Uma Arquitetura para a Distribuição Segura de Dados de Contexto em Aplicações de Internet das Coisas Móveis

An Architecture for Secure Context Data Distribution in Internet of Mobile Things Applications

André Luiz Almeida Cardoso
LSDi*
Universidade Federal do Maranhão
São Luís, Brasil
andre.cardoso@lsdi.ufma.br

Thiago Wallass Nascimento Mendes
LSDi
Universidade Federal do Maranhão
São Luís, Brasil
thiago.wallass@lsdi.ufma.br

Ariel Soares Teles
LSDi
Instituto Federal do Maranhão
Araioses, Brasil
ariel@lsdi.ufma.br

Francisco José da Silva e Silva
LSDi
Universidade Federal do Maranhão
São Luís, Brasil
fssilva@lsdi.ufma.br

RESUMO

A Internet das Coisas (IoT) está cada vez mais presente no cotidiano das pessoas. Neste paradigma, objetos do cotidiano podem se conectar à Internet, interagir entre si e trocar informações com pessoas. A IoT percorre diversos domínios como saúde e bem estar, logística, indústria, e cidades inteligentes. Os dados coletados nestes domínios são provenientes de diferentes dispositivos. Uma solução frequente para a heterogeneidade de dispositivos consiste no uso de *middleware*, o qual tem como objetivo servir de base para o desenvolvimento de aplicações IoT. Aplicações de IoT possuem requisitos de segurança. Entre as diversas soluções de *middleware* de IoT existentes, apenas algumas exibem esforços para prover mecanismos de segurança. O objetivo deste trabalho é facilitar a distribuição segura de dados de contexto em aplicações de Internet das Coisas Móveis (IoMT), por meio do desenvolvimento de um serviço de segurança. O serviço de segurança oferece mecanismos para o estabelecimento de comunicação segura, autenticação e controle de acesso. Experimentos foram conduzidos em uma aplicação móvel para comparar o custo de memória e processamento com e sem a utilização dos recursos de segurança. Os resultados confirmam que as funcionalidades de segurança podem ser utilizadas sem requerer excessivo custo computacional.

*Laboratório de Sistemas Distribuídos Inteligentes

Palavras-chave

Internet das Coisas, Computação Ubíqua, Segurança em Computação, Middleware

ABSTRACT

The Internet of Things (IoT) is increasingly present in people's daily lives. In this paradigm, everyday objects can connect to the Internet, interact with each other and exchange information with people. IoT operates in domains such as health and well being, logistics, industry, and smart cities. Data gathered in these domains comes from different devices. A frequent solution for the heterogeneity of devices is the use of middleware, which aims to serve as a basis for the development of IoT applications. IoT applications have security requirements. Among the various existing IoT middleware solutions, only a few show efforts to provide security mechanisms. The objective of this study is to facilitate the secure distribution of context data in Internet of Mobile Things (IoMT) applications, through the development of a security service. The security service provides mechanisms for establishing a secure communication channel, authentication and access control. Experiments were conducted in a mobile application to compare the cost of memory and processing with and without the use of security features. Results confirm that the security features can be used without requiring excessive computational cost.

Keywords

Internet of Things, Ubiquitous Computing, Computer Security, Middleware

CCS Concepts

•**Security and privacy** → **Security services;** •**Human-centered computing** → *Ubiquitous and mobile computing;*

1. INTRODUÇÃO

Atualmente, a Internet das Coisas (IoT) é cada vez mais utilizada em diferentes setores da sociedade [1]. Considerada um novo paradigma, a Internet das Coisas reúne diversos conceitos e tecnologias, tais como computação ubíqua e pervasiva, tecnologias de sensoriamento e comunicação. Através da IoT, dispositivos equipados com sensores e atuadores (por exemplo, *smartphones*) podem se interconectar de maneira que haja uma interação contínua entre o mundo real e o digital [2]. Sistemas IoT permitem que objetos sejam capazes de se conectar à internet para interagir entre si, atuar de acordo com as suas interpretações e trocar informações com pessoas. Estes objetos, sendo físicos ou apenas representações virtuais, constituem a base da IoT e são chamados de Objetos Inteligentes [3].

Considerando o fato de que tanto os objetos inteligentes quanto os *gateways* podem se mover, surge a necessidade de se estender o paradigma já existente. A Internet das Coisas Móveis (IoMT) é uma parte da IoT em que tanto os objetos quanto os *gateways* IoT podem ser movidos ou moverem-se de forma autônoma, permanecendo acessíveis e controláveis através da Internet [4]. A IoMT já está inserida de maneira significativa em diversos domínios de aplicação, tais como cidades inteligentes, segurança pública, logística, saúde e bem estar [5]. Um dos problemas recorrentes na IoT/IoMT consiste na heterogeneidade dos dispositivos. Objetos inteligentes podem variar bastante em relação às suas características de *hardware* e *software*, seja em tamanho, tecnologia de comunicação ou poder computacional. Uma das soluções adotadas para mitigar este problema é a utilização de *middleware*. Um *middleware* de IoT pode ser descrito como um *software* que atua como intermediário entre os dispositivos IoT e as aplicações [6]. Através da abstração de diferentes tecnologias, um *middleware* pode oferecer uma interface mais simples para o programador desenvolver suas soluções. O M-Hub/CDDL [7] é uma dessas soluções de *middleware*, a qual combina um *gateway* móvel com uma camada de distribuição de dados e visa facilitar o desenvolvimento de aplicações de IoMT [7].

Ao se desenvolver aplicações de IoT/IoMT, faz-se necessário pensar em segurança computacional, especialmente quando dados sensíveis estão envolvidos neste processo. É possível citar aplicações de IoT/IoMT que envolvem informações sensíveis, tais como dados médicos coletados remotamente através de sensores vestíveis em um paciente; uma indústria coletando dados de sensores para monitorar seus ambientes; uma delegacia de polícia utilizando a IoT para monitorar em tempo real a localização de suas viaturas. Aplicações como essas, apesar de úteis, podem ser inviabilizadas caso não haja segurança na distribuição dos dados coletados. Portanto, é fundamental garantir mecanismos para a segurança na comunicação e no acesso aos dados em aplicações de IoT e IoMT.

Este artigo é uma extensão do trabalho [8], em que foi proposto a adição de um serviço de segurança ao *middleware* M-Hub/CDDL. Composto por métodos para a criação de chaves criptográficas, gerenciamento e acesso de certificados digitais, o serviço de segurança, ao ser integrado ao M-Hub/CDDL, atua para garantir a distribuição segura de dados em aplicações de IoMT, com mecanismos para o estabelecimento de um canal seguro de comunicação, autenticação e controle de acesso. As contribuições desta versão estendida do artigo são: (i) um detalhamento da solução em relação à gerência de segurança, canal seguro de comunicação, autenticação e controle de acesso; (ii) uma avaliação de desempenho mais robusta, com maior duração e maior número de medições; e (iii) uma discussão detalhada sobre a solução em geral e a avaliação experimental.

