent*

Para avaliar a eficiência do *CryptoComponent* em um contexto real de sistemas de

software IoT, entendendo que o componente é capaz de encriptar os dados, nós realizamos uma avaliação para verificar se o *CryptoComponent* realiza a criptografia dos dados enviados dos dispositivos para o sistema de software IoT de forma eficiente. Nesse contexto, a H_0 é: "*Não há diferença de desempenho do sistema de software SAFE para o usuário final usando ou não o CryptoComponent*"

Desta forma, o objetivo do estudo foi determinado utilizando o formato GQM da seguinte forma: **analisar** o *CryptoComponent* **com o objetivo de** *caracterizar* **com respeito à** eficiência (tempo de publicação e tempo de recebimento das mensagens pelo *broker*) para criptografar e descriptografar os dados **do ponto de vista dos** pesquisadores de sistemas IoT **no contexto de** dispositivos BIoT utilizados no sistema SAFE.

5.1 Planejamento do Estudo de Observação

Com a intenção de avaliar o *CryptoComponent* no sistema SAFE, foi realizado um estudo de observação que consiste na simulação do fluxo de dados do sistema, ou seja, o envio dos dados ao *broker* e o consumo de dados do *broker*, desta forma sendo possível avaliar a eficiência do componente visando responder à seguinte questão:

- *Os tempos em segundos de publicação e recebimento das mensagens (criptografadas e não criptografadas) pelo broker afeta o desempenho do SAFE?*

Aqui foram criados três trechos de códigos que englobam o processo de simulação (*in silico*). Um trecho de código que gera o JSON a ser enviado para o *broker*, seguindo a estrutura de dados gerada pelo sistema SAFE no dispositivo IoT; um trecho de código que permite a conexão entre o simulador e o *broker*, juntamente da API que engloba o *CryptoComponent* (responsável por encriptar e descriptar o dado baseado na necessidade do usuário), e; um trecho de código que simula a quantidade de dados a ser enviada, elegida pelo usuário.

5.2 Execução do Estudo

Para execução da simulação, tendo em vista que um dos requisitos do Sistema SAFE consiste no envio de mensagens para o *broker* de 15 em 15 segundos, foi escolhida a execução para o envio e recebimento de 80 mensagens geradas aleatoriamente com 200 *bytes* de dados (como especificado nos dispositivos BIoTs) em um ensaio em triplicata para observar o desvio padrão e tempo de resposta médio.

Portanto, o recebimento de 80 mensagens consiste na espera de 15 segundos para recebimento da informação e o envio da mensagem por meio do *broker*, no qual no exato momento do envio da 80° mensagem a execução do software é interrompida. Desta forma, cada execução de 80 mensagens em triplicata alcançaria um limite de tempo próximo de uma hora, mais especificamente 3555 segundos, tempo de execução do estudo.

5.3 Resultados do Estudo

Os resultados apresentados na Tabela 2 sugerem que o algoritmo de criptografia não irá alterar a experiência do usuário com essas configurações, o tempo total da comunicação com o broker, desconsiderando a execução do código, é de 1185 segundos. Por exemplo, para receber as mensagens de um tópico, usando o dado encriptado, o sistema necessita de 1196.49 segundos para processar 80 mensagens, ou seja, 15.14 segundos por mensagem, enquanto para o dado não encriptado, o sistema necessita de 1195.47 segundos, o que dá 15.13 segundos por mensagem. Este mesmo procedimento

pode ser repetido para o processo de publicação, no qual para o dado encriptado são necessários cerca de 1187.07 segundos para processar 80 mensagens, representando 15.026 segundos por mensagem enquanto para o dado não encriptado necessitamos de 1186.06 segundos para processar 80 mensagens, resultando em 15.013 segundos por mensagem.

Tabela 2. Resultados da Prova de Conceito.

Formato dos dados	Tempo (s) para 80 mensagens	Ação
Texto Puro	1195.47 ± 7.96s	*Subscribe* para o Tópico
Texto Puro	1186.06 ± 0.01s	*Publish* para o Tópico
Texto Encriptado	1196.49 ± 7.95s	*Subscribe* para o Tópico
Texto Encriptado	1187.07 ± 0.04s	*Publish* para o Tópico

Quando ambas as inscrições para os tópicos encriptados e não encriptados são comparadas é possível notar o aumento de 0.085% depois de obter a razão entre o tempo para inscrever no tópico do *broker* e receber 80 mensagens para o dado encriptado e não encriptado. Também, para o processo de publicação, a comparação entre os mesmos valores (dado encriptado e não encriptado), tem aumento proporcional para processar 80 mensagens. Assim, por meio de observação, a utilização do *CryptoComponent* no sistema SAFE sugere um baixo impacto na eficiência, provavelmente imperceptível pelo usuário.

6. Limitações do *CryptoComponent*

Esse componente possui algumas limitações, por exemplo, só é possível encriptar toda a mensagem mesmo que o usuário queira encriptar somente um campo específico da mensagem a ser enviada. Isso aceleraria o processo de encriptação e diminuiria, consequentemente a utilização de recurso computacional do dispositivo de software IoT.

