Facilitando a Distribuição Segura de Dados de Contexto em Aplicações de Internet das Coisas Móveis

André Luiz A. Cardoso[1], **Thiago Wallass N. Mendes**[1], **Francisco José S. Silva**[1]
Ariel S. Teles[1]

[1]Laboratório de Sistemas Distribuídos Inteligentes
Universidade Federal do Maranhão (UFMA)
São Luís – MA – Brasil

{andre.cardoso, thiago.wallass, fssilva, ariel}@lsdi.ufma.br

Abstract. *Internet of Mobile Things (IoMT) applications have the potential to solve many problems. Health and wellness, logistics and smart cities are some areas that can benefit from it. A frequent approach to IoMT solutions involves the use of middleware, which aims to serve as a basis for developing applications. In this paper, we propose the addition, for optional use, of a security layer to the M-Hub/CDDL middleware. The main goal of the proposed solution is to guarantee secure communication, authentication and access control. By performing a performance evaluation, the computational cost regarding processing and memory consumption in the mobile side of the solution was identified due to the use of the layer.*

Resumo. *Aplicações de Internet das Coisas Móveis (IoMT) possuem potencial para solucionar diversos problemas. Saúde e bem estar, logística e cidades inteligentes são alguns domínios que podem se beneficiar disso. Uma abordagem frequente para soluções de IoMT envolve o uso de middleware, o qual visa servir de base para o desenvolvimento de aplicações. Este trabalho propõe a adição, para uso opcional, de uma camada de segurança ao middleware M-Hub/CDDL. O objetivo da solução proposta é garantir a comunicação segura, autenticação e controle de acesso. Através de uma avaliação de desempenho, identificou-se o custo computacional de processamento e consumo de memória no lado móvel da solução devido ao uso da camada.*

1. Introdução

O paradigma de Internet das Coisas (IoT) reúne diversos conceitos e tecnologias, tais como computação ubíqua e pervasiva, e tecnologias de sensoriamento e comunicação. Dessa forma, dispositivos equipados com sensores e atuadores (e.g., *smartphones*) podem se interconectar de maneira que haja uma interação contínua entre o mundo real e o digital [Borgia 2014]. Sistemas IoT permitem que objetos sejam capazes de interagir entre si, atuem de acordo com as suas interpretações e troquem informações com pessoas. Estes objetos, sendo físicos ou digitais, constituem a base da IoT e são chamados de Objetos Inteligentes [Kortuem et al. 2009].

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 [Talavera et al. 2015]. A

IoMT já está inserida de maneira significativa em diversos domínios de aplicação, tais como cidades inteligentes, segurança publica, logística, saúde e bem estar [Endler and e Silva 2018]. A heterogeneidade dos dispositivos de IoT é um problema recorrente, e uma possível solução consiste em utilizar soluções de *middleware*.

Um *middleware* de IoT pode ser descrito como um software que atua como intermediário entre os dispositivos IoT e as aplicações [Ngu et al. 2016]. Através da abstração de diferentes tecnologias, um *middleware* pode oferecer uma interface mais simples para o programador desenvolver suas soluções. O M-Hub/CDDL [Gomes et al. 2017] é uma dessas soluções de *middleware*, o qual combina um *gateway* móvel com uma camada de distribuição de dados e visa facilitar o desenvolvimento de aplicações de IoMT [Gomes et al. 2017]. Como exemplo de aplicação, considere um paciente que necessita de acompanhamento constante dos sinais vitais, em que os dados são coletados por sensores de dispositivos vestíveis. O médico monitora os dados à distância verificando a ocorrência de alguma anormalidade. Esses dados de saúde são especialmente sensíveis, sendo necessário, portanto, garantir a segurança na comunicação e no acesso a eles.

Este estudo propõe a adição de uma camada de segurança ao *middleware* M-Hub/CDDL, oferecendo um canal seguro de comunicação, mecanismos de autenticação e controle de acesso. A camada de segurança faz o gerenciamento dos elementos necessários para o funcionamento dos mecanismos propostos. Dessa maneira, torna-se possível a distribuição de dados de forma segura entre instâncias do M-Hub/CDDL. Além disso, é possível realizar o ajuste da privacidade na distribuição de dados, combinando o controle de acesso, componente da camada de segurança proposta por esse artigo, com os filtros já existentes no *middleware*.