A plataforma M-Hub/CDDL, que utiliza o *Message Queuing Telemetry Transport*[1], ou MQTT, para a distribuição de dados de contexto, originalmente não fornece recursos de segurança, impossibilitando o uso do M-Hub/CDDL no desenvolvimento de aplicações que utilizem dados sensíveis e requeiram mecanismos para garantir segurança nos dados que trafegam. Neste sentido, o *middleware* M-Hub/CDDL torna-se uma plataforma adequada para implementar o modelo de segurança proposto. Através da implementação da solução proposta, é possível estabelecer um canal seguro de comunicação e mecanismos de autenticação e controle de acesso. Além disso, o serviço de segurança também é responsável pela gerência dos elementos necessários (ou seja, par de chaves, certificados digitais e lista de controle de acesso) para o funcionamento dos mecanismos citados. Dessa forma, facilita-se a distribuição segura de dados entre instâncias do *middleware*.

O restante deste artigo está organizado da seguinte forma. A Seção 3 discute os trabalhos relacionados à segurança em soluções de *middleware* de IoT. A Seção 2 sumariza os conteúdos preliminares necessários para o entendimento do trabalho. A Seção 4 detalha a solução proposta, enquanto a Seção 5 apresenta uma avaliação de desempenho que compara o custo computacional de memória e processamento com e sem a utilização dos recursos de segurança. A Seção 6 discute os resultados. Por fim, a Seção 7 conclui o trabalho e apresenta as perspectivas de trabalhos futuros.

2. PRELIMINARES

Esta seção descreve os conceitos preliminares necessários para o entendimento da solução proposta. É apresentado o *middleware* M-Hub/CDDL, o qual foi utilizado como plataforma para implementar os mecanismos de segurança propostos.

Esta solução de *middleware* é desenvolvida em parceria entre o Laboratório de Sistemas Distribuídos Inteligentes (LSDi) da Universidade Federal do Maranhão (UFMA) e o *Laboratory for Advanced Colaboration* (LAC) da Pontifícia Univesidade Católica do Rio de Janeiro (PUC-Rio). O *middleware* é composto por dois componentes de *software*: o *Mobile Hub* (M-Hub) e o *Context Data Distribution Layer* (CDDL), os quais são detalhados a seguir.

2.1 O middleware M-Hub/CDDL

O M-Hub/CDDL possui recursos para a aquisição, processamento e distribuição de dados de contexto, oferecendo suporte para o desenvolvimento de aplicações de IoT/IoMT [7]. Seu funcionamento ocorre através da união entre um *gateway* móvel e uma camada de distribuição de dados. Utilizando o *middleware*, torna-se possível descobrir, registrar e enviar dados entre objetos inteligentes. O componente M-Hub atua como *gateway* móvel para a aquisição dos dados, enquanto o CDDL é o responsável pela distribuição dos dados com suporte de qualidade de contexto [9]. Devido à escassez de recursos, muitos objetos inteligentes não implementam a pilha de protocolos TCP/IP, não sendo ca-

[1] `https://mqtt.org/`

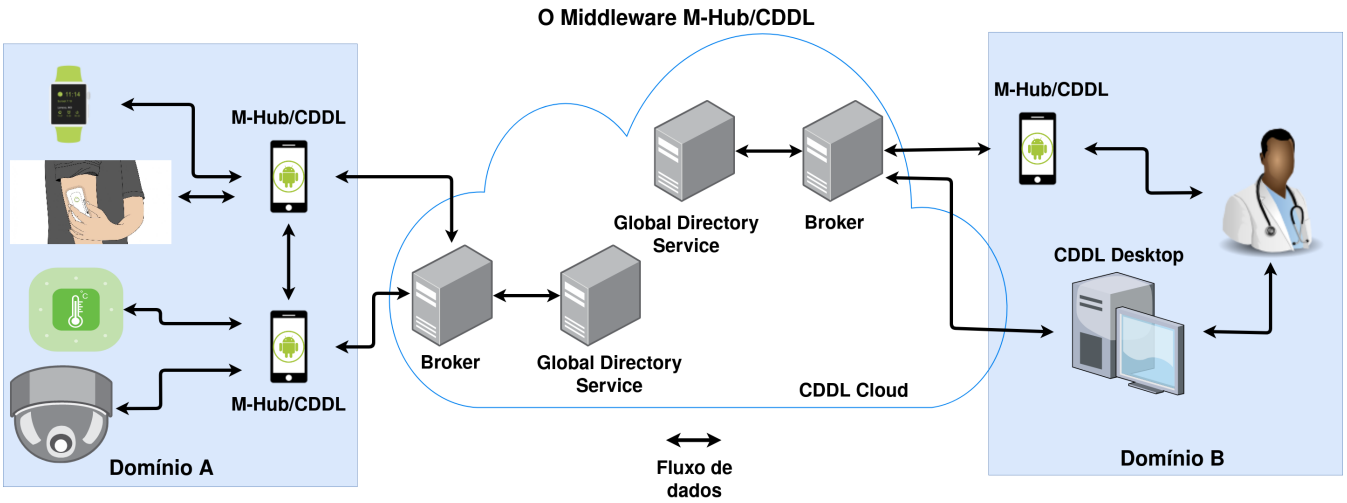

Figura 1: Domínios de Atuação do M-Hub/CDDL. Adaptado de [7].

pazes de se conectar à internet. Assim, o M-Hub/CDDL tem capacidade para transformar objetos equipados apenas com comunicação de curto alcance em objetos inteligentes, garantindo a eles conexão com a Internet e poder de processamento. Isto ocorre quando um simples sensor tem seus dados coletados pelo M-Hub via *Bluetooth*, então tais dados podem ser previamente analisados e distribuídos na Internet via CDDL.

Ambos componentes (M-Hub e CDDL) são independentes entre si. Em [4], utiliza-se o M-Hub em conjunto com outra camada de distribuição de dados chamada SDDL [10]. Para este estudo, utilizou-se o M-Hub integrado ao CDDL. O M-Hub atua em dispositivos pessoais (ou seja, *smartphones*) que utilizam o sistema operacional *Android*. Já o CDDL possui três tipos de clientes: Móveis (*Android*), *Desktop* e Web. Estes componentes (M-Hub/CDDL), ao serem utilizados em conjunto, permitem *smartphones* coletar dados de sensores, enriquecê-los com informação de contexto, e transmiti-los entre instâncias móveis do M-Hub/CDDL ou instâncias que operam na nuvem.

A Figura 1 ilustra os domínios de atuação do *middleware*, facilitando o entendimento sobre como ele pode ser utilizado em aplicações reais para coleta e distribuição de dados. As setas representam o fluxo de dados, sendo possível a comunicação entre sensores e M-Hub, e entre diferentes instâncias do CDDL. No domínio A, temos objetos inteligentes equipados com capacidade de comunicação de curto alcance, como por exemplo: *Bluetooth* clássico e *Blutooth Low Energy*.

O M-Hub/CDDL, em sua versão móvel, é capaz de se comunicar com tais objetos para a coleta de dados e o envio de comandos de atuação. Além disso, os dados podem ser distribuídos entre instâncias do *middleware*, móveis ou na nuvem, ou para qualquer *broker* MQTT. Observando a imagem da Figura 1, é exemplificada uma aplicação de IoT em Saúde e Bem Estar, em que no primeiro domínio tem-se um paciente equipado com sensores vestíveis ou com sensores em seu ambiente, e o M-Hub/CDDL sendo utilizado para a leitura dos dados e distribuição destes para a nuvem. No domínio B tem-se o profissional de saúde utilizando o M-Hub/CDDL, ou apenas o CDDL em sua versão *Desktop*, para a recepção e análise dos dados de seus pacientes.

