LegitimateBroker: Mitigando Ataques de Personificação em Broker MQTT na Internet das Coisas

Charles Rampelotto Junior[1], **Silvio E. Quincozes**[2] **e Juliano F. Kazienko**[1]

[1]CTISM – Universidade Federal de Santa Maria (UFSM)
Caixa Postal 5.082 – 97.105-900 – Santa Maria – RS – Brasil

[2]IC – Universidade Federal Fluminense (UFF)
Caixa Postal 100.092 – 24.220-971 – Niterói – RJ – Brasil

{charles,kazienko}@redes.ufsm.br, sequincozes@id.uff.br

Abstract. *The Internet of Things (IoT) has gained a prominent role in the emerging technologies scenario. One of the major security problems consist in the impersonation of devices. This work proposes a mechanism to mitigate impersonation attacks on MQTT Broker device in the Internet of Things, called LegitimateBroker. The proposal is based on mutual authentication between Publishers and Broker, indirect keys storage in the Broker and periodic renewal of keys in the Broker and in the Publisher. Experiments indicate that the proposed mechanism presents a low overhead compared to other approaches and can be easily parameterized so that the key renewal rate be given by the periodicity of publications carried out by the network nodes and the attack execution time.*

Resumo. *A Internet das Coisas (IoT) tem ganhado elevado destaque no âmbito das tecnologias emergentes. Nesse cenário, um problema de segurança relevante que carece de estudos consiste na personificação de dispositivos. Este trabalho propõe um mecanismo, chamado LegitimateBroker, a fim de mitigar ataques de personificação em um dispositivo Broker MQTT na IoT. A proposta fundamenta-se na autenticação mútua entre Publicadores e Broker, armazenamento indireto de chaves no Broker e renovação periódica de chaves no Broker e no Publicador. Experimentos indicam que o mecanismo proposto tem baixa sobrecarga comparado a outras abordagens e pode ser facilmente parametrizado a fim de que a taxa de renovação de chaves se adéque a periodicidade de publicações realizadas pelos nós da rede e ao tempo de efetivação do ataque.*

1. Introdução

A Internet das Coisas, do inglês, *Internet of Things* (IoT), é um dos principais paradigmas de comunicação da atualidade. Ele preceitua a comunicação de dispositivos do nosso cotidiano dotados de recursos computacionais e de comunicação, referidos como Objetos Inteligentes. O *Message Queuing Telemetry Transport* (MQTT) [OASIS 2019] é um dos protocolos proeminentes da camada de aplicação. Esse protocolo opera dentro do modelo Publicador (*Publisher*) - Assinante (*Subscribe*), o qual se argumenta ser apropriado para a Internet das Coisas devido à sua baixa sobrecarga e elevada eficiência energética [Waher 2015] [Gartner 2017] [Santos et al. 2016] [Eugster et al. 2003].

Com o crescimento acelerado da IoT, nos últimos tempos, os desafios de prover soluções de segurança para dispositivos com poucos recursos computacionais

intensificaram-se. O Modelo Publicador - Assinante tem como dispositivo central um *Broker*, o qual requer medidas adequadas a fim de preservar as propriedades de segurança da informação, especialmente, a autenticidade, confidencialidade e disponibilidade — uma vez que o mesmo representa um ponto único de falhas. O comprometimento de dados sensíveis, como chaves criptográficas, pode viabilizar ataques como o de personificação. Esse ataque pode ser facilitado através da prévia indisponibilidade dos serviços prestados pelo *Broker* legítimo, por meio de um ataque de negação de serviço, do inglês, *Denial of Service* (*DoS*), seguido de sua substituição, por exemplo [Wells et al. 2014] [Firdous et al. 2017] [Borgia 2014] [Waher 2015] [Jerkins 2017].

Embora as especificações do protocolo MQTT mencionem alternativas de mecanismos de segurança, tais mecanismos não são suficientes para eliminar as ameaças na comunicação entre o *Broker* e os publicadores, ou assinantes [OASIS 2019]. A criptografia assimétrica é um exemplo de método de segurança comumente adotado em tais cenários, no entanto, o custo computacional gerado por tal método pode ser inviável em dispositivos de recursos limitados. Por outro lado, mecanismos de autenticação baseados em credenciais (*e.g.*, nome de usuário e senha ou *Personal Identifier Number* - PIN) não contemplam confidencialidade, podendo facilmente serem violados. Portanto, devido à natureza limitada dos dispositivos da IoT que adotam o MQTT, um mecanismo robusto e leve é necessário para a autenticação e mitigação da personificação desses dispositivos.