O restante do artigo está organizado da seguinte forma. A Seção 2 introduz o *middleware* M-Hub/CDDL, necessário para o entendimento da solução proposta. A Seção 3 aborda a solução proposta. A Seção 4 apresenta uma avaliação de desempenho que compara o custo computacional com e sem a utilização dos recursos de segurança. A Seção 5 discute os resultados e compara a solução com trabalhos relacionados. Por fim, a Seção 6 conclui o trabalho e apresenta as perspectivas de trabalhos futuros.

2. Fundamentação Teórica

2.1. O *middleware* M-Hub/CDDL

O M-Hub/CDDL é um *middleware* com recursos para a aquisição, processamento e distribuição de dados de contexto, oferecendo suporte para o desenvolvimento de aplicações de IoT/IoMT [Gomes et al. 2017]. O M-Hub (*Mobile Hub*) atua como *gateway* móvel para a aquisição dos dados, enquanto o CDDL (*Context Data Distribution Layer*) é o responsável pela distribuição dos dados com suporte de qualidade de contexto [Manzoor et al. 2014].

O *middleware* suporta diferentes plataformas. O M-Hub atua em dispositivos móveis que utilizam o sistema operacional *Android*, já o CDDL possui três tipos de clientes: móveis (*Android*), *desktop* e web. Dessa forma, *smartphones* podem transmitir dados para instâncias móveis do M-Hub/CDDL ou para instâncias que funcionam na nuvem.

2.2. Mobile Hub

O M-Hub atua como intermediário entre sensores, atuadores e a conexão com servidores na nuvem (i.e., via CDDL). Implementada no M-Hub, a interface `Short-range Sensing, Presence and Actuation` (S2PA), atua como protocolo para comunicação de curto alcance com objetos móveis [Talavera et al. 2015]. Atualmente a S2PA implementa comunicação via *Bluetooth* 4.0 (BLE) e *Bluetooth* clássico. Além disso, também fornece acesso aos dados provenientes dos sensores internos de *smartphones Android*. Vários dispositivos vestíveis, veículos, sensores e atuadores podem ter seus dados coletados e receber comandos de atuação através do M-Hub.

2.3. CDDL

A camada de distribuição de dados de contexto (i.e., CDDL) pode ser utilizada de maneira integrada ao M-Hub para transmissão e enriquecimento dos dados coletados. O CDDL utiliza o MQTT[1] como protocolo de comunicação de dados, seja em conexões locais ou remotas [Gomes et al. 2017]. A Figura 1 exibe a arquitetura atual do M-Hub/CDDL.