2.2 Mobile Hub

O M-Hub é o componente que atua como intermediário entre sensores, atuadores e a conexão com servidores na nuvem (ou seja, via CDDL). Ele é capaz de descobrir, registrar e coletar dados provenientes de sensores, assim como enviar comandos de atuação aos mesmos. A comunicação entre M-Hub e sensores é feita por meio da abstração de diferentes tecnologias de comunicação de curto alcance. Esta abstração é responsabilidade de uma interface, denominada `Short-range Sensing, Presence and Actuation` (S2PA) e implementada no M-Hub. A função dela é atuar como protocolo para a comunicação de curto alcance com objetos móveis [4]. Atualmente, a S2PA implementa comunicação via *Bluetooth* 4.0 (BLE), *Bluetooth* clássico, e consegue acessar os dados provenientes de sensores internos de *smartphones Android*. A interface S2PA é extensível, de forma que o suporte a outras tecnologias de comunicação de curto alcance podem ser implementadas.

Os *smartphones Android* são dispositivos móveis e, apesar de limitados, possuem certo poder de processamento e conectividade. Ao rodar o M-Hub, estes podem ser transformados em *gateways* móveis, para serem utilizados como soluções de IoT/IoMT, capazes de coletar dados de dispositivos vestíveis, veículos, sensores, dentre outros, e ainda enviar comandos de atuação. Os dados coletados podem ser consumidos e processados no próprio dispositivo ou transmitidos através de algum protocolo de comunicação ou camada de distribuição de dados.

As versões mais recentes do M-Hub contam com diversas melhorias para problemas específicos. Por exemplo, existem situações em que o mesmo objeto inteligente é elegível para ser monitorado por diversos M-Hubs independentes. Para lidar com esse problema, em [11] é proposto o *Neighborhood-aware Mobile Hub*, ou NAM-Hub, em que é implementado no M-Hub um mecanismo para eleição de líderes. Dessa forma, diversos NAM-Hubs entram em consenso sobre quais objetos cada um deve monitorar, evitando assim o desperdício de recursos que ocorreria caso objetos fossem monitorados de forma redundante por M-Hubs. Para decidir qual NAM-Hub é mais adequado para cada objeto, utiliza-se as informações

de contexto coletadas do próprio dispositivo móvel.

2.3 CDDL

A camada de distribuição de dados de contexto (ou seja, o CDDL) pode ser utilizada de maneira integrada ao M-Hub para transmissão e enriquecimento dos dados coletados. Em [4], o M-Hub é utilizado juntamente a outra camada de distribuição de dados, o *Scalable Data Distribution Layer* - SDDL [10], para facilitar o desenvolvimento de soluções de IoT. O uso do M-Hub não é condicionado ao CDDL, isto é, ambos os componentes são independentes e podem ser integrados a outras tecnologias. O M-Hub pode ser utilizado com outras tecnologias para distribuição de dados. Analogamente, o CDDL pode utilizar outros mecanismos para detecção, descoberta e leitura de dados oriundos de sensores e/ou objetos inteligentes. Portanto, as soluções são fracamente acopladas. Para este estudo, optou-se por utilizar o M-Hub integrado ao CDDL como *middleware* (ou seja, M-Hub/CDDL) de IoT.

O CDDL utiliza internamente o MQTT como protocolo de comunicação para a transferência de dados, seja em conexões locais ou remotas [7]. As conexões CDDL atuam por meio do encapsulamento de clientes MQTT em um componente chamado `ConnectionImpl`. Além disso, em cada instância do CDDL existe um *broker* MQTT interno, encapsulado em um componente denominado `Microbroker`. Dessa maneira, torna-se possível distribuir dados apenas dentro de uma mesma instância, entre diferentes instâncias do CDDL e entre uma instância do CDDL e qualquer *broker* MQTT externo. Os componentes `ConnectionImpl` e `Microbroker`, citados a cima, podem ser observados na Figura 2, que exibe a arquitetura do M-Hub/CDDL e destaca o fluxo de dados entre seus componentes internos.

O M-Hub, através da interface S2PA, abstrai diferentes tecnologias, permitindo assim a coleta de dados provenientes de objetos inteligentes. Uma vez coletados, o componente `QoCEvaluator` do CDDL é responsável por enriquecer os dados com informações de contexto, tais como latitude, longitude, altitude, acurácia, e *timestamp*. O enriquecimento é possível através de medições e acesso aos dados de geolocalização do *smartphone Android*. O componente `MonitorImpl` e `FilterImpl` utilizam o Processamento de Eventos Complexos (CEP) [12] para operar sobre dados coletados a partir do M-Hub. O monitoramento de dados permite receber alertas quando determinadas regras forem ativadas, enquanto os filtros permitem limitar o recebimento de informações de acordo com condições pré-estabelecidas. Por fim, o CDDL oferece interfaces que são utilizadas por aplicações e programadores e, através delas, pode-se acessar os recursos do CDDL de maneira simples.

2.4 Problemática Abordada

Em alguns domínios de atuação da IoMT, como Saúde e Bem Estar, os dados são especialmente sensíveis. Portanto, é necessário garantir a segurança na comunicação e no acesso a eles. Em [13], apresenta-se uma infraestrutura de IoT, utilizando o M-Hub e SDDL, para desenvolvimento de aplicações de *Ambient Assisted Living* (AAL). *Ambient Assisted Living* é um campo de estudo que objetiva desenvolver sistemas para monitoramento da saúde de pacientes idosos em suas casas, de forma a aumentar a independência destes durante o tratamento de doenças crônicas [13]. Não somente idosos, mas também pacientes com doenças crônicas ou pessoas que realizam atendimento em suas casas, não necessitando se deslocar para receber atendimento em saúde

A implantação de soluções em cenários como este de *Ambient Assisted Living* pode ser inviabilizada caso não exista segurança na distribuição dos dados. Dessa forma, faz-se necessário adotar mecanismos para facilitar a distribuição segura de dados em aplicações de IoT e IoMT. A plataforma M-Hub/CDDL, escolhida para implementar a solução de segurança, não fornece mecanismos suficientes para garantir todos os requisitos de segurança impostos (ou seja, canal seguro de comunicações, autenticação e controle de acesso) pelas aplicações da IoMT [14].

3. TRABALHOS RELACIONADOS

Na literatura, existem diversos trabalhos relacionados à segurança e soluções de *middleware* para IoT. Em [14], é apresentado um estudo que revisa e classifica os problemas de segurança em IoT. Também são apresentados os requisitos gerais para a segurança em IoT juntamente com os ataques, ameaças e soluções já existentes. Os principais requisitos de segurança para IoT citados pelos autores são: privacidade, confidencialidade, integridade, autenticação, autorização e auditoria. A segurança em soluções de *middleware* de IoT é classificada como problema de segurança de alto nível.

Em [6], o *middleware* é destacado como uma tecnologia chave para o desenvolvimento de sistemas IoT, devido à sua capacidade de atuar como intermediário entre dispositivos e aplicações. É apresentado um *survey* sobre as capacidades das soluções de *middleware* existentes e os principais desafios são levantados. Garantir a segurança de aplicações IoT é considerada pelos autores como um dos grandes desafios para o desenvolvimento de soluções de *middleware* para a IoT. Assim como o trabalho de Khan et al. [14], Ngu et al. [6] destaca que as soluções de *middleware* devem possuir algumas propriedades para assegurar a segurança e confiança. A segurança pode ser obtida através de confidencialidade, integridade e disponibilidade. Já a confiança está relacionada à autenticação, autorização e auditoria.