O objetivo deste trabalho consiste em propor um mecanismo leve, chamado *LegitimateBroker*, a fim de mitigar a personificação de *Brokers* MQTT e de dispositivos em cenários da IoT. Tal mecanismo é calcado na autenticação mútua entre o *Broker* e os dispositivos publicadores, no armazenamento indireto de chaves pelo *Broker* e na renovação periódica de chaves em ambas as partes. Adicionalmente, é definido um esquema a fim de evitar a reprodução de mensagens. O mecanismo proposto pode ser facilmente parametrizado de modo a definir a frequência de renovação de chaves de acordo com a periodicidade de publicações realizadas pelos nós da rede, permitindo calibrar a redução do tempo de ação de um potencial atacante e o consumo de energético para a renovação de chaves. Para fins de avaliação, são realizados experimentos práticos onde as métricas de atraso e consumo energético do mecanismo proposto são comparadas com outras abordagens. Para tanto, a especificação do MQTT é levada em consideração [OASIS 2019].

O restante deste trabalho está organizado como segue. Na Seção 2, são levantados alguns trabalhos sobre o tema, apontando-se suas diferenças e limitações. Na Seção 3, os detalhes da proposta são apresentados. Na Seção 4, é apresentada a avaliação, onde o cenário de experimentação é descrito e os resultados obtidos são discutidos. Por fim, na Seção 5, são apresentadas as conclusões e as possibilidades de trabalhos futuros.

2. Trabalhos Relacionados

Em [Biju and Shekokar 2017] é apresentada uma abordagem de segurança para o protocolo MQTT em um cenário da IoT. A proposta conta com um esquema de autenticação baseado em criptossistema assimétrico nos dispositivos publicadores móveis, os quais são delegados a um servidor externo localizado na internet. Os autores argumentam que a personificação pode ser evitada quando os dispositivos móveis locomovem-se de um domínio local para outro. No entanto, as informações dos dispositivos armazenadas no servidor não são renovadas durante a operação do sistema, salvo quando estes efetuam a

troca de domínios. Neste momento, uma nova credencial, denominada de *ticket*, é gerada. Entretanto, a periodicidade em que os dispositivos mudam de domínio e, consequentemente, possuem suas chaves renovadas não é estabelecida. Portanto, o problema não é devidamente tratado, pois um atacante com tempo hábil para obter as informações secretas de algum destes dispositivos tem a possibilidade de executar a sua personificação.

Esfahani et. al [Esfahani et al. 2019], apresentam um mecanismo de autenticação mútua entre um dispositivo sensor e um roteador por meio de criptografia simétrica. O mecanismo explora o uso de *nonces*, *hashes*, operações Ou-Exclusivo (*XOR*) e chaves simétricas pré-compartilhadas. Todavia, o esquema depende de uma terceira parte confiável que consiste em um servidor de autenticação. Assim, o dispositivo sensor deve primeiramente registrar-se nesse servidor para, posteriormente, autenticar-se mutuamente com o roteador. Além do atraso decorrente do maior número de mensagens trocadas, um ponto único de falhas é introduzido — o qual pode impedir o funcionamento do sistema como um todo em caso indisponibilidade em razão de falhas ou ataques. Ademais, o dispositivo roteador depende de um módulo de plataforma confiável o qual exige uso de hardware específico. Portanto, além dessa abordagem aumentar os custos dos dispositivos e, consequentemente, o custo da implantação do sistema, tais dispositivos não são interoperáveis com os demais. Por fim, a proposta não conta com renovação de chaves, possibilitando ataques de personificação em caso de exposição da chave pré-compartilhada.

Em [Bisne and Parmar 2017], é proposto um esquema de autenticação com criptografia assimétrica. Os dispositivos publicadores e assinante são autenticados através de um terceiro confiável chamado Autoridade de Chaves. Como a segurança do *Broker* não é tratada e há a inclusão de mais uma entidade ao cenário, surgem novas vulnerabilidades, como intrusões que podem comprometer, ou até mesmo indisponibilizar, a operação de todo sistema. Levando-se em conta que a autenticação não é suficiente para mitigar ataques de personificação, pois um atacante de posse da chave pode autenticar-se e passar por um dispositivo legítimo, a possibilidade de tal ataque também está presente.

Em [Bhawiyuga et al. 2017] é proposta a autenticação baseada em *tokens*. O dispositivo publicador informa suas credenciais para um servidor de *tokens* utilizando o protocolo MQTT. Tal servidor deve retornar um *token* que é validado posteriormente pelo *Broker*. Mecanismos anti-reprodução e anti-personificação não são abordados. Ademais, a inclusão de um terceiro confiável implica nas vulnerabilidades discutidas anteriormente. Portanto, existe a possibilidade de personificação dos dispositivos nesse cenário, além de uma eventual indisponibilidade do servidor de *tokens* comprometer todo o sistema.