O ensaio foi realizado de forma simulada para um único dispositivo enviando e recebendo a mensagem, contudo, é possível que, caso haja mais dispositivos trafegando informação, o aumento proporcional para processar 80 mensagens pode ser maior que para um único dispositivo.

O *CryptoComponent* foi avaliado utilizando somente uma composição do SPECK, no entanto, por ser uma família de algoritmos, é possível que exista outra configuração em número de *bytes* de mensagem e tamanho da chave que permita uma melhor eficiência na utilização deste algoritmo.

7. Conclusão e Trabalhos Futuros

O componente *CryptoComponent* tem como o objetivo disponibilizar o algoritmo de criptografia *SPECK* e o algoritmo de XOR para tornar sistemas de software IoT de baixo custo mais seguros. Adicionalmente, ele pode ser utilizado com pouco esforço de implementação, tendo em vista ser necessário instanciar apenas uma classe para encriptar os dados. De acordo com as avaliações experimentais realizadas com o componente, infere-se que o *CryptoComponent* causará impacto mínimo na eficiência dos sistemas em que for utilizado, sugerindo que a proteção no tráfego de dados não será percebida pelos usuários.

Este componente só tem suporte a algoritmos de cifra de bloco, desta forma, um dos possíveis trabalhos futuros é atuar na construção de outros tipos de algoritmos, por exemplo criptografia assimétrica, permitindo a utilização de encriptação/descriptação por validação de uma chave pública e privada, que são algoritmos mais utilizados atualmente.

Como trabalho futuro está prevista a execução de um estudo experimental do *CryptoComponent* em outros sistemas de software IoT para reforçar a evidência de sua viabilidade e eficiência. Além disso, nós pretendemos expandir o *CryptoComponent* para utilizar outras linguagens de programação como *Javascript*, C, dentre outra.

Agradecimentos

Este trabalho é parcialmente financiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior -Brasil (CAPES) - Código Financeiro 001 e CNPq pelo apoio financeiro para esta pesquisa. O Prof. Travassos é pesquisador do CNPq (305701/2022-3) e CNE FAPERJ (E-26/201.170/2021).

Referências

Albalawi, U. and Joshi, S. (2018). Secure and trusted telemedicine in internet of things iot. In 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), pp. 30–34.

Basili, V. R., Caldiera, G. and Rombach, H. D. (1994). The goal question metric approach. Encycl. Softw. Eng., 2, pp. 528–532.

Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B. and Wingers, L. (2015). Simon and speck: Block ciphers for the internet of things. Cryptology ePrint Archive.

Bradley, C., El-Tawab, S. and Heydari, M. H. (2018). Security analysis of an IoT system used for indoor localization in healthcare facilities. In (SIEDS), pp. 147–152.

Cisco (2020). Relatório anual da internet da cisco (2018-2023): White Paper. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (acesso em 20/9/2023).

Chanal, P. M. and Kakkasageri, M. S. (2019). Hybrid algorithm for data confidentiality in internet of things. In 10th (ICCCNT), pp. 1–5.

De Paiva, B. D., de Souza, B. P., and Travassos, G. H. (2023). CRYPTOCOMPONENT: UM COMPONENTE DE CRIPTOGRAFIA PARA SISTEMAS DE SOFTWARE IOT COM DISPOSITIVOS DE BAIXO CUSTO [Zenodo]. https://doi.org/10.5281/zenodo.8352063

De França, B. B. N. and Travassos, G. H. (2016). Experimentation with dynamic simulation models in software engineering: planning and reporting guidelines. Empirical Software Engineering, 21(3), pp. 1302-1345.

Garg, D., Rani, S., Herencsar, N., Verma, S., Wozniak, M. and Ijaz, M. (2022). Hybrid technique for cyber-physical security in cloud-based smart industries. Sensors, 22(12), 4630.

Motta, R. C., de Oliveira, K. M. and Travassos, G. H. (2018). On challenges in engineering IoT software systems. In Proceedings of the XXXII SBES, pp. 42-51.

Mousavi, S. K., Ghaffari, A., Besharat, S. and Afshari, H. (2021). Security of internet of things based on cryptographic algorithms: A survey. Wireless Networks, 27(2), pp. 1515–1555.

Ogonji, M. M., Okeyo, G. and Wafula, J. M. (2020). A survey on privacy and security of internet of things. Computer Science Review, 38:100312.

Othman, M. M. and El-Mousa, A. (2020). Internet of things cloud computing internet of things as a service approach. In ICICS, pp. 318–323.

OWASP (2019). OWASP Internet of Things (IoT) Project. https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project (acesso em 20/9/2023)

Rachit, S. B. and Ragiri, P. R. (2021). Security trends in internet of things: a survey. SN Applied Sciences, 3(1), pp. 1-14.

Rana, M., Mamun, Q. and Islam, R. (2022). Lightweight cryptography in IoT networks: A survey. Future Generation Computer Systems, 129, pp. 77–89.