Uma arquitetura de segurança para um *middleware* transparente de IoT é apresentada em [15]. Com o objetivo de representar objetos reais no meio virtual, a arquitetura proposta utiliza tecnologias de segurança já existentes e bem estabelecidas, tais como AES (*Advanced Encryption Standard*), TLS (*Transport Layer Security*) e *oAuth*. Através da integração de tais tecnologias no *middleware*, garante-se a privacidade, autenticidade, integridade e confidencialidade nos serviços de troca de dados. Não foram realizados experimentos para avaliar a solução de segurança proposta.

O *middleware* VIRTUS é apresentado em [16]. Seu desenvolvimento foi realizado levando em conta questões de segurança. O VIRTUS utiliza o protocolo *open* XMPP (*Extensible Messaging and Presence Protocol*) e tem como objetivo prover a comunicação segura baseada em eventos, em cenários de IoT. Utilizando as funcionalidades de segurança já ofertadas pelo XMPP, o VIRTUS consegue oferecer um canal confiável e seguro de comunicações para aplicações distribuídas. A autenticação é feita através do protocolo TLS e, para a criptografia, foi utilizado o protocolo *Simple Authentication and Security Layer* (SASL). A avaliação ocorreu através de um estudo de caso, em que o VIRTUS foi utilizado em uma aplicação da área de Saúde e Bem Estar. Os sinais vitais de um paciente são medidos por diversos instrumentos, os dados eram codificados no formado XML e inclusos

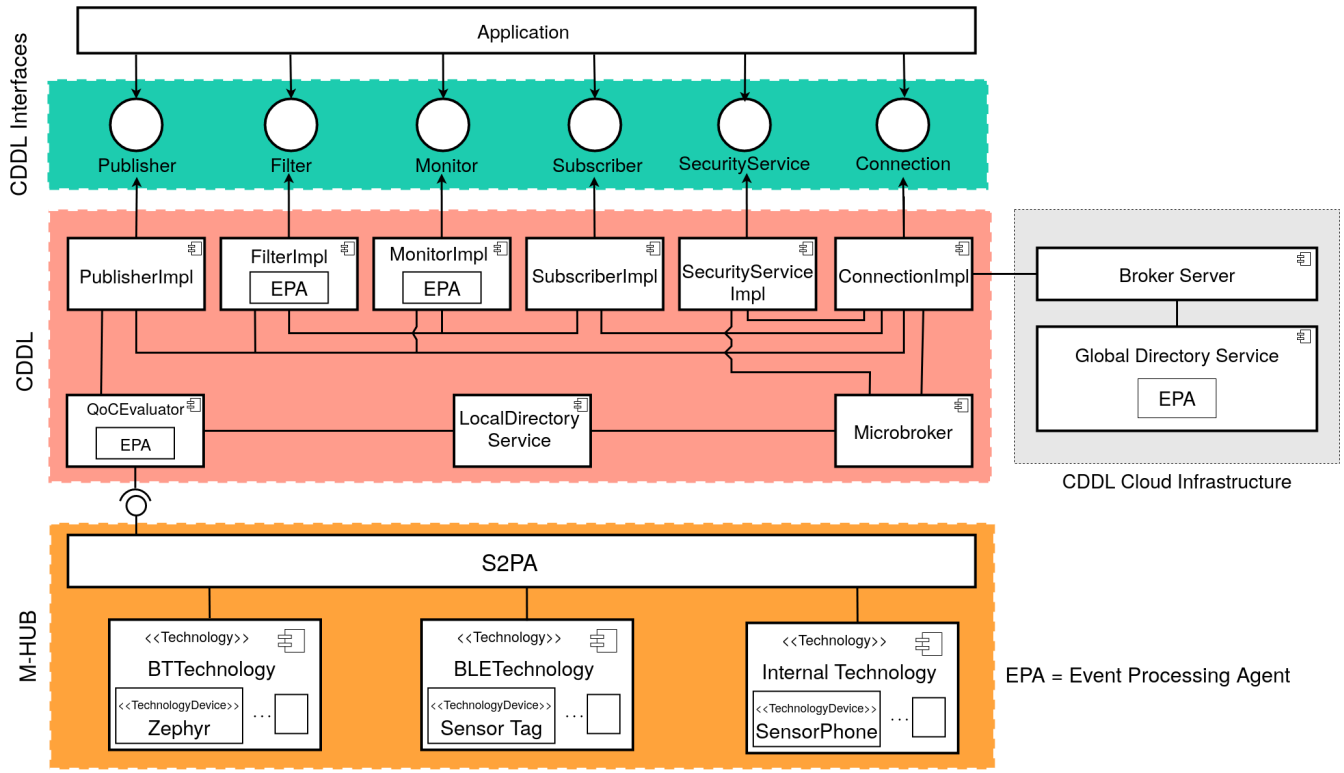

Figura 2: Nova arquitetura do M-Hub/CDDL(SS), com a adição do serviço de segurança. Adaptado de [7].

em mensagens XMPP. Dessa forma, os dados podem ser distribuídos entre instâncias do *middleware*. O estudo de caso demonstra que a solução é escalável, conseguindo manter instâncias do *middleware* para atender um alto número de pacientes.

A adição de recursos de segurança ao *middleware* M-Hub/CDDL o torna vantajoso em relação aos outros. Conforme pode ser visto em [17], apesar das diferentes tecnologias e objetivos de cada *middleware*, poucos possuem mecanismos para garantir a segurança. A solução apresentada nesse artigo, integrada ao M-Hub/CDDL, se alinha às atuais soluções de segurança de *middleware* IoT, e diferencia-se por atuar em um *gateway* móvel, em que os recursos são restritos. Considerando este diferencial, pode-se utilizar um *smartphone* pessoal, de maneira segura, como *gateway* em soluções de IoMT. Por funcionar em um *middleware publish/subscribe* baseado em tópicos, a camada de segurança também permite que se estabeleça uma conexão segura com *brokers* MQTT que suportem o TLS (*Transport Layer Security*), aumentando sua aplicabilidade.

4. SOLUÇÃO PROPOSTA: UM SERVIÇO DE SEGURANÇA

A solução proposta, chamada de *Security Service* (SS), oferece métodos que podem ser integrados ao M-Hub/CDDL para acrescentar uma camada de segurança ao *middleware*. Seu objetivo é garantir mecanismos para o estabelecimento de um canal seguro de comunicação, autenticação e controle de acesso. O funcionamento destes recursos de segurança ocorre por meio de certificados digitais, chaves criptográficas e lista de controle de acesso. Além disso, o SS também visa facilitar o processo de geração de certificados digitais, aumentando sua versatilidade e contribuindo para o desenvolvimento de aplicações seguras. Durante a concepção do *Security Service*, a seguinte política de segurança foi considerada:

- O SS deve manter dois certificados digitais, um para o usuário e outro para a autoridade certificadora;
- O SS deve facilitar a geração e a importação de certificados digitais;
- A troca de mensagens deve ser criptografada;
- A autenticação sempre deve ser mútua em uma conexão;
- O controle de acesso deve, por padrão, recusar todas as mensagens, a não ser que a permissão seja garantida explicitamente pelo usuário;
- Por fim, o uso do *middleware* com os recursos de segurança deve ser opcional, permitindo a não utilização do SS.