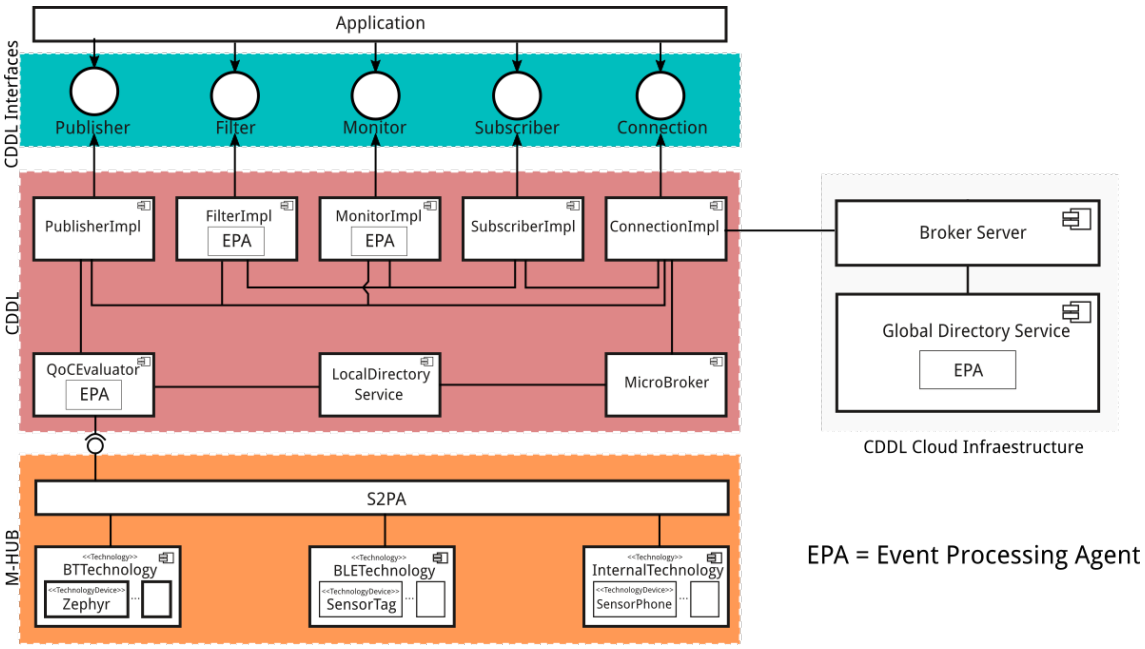


Figura 1. Arquitetura atual do M-Hub/CDDL [Gomes et al. 2017]

O M-Hub, através do 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, acurácia, e *timestamp*. Os componentes responsáveis pelo transporte dos dados são o `Microbroker` e o `ConectionImpl`.

O `Microbroker` encapsula uma versão modificada de um *broker* MQTT, já o componente `ConectionImpl` encapsula um cliente MQTT, que pode ser utilizado para publicar/subscrever em tópicos. Por fim, o CDDL oferece interfaces que são utilizadas por aplicações e, através delas, pode-se consumir os recursos do CDDL de maneira simples.

[1] http://mqtt.org/

2.4. Problemática Abordada

O protocolo de comunicação MQTT, utilizado pelo M-Hub/CDDL, já oferece suporte para mecanismos de segurança. A nível de transporte, tem-se o *Transport Layer Security* (TLS), que objetiva prover segurança de ponta a ponta contra ataques ativos do tipo *man in the middle* [Krawczyk et al. 2013]. Além de um canal de comunicação seguro, o TLS também pode impedir usuários não autenticados de estabelecerem conexão. O controle de acesso pode variar de acordo com o *broker*, mas uma forma comum é através de uma lista que armazena os tópicos que os usuários podem acessar. Atualmente, o M-Hub/CDDL não oferece meios para aproveitar-se de tais recursos.

3. Solução Proposta: um Serviço de Segurança

A solução proposta, chamada de *Security Service* (SS), acrescenta uma camada de segurança ao M-Hub/CDDL. Esta camada fornece meios para o estabelecimento de um canal seguro de comunicação, mecanismos de autenticação e controle de acesso. Para o funcionamento desses recursos, é necessário possuir os seguintes itens: chaves criptográficas, certificados digitais e lista de controle de acesso. Tais elementos podem ser gerados e armazenados utilizando o próprio SS. E para o desenvolvimento da camada de segurança, a seguinte política foi considerada:

- Uma instância do M-Hub/CDDL deve manter dois certificado digitais, um para o usuário e outro para a autoridade certificadora;
- A troca de mensagens deve utilizar criptografia;
- A autenticação 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 pelo usuário;
- Por fim, o uso do *middleware* com os recursos de segurança deve ser opcional, permitindo a não utilização do SS.

3.1. Gerência da Segurança

O SS oferece meios para: gerar chaves criptográficas, gerar requisição de assinatura de certificado (*Certificate Signing Request* - CSR), importar o certificado da autoridade certificadora confiável e importar o certificado do usuário do CDDL assinado pela mesma autoridade. Estes métodos são disponibilizados através de uma interface, e apresentados na Figura 2. Através deles, torna-se possível configurar o M-Hub/CDDL para que possa ser utilizado no modo seguro.