Para estabelecer o canal seguro de comunicação e realizar autenticação mútua entre as partes comunicantes, optou-se por utilizar certificados digitais. Esta decisão fundamenta-se no fato de que diversos protocolos de comunicação utilizados na IoT/IoMT (por exemplo, MQTT, XMPP, HTTP) já oferecem suporte para o TLS. Através do TLS, a troca de mensagens é, por padrão, criptografada, e durante o processo de *Handshake* é efetuada a autenticação simples ou mútua. A confiança entre as partes durante a autenticação

é estabelecida através do certificado da autoridade certificadora. A conexão ocorre entre dois clientes somente caso eles apresentem certificados confiáveis para ambas as partes.

A Figura 2, ilustra a nova arquitetura do M-Hub/CDDL com a adição do *Security Service*. As linhas representam o fluxo de dados interno ao M-Hub/CDDL. Adicionamos o componente `SecurityServiceImpl`, que se comunica diretamente com os componentes `ConnectionImpl` e `Microbroker`. Esta comunicação ocorre devido a eles serem os componentes que precisam do contexto SSL, fornecido pelo SS, para estabelecer a comunicação segura. Por fim, ao topo da imagem temos as interfaces que o CDDL disponibiliza para que as aplicações consumam os recursos do SS e os demais recursos do M-Hub/CDDL.

4.1 Gerência da Segurança

A principal funcionalidade do SS consiste na capacidade de gerenciar os elementos necessários para estabelecer a comunicação segura. A Figura 3 apresenta a interface implementada no CDDL para manter e gerenciar os elementos de segurança. O método `generateKeyPair()` gera e armazena um par de chaves criptográficas RSA (*Rivest-Shamir-Adleman*) [18]. O par de chaves é utilizado pelo segundo método `generateCSR()` para obter a requisição de assinatura de certificado (*Certificate Signing Request* - CSR). A requisição deve ser assinada por uma autoridade certificadora de confiança e então importada no SS através do método `setCertificate()`. O certificado da autoridade confiável deve ser importada pelo método `setCACertificate()`. Os métodos `getCertificate()` e `getCaCertificate()` acessam facilmente os certificados armazenados pelo SS, e podem ser utilizados para verificar quais os certificados importados. Por fim, o método `getSSLContext()` acessa a chave privada, o certificado do usuário e o certificado da autoridade certificadora confiável para, então, criar o contexto utilizado pelo TLS. A chave privada apenas pode ser acessada de dentro do SS, não sendo disponibilizado nenhum método público na interface para obtê-la. Dessa forma, garante-se a integridade da chave privada. Tais métodos, oferecidos pela interface, possibilitam configurar os certificados digitais necessários para utilizar o M-Hub/CDDL com segurança.

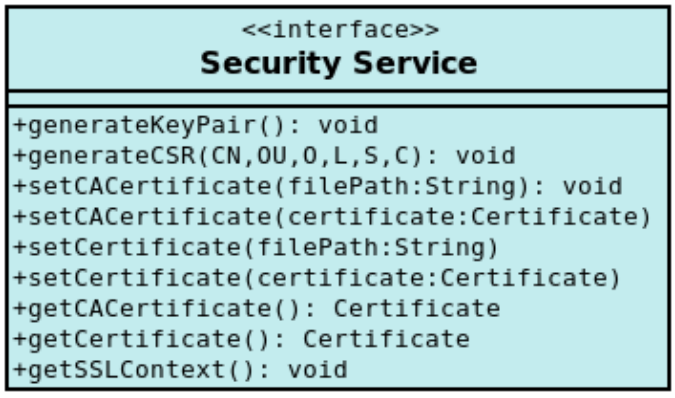

Figura 3: Métodos da interface relacionados ao gerenciamento de segurança.

Para armazenar as chaves e certificados de maneira segura, foi levado em consideração o ambiente de funcionamento do CDDL. Para as instâncias móveis do M-Hub/CDDL, o SS utiliza o armazenamento interno do *Android*. Nesse caso, o sistema operacional garante que apenas a aplicação pode acessar os arquivos que ela gerou. Para as instâncias *Desktop*, foi utilizado uma *KeyStore* do tipo PKCS12 [19] com senha.

4.2 Canal Seguro de Comunicação e Autenticação

Conforme já discutido no inicio desta seção, optou-se por utilizar o TLS para estabelecer o canal seguro de comunicação e autenticação. Como o SS já possui todos os elementos necessários para estes mecanismos de segurança, apenas foi necessário integrar o SS ao CDDL de forma que o contexto do TLS seja levado aos componentes certos. O transporte de dados no CDDL ocorre por meio de dois componentes, `Microbroker` e `ConnectionImpl`. Para utilizá-los de forma segura e opcional, criou-se dois novos métodos de inicialização. A Figura 4 ilustra os novos métodos de inicialização, `secureConnect()` e `secureStart()` que, respectivamente, levam de forma automática o contexto *Secure Sockets Layer* (SSL) [20] para o cliente e *broker* MQTT interno ao CDDL. Com essas alterações, torna-se possível utilizar o `Microbroker` e se conectar a ele de forma segura e opcional.

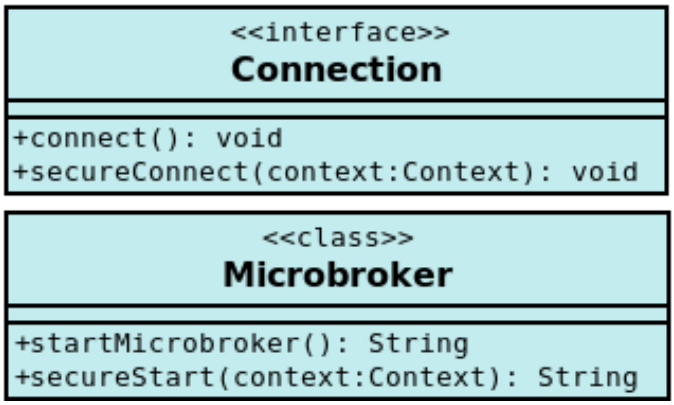

Figura 4: Novos métodos de inicialização.

4.3 Controle de Acesso

O controle de acesso determina quais dados um usuário conectado pode acessar. Uma forma comum de efetuar tal controle, em protocolos como o MQTT, é através de uma lista de controle de acesso. O CDDL possui uma estrutura predefinida de tópicos baseada em serviços, em que o usuário apenas define quais serviços quer publicar e/ou subscrever dados, e o CDDL determina os tópicos automaticamente. Para gerenciar o acesso em uma instância do CDDL, é necessário criar regras que determinem quais serviços podem ser acessados por quais clientes. Através de sua interface, o SS oferece métodos para a geração de regras. A Figura 5 exibe os métodos disponibilizados pelo SS para adicionar e remover as regras.

Utilizando os métodos acima, é possível conceder e revogar permissões para os clientes que acessam uma instância do CDDL. Ao adicionar regras utilizando os métodos acima, elas são armazenadas pelo SS em um arquivo de texto. Tais regras são geradas de acordo com o seguinte padrão: "IdentificadorDoCliente tópico permissão". A Figura 6 exempli-

```
<<interface>>
Security Service
+grantPermissionByServiceName(clientID:String,
                              serviceName:String,
                              permission:String): void
+removePermissionByServiceName(clientID:String,
                               serviceName:String,
                               permission:String): void
```

Figura 5: Modificação dos componentes `Connection` e `Microbroker` para garantir o canal seguro de comunicação.

fica o resultado obtido ao se adicionar regras de controle de acesso. A primeira regra permite que o usuário com ID "usuarioXYZ" se subscreva para a leitura de dados do sensor "XYZ" na instância do M-Hub/CDDL que adicionou a regra. Analogamente, a segunda regra permite a publicação dos dados pelo usuário.

```
Regras.txt

usuarioXYZ mhub/+/service_topic/sensorXYZ read
usuarioXYZ mhub/usuarioXYZ/service_topic/sensorXYZ write
```

Figura 6: Exemplo de regras geradas utilizando o `Security Service`.

O `Microbroker` oferece uma interface autorizadora que possui dois métodos: `canRead` e `canWrite`. Esses métodos são chamados automaticamente no *broker* quando um cliente tenta publicar ou subscrever em tópicos. Assim, para integrar o SS ao CDDL, implementou-se esta interface autorizadora e seus métodos de forma que busquem e utilizem regras armazenadas no SS para permitir ou bloquear o acesso durante a publicação e a subscrição de dados.

5. AVALIAÇÃO DE DESEMPENHO

5.1 Objetivo e Metodologia

Em [8], foi feita uma avaliação preliminar de desempenho da solução proposta, em que utilizamos uma aplicação para gestão de presença e encontros de pessoas em ambientes fechados [21]. Esta aplicação utiliza *beacons Bluetooth Low Energy* (BLE) para representar espaços físicos e *smartphones*, rodando o M-Hub/CDDL, para representar pessoas. Quando o *middleware* detecta a presença de um *beacon*, é possível inferir a posição da pessoa através da identificação do *beacon* detectado. A frequência da detecção é capaz de perceber aproximadamente 5 *beacons* por segundo. Após armazenar o resultado de dez detecções, a aplicação móvel identifica o *beacon* de sinal mais forte e envia, através do M-Hub/CDDL, a localização a um servidor rodando uma instância do CDDL *Desktop*. Uma vez que a localização é enviada, as dez detecções são descartadas e este processo se inicia novamente. Por ser o componente do M-Hub/CDDL com restrição de recursos, avaliou-se a instância móvel objetivando medir o desempenho da aplicação com e sem a utilização da camada de segurança. Naquela avaliação, foi utilizado um *smartphone* do modelo Moto G8 Plus, o qual possui as seguintes especificações: sistema operacional Android 9, processador *Qualcomm Snapdragon* 665 (4 x 1.8GHz), 4 GB de memória RAM, e 64 GB de armazenamento em disco. A avaliação consistiu em utilizar o *Android Profiler* [22] para verificar o consumo total de memória e CPU pela aplicação. O experimento durou 25 minutos e as medições foram verificadas a cada intervalo de 1 minuto. Os resultados preliminares apresentados em [8] indicaram que houve um aumento no consumo de memória e processamento ao se utilizar a camada de segurança.

Para consolidar os resultados preliminares, optou-se por uma nova avaliação, dessa vez mais extensa. Para a nova avaliação, utilizou-se a mesma aplicação móvel, porém os dados agora foram enviados a uma instância local do *Broker* MQTT Mosquitto [23]. Isto foi possível pois o `Mosquitto` oferece suporte para o TLS. Os certificados digitais do *Broker* foram configurados corretamente para a execução deste experimento e, para facilitar o processo de geração dos certificados da aplicação móvel, utilizou-se os métodos do SS. A duração do experimento aumentou para 60 minutos, e os intervalos de medições foram reduzidos. Em contraste com o experimento anterior, onde foram feitas apenas 25 medições, agora foram efetuadas 3090 medições em um intervalo de uma hora. A frequência foi aproximadamente de uma medição a cada 1,16 segundos. Dessa forma, os resultados se tornam mais consistentes uma vez que um maior número de medições estima de maneira mais precisa o comportamento da solução proposta. Nesta nova avaliação, foi utilizado um *smartphone* do modelo Moto G8 Power Lite, o qual possui as seguintes especificações: sistema operacional Android 10, processador MediaTek Helio P35 (octa-core de até 2,3 GHz), 4 GB de memória RAM, e 64 GB de armazenamento em disco. As medições foram coletadas a partir de *scripts* em Python utilizando a ferramenta *dumpsys* [24], acessada via linha de comando através do *Android Debug Bridge* (ADB) [25]. O experimento foi repetido duas vezes, onde mediu-se o consumo de memória e processamento da aplicação com e sem a utilização do SS.

5.2 Resultados

A Figura 7 apresenta os resultados do consumo de memória em Megabytes (MB). A Figura 8 apresenta o resultado do consumo de processamento em termos do percentual total da capacidade de processamento do dispositivo.

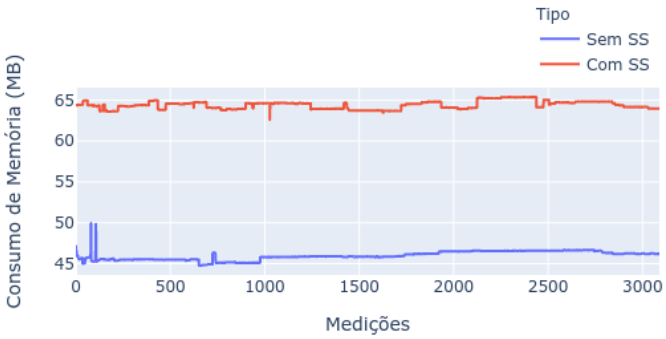

Figura 7: Consumo de memória com e sem a utilização do SS.

Para facilitar o entendimento dos resultados, a Figura 9 apresenta as médias móveis obtidas a partir dos resultados do consumo de memória e processamento com e sem a uti-

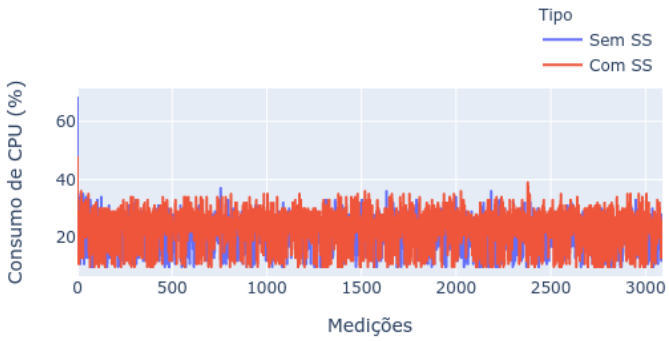

Figura 8: Consumo de processamento com e sem a utilização do Security Service.

lização do SS. Por fim, também foram gerados gráficos de *boxplot* para uma melhor visualização dos *Outliers*, os quais são apresentados nas Figuras 10 e 11.