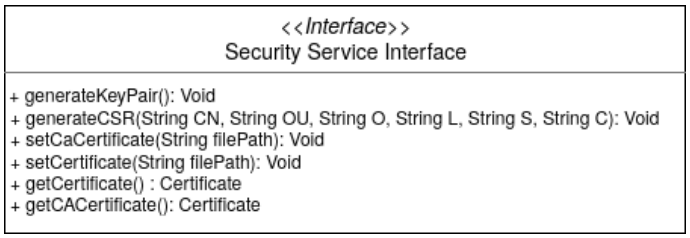


Figura 2. Métodos da interface relacionados ao gerenciamento de segurança.

Para armazenar as chaves e certificados de maneira segura, levou-se em conta o tipo de cliente CDDL. Para as instâncias móveis do M-Hub/CDDL, utilizou-se 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 [Focardi et al. 2018] com senha.

3.2. Canal Seguro de Comunicação e Autenticação

Uma vez que o MQTT possui suporte ao TLS, ele foi utilizado para garantir o canal seguro de comunicação com autenticação mútua (cliente e *broker*). Os componentes, pertencentes ao M-Hub/CDDL, cuja responsabilidade é estabelecer conexões, são respectivamente o `Microbroker` e o `Connection`. Esses componentes são acessados por meio da interface do CDDL. Para tornar o uso da segurança opcional, adicionou-se novos métodos à interface `Connection` e à classe `Microbroker` (Figura 3).

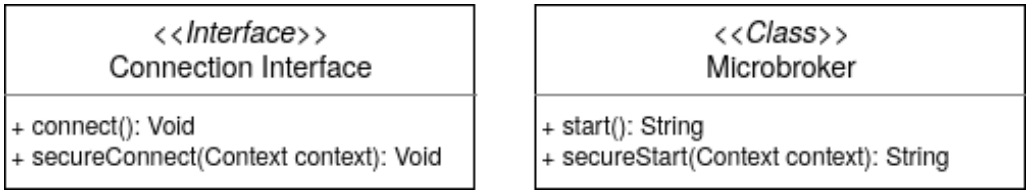


Figura 3. Componentes com os novos métodos de inicialização.

Com essas alterações, é possível iniciar o *broker* local e se conectar de forma segura. Ao serem chamados, os novos métodos (i.e., `secureConnect` e `secureStart`) automaticamente buscam as chaves e certificados anteriormente configurados e os utilizam na construção do contexto SSL (*Secure Sockets Layer*) [2] para a comunicação.

3.3. Controle de Acesso

O CDDL possui uma estrutura predefinida de tópicos baseada em serviços, o usuário apenas define quais serviços quer publicar/sobrescrever dados e o CDDL determina os tópicos automaticamente. Para gerenciar o acesso, é necessário criar regras que determinem quais serviços podem ser acessados. Através de sua interface, o SS oferece métodos para geração de regras. A Figura 4 exibe os métodos disponibilizados pelo SS para adicionar e remover as regras.

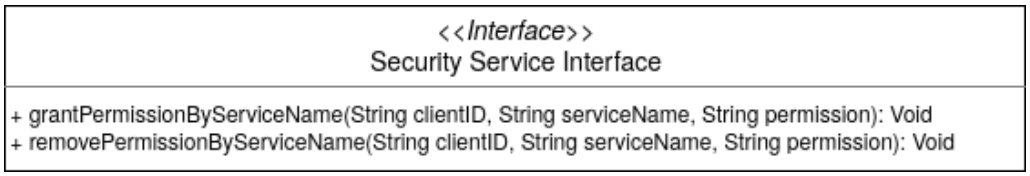


Figura 4. Modificação dos componentes `Connection` e `Microbroker`.

Utilizando os métodos acima, é possível conceder e revogar permissões para os clientes. Ao serem utilizados, regras são armazenadas pelo SS em um arquivo. Tais regras são geradas de acordo com o seguinte padrão: "ClientID tópico permissão". A Figura 5 exemplifica o resultado obtido ao se adicionar regras de controle de acesso.