	Média	Mediana	Moda	Desvio Padrão
Memória sem SS	45.968655	45.889648	46.590820	0.516518
Memória com SS	64.286068	64.094727	63.945312	0.468532
CPU sem SS	22.287411	23.000000	23.000000	4.721909
CPU com SS	23.423754	24.000000	25.000000	5.350205

Figura 9: Média, mediana, moda e desvio padrão do consumo de recursos durante a avaliação.

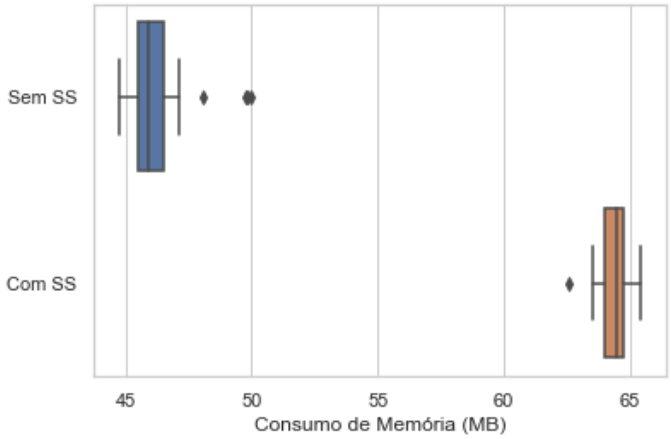

Figura 10: Boxplot do consumo de memória com e sem a utilização do SS.

6. DISCUSSÃO

Para estabelecer a comunicação segura em um cenário de soluções de *middleware* IoT, devemos levar em conta os dois tipos de comunicação: a coleta de dados provenientes de sensores e enviados ao M-Hub (ou seja, entre sensor/objeto inteligente e M-Hub), e a distribuição de dados entre instâncias do CDDL (ou seja, entre M-Hub e *brokers* CDDL).

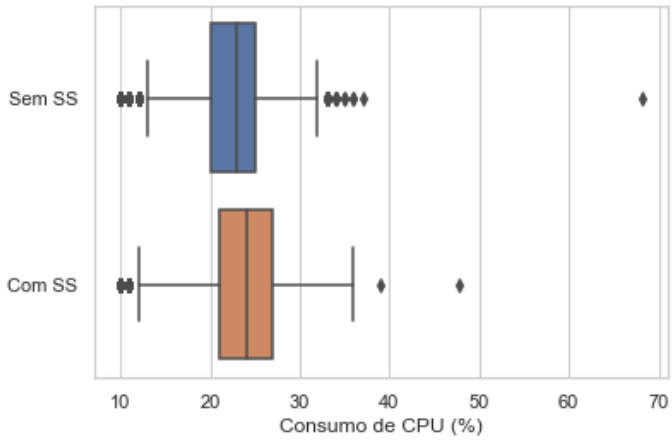

Figura 11: Boxplot do consumo de processamento com e sem a utilização do SS.

Considerando o caso específico do M-Hub/CDDL, a comunicação entre M-Hub e sensores pode ser assegurada utilizando os mecanismos que as tecnologias WPAN (*Wireless Personal Area Network*) oferecem (por exemplo, pareamento *Bluetooth*). Já a comunicação segura, internamente em uma instância do CDDL ou entre diferentes instâncias, pode ser garantida utilizando o SS, desde que ambas as partes possuam confiança na mesma autoridade certificadora. Este mesmo princípio se aplica para as instâncias do CDDL Desktop e em *Brokers*/clientes MQTT externos, conforme demonstrado na Seção 5. Dessa forma, pode-se assegurar a comunicação em diferentes contextos de aplicação do M-Hub/CDDL.

6.1 Análise dos Resultados de Consumo de Memória

Os resultados obtidos são compatíveis e confirmam a tendência dos resultados preliminares [8]: houve um aumento no consumo de memória ao se utilizar o SS. O custo médio sem a utilização da camada foi de 45,96 MB, já utilizando-a houve um aumento de 39.84 %, elevando o custo médio de memória para 64,28 MB. O desvio padrão obtido em ambos os casos é baixo, indicando que o custo adicional de memória é bem distribuído ao longo da execução do experimento. Acreditamos que alguns *outliers*, exibidos na Figura 10, ocorreram quando medições foram feitas no momento em que variáveis temporárias foram alocadas em memória para efetuar alguma operação do *middleware*.

6.2 Análise dos Resultados de Consumo de Processamento

Em relação ao processamento, os resultados preliminares obtidos em [8] não foram confirmados. Através do novo experimento, constatou-se que não houve diferença expressiva entre o consumo de processamento com e sem os recursos de segurança. Sem a utilização da camada de segurança, o valor médio foi de 22,28 % do poder de processamento do dispositivo. Ao se utilizar a camada de segurança, o valor médio aumenta para 23,42 %. A diferença equivale a um aumento de 5,09 % no custo do processamento. O aumento no custo de processamento pode ser melhor percebido no gráfico de *boxplot*, apresentado na Figura 11.

Ao se adicionar uma carga extra ao processamento e memória, devido à criptografia, já era esperado que houvesse aumento nos valores. Estes resultados são mais precisos que os preliminares, uma vez que apenas 25 medições, feitas no experimento anterior, não se mostraram adequadas para avaliar a solução, pois um menor número de medições torna o resultado mais susceptível a alterações devido a ocorrência de *outliers*. Apesar de apresentar alguns *outliers*, a maioria dos valores de processamento estão entre os quartis 1 e 3, compreendendo a faixa de valores entre 20 e 27 %. Utilizando a camada ou não, obtivemos valores de desvio padrão bem próximos no custo do processamento, indicando que o custo extra é distribuído uniformemente ao longo da execução.

6.3 Limitações da Solução Proposta

Apesar de demonstrada a eficiência e aplicabilidade da solução proposta, o *Security Service* possui algumas limitações. Atualmente o SS suporta apenas um certificado digital para o cliente e um para a autoridade certificadora por instância. Esta limitação impede que uma mesma aplicação possa estabelecer confiança em dois sistemas que utilizem certificados e autoridades certificadoras distintos. Com o suporte para múltiplos certificados, uma aplicação poderia escolher entre seus diferentes certificados ou confiar em mais de uma autoridade certificadora ao mesmo tempo, facilitando ainda mais a distribuição segura de dados no *middleware*.

Outa limitação é a não existência de um componente para automatizar a assinatura das requisições de certificado. A requisição de certificado do cliente, gerada pelo SS, precisa ser assinada por uma autoridade certificadora, e esta autoridade deve ser mantida por alguma entidade. Portanto, em uma aplicação IoT, o administrador deve manter o certificado da autoridade certificadora e assinar os certificados digitais dos clientes que possuem direito de utilizar a aplicação.

7. CONCLUSÃO E TRABALHOS FUTUROS

O objetivo deste trabalho foi facilitar a distribuição segura de dados de contexto em aplicações de IoMT. Aplicações de IoT/IoMT podem ser inviabilizadas caso não existam mecanismos para garantir a segurança dos dados. Com a proposta do SS, conseguimos atingir este objetivo, o serviço de segurança foi implementado, de maneira integrada ao *middleware* M-Hub/CDDL, para garantir os requisitos de segurança desejados.

A solução de segurança, apresentada como proposta de solução deste trabalho, oferece meios práticos para que o desenvolvedor possa utilizar o M-Hub/CDDL em aplicações de IoT/IoMT de forma segura, com mecanismos para o estabelecimento de comunicação segura, autenticação e controle de acesso dos dados. Além disso, as requisições de certificado digital podem ser geradas pelo SS, e exportadas para o dispositivo, facilitando o processo de obtenção do certificado assinado. A nova avaliação de desempenho da solução proposta confirmou que ela não gera uma sobrecarga excessiva ao dispositivo móvel. Apesar de adicionar algum custo computacional, os recursos de segurança são opcionais.

Como perspectivas futuras para esta pesquisa, podemos citar a implementação do suporte para múltiplos certificados digitais. Além disso, existe também a possibilidade do uso de *blockchain* como recurso de segurança aplicado a IoT/IoMT [26, 27, 28]. A *blockchain* é uma tecnologia complementar a IoT, sendo capaz de aprimorar aplicações de IoT em termos de interoperabilidade, privacidade, segurança, confiança e escalabilidade [29]. Especificamente, em relação o M-Hub/CDDL, a *blockchain* pode ser utilizada para prover um mecanismo de identidade para os dispositivos de IoT. Dessa forma, os dados publicados podem ser rastreáveis e a integridade deles pode ser facilmente verificada acessando a *blockchain*.

8. AGRADECIMENTOS

Este trabalho foi financiado em parte pela CAPES (Código de financiamento 001 e 88887.200532/2018-00); Instituto Nacional de Ciência e Tecnologia (INCT) da Internet do Futuro para Cidades Inteligentes (CNPq 465446/2014-0, CAPES 88887.136422/2017-00, e FAPESP 14/50937-1 e 15/24485-9); Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (311608/2017-5, 420907/2016-5, 312324/2015-4); e Fundação de Amparo à Pesquisa e ao Desenvolvimento Científico e Tecnológico do Maranhão - FAPEMA (BIC 04405/19).

9. REFERENCES

[1] "IoT World Today." [online] Available at: `https://www.iotworldtoday.com`. Accessed: 13-11-2020.

[2] E. Borgia, "The internet of things vision: Key features, applications and open issues," *Computer Communications*, vol. 54, pp. 1–31, 2014.

[3] G. Kortuem, F. Kawsar, V. Sundramoorthy, and D. Fitton, "Smart objects as building blocks for the internet of things," *IEEE Internet Computing*, vol. 14, no. 1, pp. 44–51, 2009.

[4] L. E. Talavera, M. Endler, I. Vasconcelos, R. Vasconcelos, M. Cunha, and F. J. d. S. e Silva, "The mobile hub concept: Enabling applications for the internet of mobile things," in *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pp. 123–128, IEEE, 2015.

[5] M. Endler and F. S. e Silva, "Past, present and future of the contextnet iomt middleware," *Open Journal of Internet Of Things (OJIOT)*, vol. 4, no. 1, pp. 7–23, 2018.

[6] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "Iot middleware: A survey on issues and enabling technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, 2016.

[7] B. D. T. P. Gomes, L. C. M. Muniz, F. J. Da Silva e Silva, D. V. Dos Santos, R. F. Lopes, L. R. Coutinho, F. O. Carvalho, and M. Endler, "A middleware with comprehensive quality of context support for the internet of things applications," *Sensors*, vol. 17, no. 12, 2017.

[8] A. Cardoso, T. Mendes, F. S. Silva, and A. Teles, "Facilitando a distribuição segura de dados de contexto em aplicações de internet das coisas móveis," in *Anais da VIII Escola Regional de Computação do Ceará, Maranhão e Piauí*, pp. 252–259, SBC, 2020.

[9] A. Manzoor, H.-L. Truong, and S. Dustdar, "Quality of context: models and applications for context-aware

systems in pervasive environments," *The Knowledge Engineering Review*, vol. 29, no. 2, p. 154–170, 2014.

[10] L. David, R. Vasconcelos, L. Alves, R. André, and M. Endler, "A dds-based middleware for scalable tracking, communication and collaboration of mobile nodes," *Journal of Internet Services and Applications*, vol. 4, no. 1, p. 16, 2013.

[11] M. Silva, A. Teles, R. Lopes, F. Silva, D. Viana, L. Coutinho, N. Gupta, and M. Endler, "Neighborhood-aware mobile hub: An edge gateway with leader election mechanism for internet of mobile things," *Mobile Networks and Applications*, pp. 1–14, 2020.

[12] S. Perera, S. Sriskandarajah, M. Vivekanandalingam, P. Fremantle, and S. Weerawarana, "Solving the grand challenge using an opensource cep engine," in *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, pp. 288–293, 2014.

[13] B. Gomes, L. Muniz, F. J. d. S. e Silva, L. E. T. Ríos, and M. Endler, "A comprehensive cloud-based iot software infrastructure for ambient assisted living," in *2015 International Conference on Cloud Technologies and Applications (CloudTech)*, pp. 1–8, IEEE, 2015.

[14] M. A. Khan and K. Salah, "Iot security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[15] H. G. C. Ferreira, R. T. de Sousa, F. E. G. de Deus, and E. D. Canedo, "Proposal of a secure, deployable and transparent middleware for internet of things," in *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1–4, 2014.

[16] D. Conzon, T. Bolognesi, P. Brizzi, A. Lotito, R. Tomasi, and M. A. Spirito, "The virtus middleware: An xmpp based architecture for secure iot communications," in *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–6, IEEE, 2012.

[17] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 70–95, 2016.

[18] M. Calderbank, "The rsa cryptosystem: History, algorithm, primes," *Chicago: math. uchicago. edu*, 2007.

[19] R. Focardi, F. Palmarini, M. Squarcina, G. Steel, and M. Tempesta, "Mind your keys? a security evaluation of java keystores," in *NDSS*, 2018.

[20] "Java^{TM} Platform, Standard Edition 7 API Specification, (2017). Class SSLContext.." [online] Available at: `https://docs.oracle.com/javase/7/docs/api/javax/net/ssl/\SSLContext.html`. Accessed: 11-11-2020.

[21] T. Mendes, A. Cardoso, A. Silva, D. Carvalho, S. Francisco, A. Teles, M. Endler, M. R. Júnior, *et al.*, "Uma arquitetura distribuída iot para gestão de presença e encontros de pessoas em ambientes internos," in *Anais da VIII Escola Regional de Computação do Ceará, Maranhão e Piauí*, pp. 236–243, SBC, 2020.

[22] "Google and Open Handset Alliance n.d. Android API Guide. Android Profiler." [online] Available at: `https://developer.android.com/studio/profile/android-profiler?hl=pt-br`. Accessed: 11-11-2020.

[23] R. A. Light, "Mosquitto: server and client implementation of the mqtt protocol," *Journal of Open Source Software*, vol. 2, no. 13, p. 265, 2017.

[24] "Google and Open Handset Alliance n.d. Android API Guide. Dumpsys." [online] Available at: `https://developer.android.com/studio/command-line/dumpsys`. Accessed: 11-11-2020.

[25] "Google and Open Handset Alliance n.d. Android API Guide. ADB." [online] Available at: `https://developer.android.com/studio/command-line/adb`. Accessed: 11-11-2020.

[26] M. A. Khan and K. Salah, "Iot security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[27] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with iot. challenges and opportunities," *Future generation computer systems*, vol. 88, pp. 173–190, 2018.

[28] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[29] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8076–8094, 2019.