O `Microbroker` oferece uma interface autorizadora que possui dois métodos: `canRead` e `canWrite`. Esses métodos são chamados automaticamente no *broker*

[2] https://docs.oracle.com/javase/7/docs/api/javax/net/ssl/SSLContext.html

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

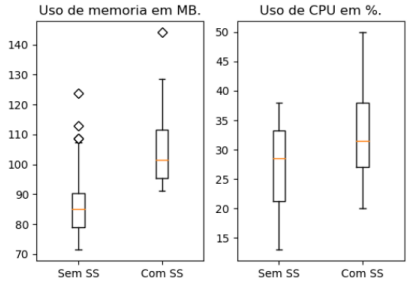
Figura 5. Exemplo de regras geradas utilizando o `Security Service`.

quando um cliente tenta publicar ou subscrever em tópicos. Assim, implementou-se esta interface autorizadora e seus métodos de forma que busquem e utilizem regras armazenadas no SS para permitir ou bloquear o acesso.

4. Avaliação de Desempenho

Para avaliar a solução proposta, foi utilizado uma aplicação para gestão de presença e encontros de pessoas em ambientes fechados. Esta aplicação utiliza um *smartphone* com o M-Hub/CDDL para detectar *beacons Bluetooth Low Energy* (BLE) e transmitir dados para um servidor executando uma instância *desktop* do CDDL. A aplicação móvel utiliza o `Microbroker` para coletar os dados do ambiente e uma conexão remota para enviar a notificação ao servidor. Esta aplicação trabalha de maneira intensa, sendo capaz de detectar aproximadamente 5 objetos por segundo. Por ser a 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.

Para esta avaliação, utilizou-se 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*[3] 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. A tela do dispositivo se encontrava apagada durante a avaliação. A Figura 6 exibe os resultados obtidos, o consumo de memória é medido em MB enquanto o processamento é medido em percentual da capacidade total de processamento do dispositivo.



	Média	Mediana	Moda	Desvio Padrão
Memória sem SS	88.415385	85.2	77.4	12.878536
Memória com SS	105.111538	101.6	99.5	12.959117
CPU sem SS	27.192308	28.5	20.0	7.076610
CPU com SS	32.846154	31.5	27.0	8.839128

Figura 6. Resultado: custo adicional em memória e processamento.

5. Discussão

Para estabelecer uma comunicação segura, deve-se levar em conta as diferentes plataformas que suportam o M-Hub/CDDL. A comunicação entre objetos inteligentes e M-Hub

[3]https://developer.android.com/studio/profile/android-profiler

pode ser protegida utilizando os mecanismos que as tecnologias WPAN oferecem (e.g., pareamento *Bluetooth*). A comunicação segura entre instâncias móveis do M-Hub/CDDL é garantida pelo SS, desde que ambas utilizem a mesma autoridade certificadora. Este princípio também se aplica às instâncias *desktop* ou a qualquer *broker* MQTT que ofereça suporte para o TLS. Dessa forma, pode-se assegurar a comunicação em diferentes contextos de aplicação do M-Hub/CDDL.

5.1. Análise dos Resultados da Avaliação de Desempenho

Os resultados obtidos demonstram que a camada de segurança é aplicável a um cenário real. A segurança foi implementada para ser um recurso opcional, mas seu uso adiciona algum custo computacional. Em relação a memória, o custo médio sem a utilização da camada foi de 85,2 MB, já utilizando-a houve um aumento de 19,24 %. Portanto, houve custo extra ao se utilizar a camada de segurança. *Outliers* ocorreram quando a medição foi feita no momento em que variáveis temporárias foram alocadas em memória para efetuar alguma operação do *middleware*. Em relação ao processamento, sem a utilização da camada de segurança, o valor médio foi de 28,5 % do poder de processamento do dispositivo. Ao se utilizar a camada de segurança, o valor médio aumenta para 31,5 %. A diferença equivale a um aumento de 10,52 % no custo do processamento. Verificou-se que houve aumento no consumo de ambos recursos ao se utilizar a camada de segurança. Utilizando a camada ou não obtivemos valores de desvio padrão bem próximos, indicando que o custo extra é distribuído uniformemente ao longo da execução. Ao se adicionar uma carga extra ao processamento e memória, devido à criptografia, já era esperado que houvesse aumento nos valores.

5.2. Comparação com Trabalhos Relacionados

Na literatura, existem diversos recursos de segurança para soluções de *middleware* de IoT [Ngu et al. 2016, Khan and Salah 2018]. Ferreira et al. [Ferreira et al. 2014] apresentaram uma arquitetura de segurança para um *middleware* transparente. Através dele, pode-se transformar objetos comuns em inteligentes, utilizando mecanismos que garantem privacidade, autenticação, integridade e confidencialidade na troca de dados. Conzon et al. [Conzon et al. 2012] propuseram o VIRTUS, um *middleware* IoT que utiliza o protocolo XMPP para fornecer comunicações seguras. Além de aproveitar os serviços padrões de segurança do protocolo, o VIRTUS faz uso dos protocolos TLS e SASL para autenticação e criptografia.

Apesar das diferentes tecnologias e objetivos de cada *middleware*, alguns exibem esforços para proteger a transmissão de dados. A solução apresentada nesse artigo diferencia-se por atuar em um *gateway* móvel, onde os recursos são restritos, e acrescenta um custo baixo de memória e processamento que é distribuído uniformemente ao longo da execução. 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 estabelece conexão segura com *brokers* MQTT que suportem o TLS, aumentando sua aplicabilidade.

6. Conclusão e Trabalhos Futuros

A camada de segurança, apresentada como proposta de solução deste estudo, oferece meios práticos para que o desenvolvedor possa utilizar o M-Hub/CDDL de forma a garantir um canal de comunicação seguro, além de autenticação e controle de acesso das

entidades. A avaliação de desempenho da solução proposta mostrou que ela não gera uma sobrecarga excessiva ao dispositivo móvel. Apesar disso, ainda existem melhorias a serem implementadas. Como perspectivas futuras para este trabalho, podemos citar o suporte para múltiplos certificados digitais. Além disso, existe também a possibilidade do uso de *blockchain* como recurso de segurança, uma vez que essa tecnologia pode ser utilizada para autenticar, autorizar e auditar os dados gerados.

Referências

Borgia, E. (2014). The internet of things vision: Key features, applications and open issues. *Computer Communications*, 54:1–31.

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

Endler, M. and e Silva, F. S. (2018). Past, present and future of the contextnet iomt middleware. *Open Journal of Internet Of Things (OJIOT)*, 4(1):7–23.

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

Focardi, R., Palmarini, F., Squarcina, M., Steel, G., and Tempesta, M. (2018). Mind your keys? a security evaluation of java keystores. In *NDSS*.

Gomes, B. D. T. P., Muniz, L. C. M., Da Silva e Silva, F. J., Dos Santos, D. V., Lopes, R. F., Coutinho, L. R., Carvalho, F. O., and Endler, M. (2017). A middleware with comprehensive quality of context support for the internet of things applications. *Sensors*, 17(12).

Khan, M. A. and Salah, K. (2018). Iot security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems*, 82:395–411.

Kortuem, G., Kawsar, F., Sundramoorthy, V., and Fitton, D. (2009). Smart objects as building blocks for the internet of things. *IEEE Internet Computing*, 14(1):44–51.

Krawczyk, H., Paterson, K. G., and Wee, H. (2013). On the security of the tls protocol: A systematic analysis. In *Annual Cryptology Conference*, pages 429–448. Springer.

Manzoor, A., Truong, H.-L., and Dustdar, S. (2014). Quality of context: models and applications for context-aware systems in pervasive environments. *The Knowledge Engineering Review*, 29(2):154–170.

Ngu, A. H., Gutierrez, M., Metsis, V., Nepal, S., and Sheng, Q. Z. (2016). Iot middleware: A survey on issues and enabling technologies. *IEEE Internet of Things Journal*, 4(1).

Talavera, L. E., Endler, M., Vasconcelos, I., Vasconcelos, R., Cunha, M., and e Silva, F. J. d. S. (2015). 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)*, pages 123–128. IEEE.