Em [Khemissa and Tandjaoui 2016], é proposto um esquema de autenticação para redes de sensores sem fio. A abordagem proposta combina o uso de criptografia simétrica, em dispositivos com recursos limitados, e criptografia assimétrica, em dispositivos dotados dos recursos computacionais. Operações criptográficas, *nonces* e códigos de autenticação de mensagens baseados em *hash*, do inglês, *Hash Message Autentication Code* (HMAC), são utilizadas para fins de autenticação mútua e estabelecimento de chaves simétricas para autenticação de mensagens. Apesar de haver uma técnica de mascaramento da identidade dos sensores, um atacante de posse desta informação é capaz de personificar o dispositivo. A não renovação de chaves também contribui para a efetivação deste tipo de ataque. Ataques de reprodução não são evitados, pois as mensagens não envolvem o uso de informações atualizadas, como *nonces* ou contadores.

3. O Mecanismo *LegitimateBroker*

O mecanismo *LegitimateBroker* é fundamentado na autenticação mútua das partes Publicador e *Broker*, no armazenamento indireto de chaves criptográficas pelo *Broker* e na renovação periódica dessas chaves. Para tal, o mecanismo adota operações criptográficas leves, de modo a suportar dispositivos com recursos computacionais limitados. Para a autenticação, o mecanismo adota uma abordagem baseada em desafio-resposta [Tanenbaum and Wetherall 2010], onde são computados códigos de autenticação de mensagens para autenticação mútua de dispositivos, bem como a autenticação de mensagens. Para o resgate e composição da Chave de Sessão, o mecanismo emprega operações *XOR*, as quais são computadas tanto pelo *Broker* quanto pelo dispositivo publicador.

O mecanismo proposto assume o pré-compartilhamento de um Elemento de Composição de Chave de 128 bits, entre o dispositivo Publicador e o *Broker*, o qual é utilizado para a composição da Chave de Sessão. A pré-instalação da Chave de Sessão ocorre somente no dispositivo Publicador. O Elemento de Composição de Chave e a Chave de Sessão são renovados periodicamente de acordo com um parâmetro ajustável, que permite a calibragem entre a redução do consumo de energia para esse cômputo e o tempo de ação de um atacante em caso de exposição da chave. A Chave de Sessão é computada e renovada pelo dispositivo *Broker*.

O protocolo MQTT conta com 3 níveis de QoS, do inglês, *Quality of Service*, sendo o de nível o de nível zero "*No Máximo Um*", em que há somente uma tentativa de envio da mensagem de publicação, sem que haja a confirmação de entrega da mesma. O nível de QoS 1 é definido como "*Ao Menos Um*", ou seja, há ao menos uma confirmação de entrega da mensagem de publicação. Já o nível de QoS 2 é definido como "*Exatamente Um*", onde há uma confirmação de entrega da mensagem de publicação e uma nova confirmação de não duplicação da mesma [OASIS 2019].

O mecanismo *LegitimateBroker* adota o QoS nível 1. Essa escolha se deve à garantia de que haverá a entrega de ao menos uma mensagem de publicação. Uma vez que o mecanismo conta com um esquema de Contadores de Mensagem, o efeito de QoS 2 é atingido sem necessidade de Mensagens adicionais, já que é possível evitar a duplicação das mesmas a partir de tal contador. O QoS 1 demonstra melhor desempenho em tempo de execução em relação ao QoS 2, conforme é demostrado na Seção seguinte. Embora que o QoS 0 apresente a menor sobrecarga, devido à não confirmação do recebimento de mensagens, o mecanismo proposto exige a entrega de todas mensagens do processo de autenticação. Portanto, o nível de QoS 1 mostra-se o mais adequado.

O mecanismo proposto é integrado ao processo de publicação, adaptando as mensagens envolvidas nas interações Publicador - *Broker* de acordo com as especificações do protocolo MQTT. Portanto, as mensagens abordadas consistem em: *CONNECT*, *CONNACK*, *PUBLISH*, *PUBACK* e *DISCONNECT*. Recentemente, foi publicada a especificação da versão 5.0 do protocolo MQTT [OASIS 2019], que além de especificar essas mensagens, presentes nas versões anteriores, introduz um novo tipo de mensagem denominado *AUTH*. Esse tipo de mensagem tem como propósito permitir a implementação de mecanismos de autenticação estendida, como autenticação do tipo desafio/resposta. A definição desta nova mensagem vem ao encontro do mecanismo *LegitimateBroker*. Em particular, as Mensagens M2 e M3 da Figura 1 consistem em mensagens *AUTH*. No entanto, cabe ressaltar que o suporte a esse tipo de mensagem ainda é limitado

nas bibliotecas MQTT atuais, portanto, a sua implementação no LegitimateBroker se dá, experimentalmente, por meio de em um *socket* TCP. Assim, embora a implementação proposta seja direcionada ao protocolo MQTT na versão 5, a mesma pode ser suportada por versões anteriores do protocolo, onde o tipo de mensagem *AUTH* não é especificado.

Na Figura 1, são ilustradas as mensagens trocadas entre um dispositivo Publicador (P_i) e o dispositivo *Broker* (Br) no momento da abertura de uma conexão seguida da publicação de uma dada informação. A relação completa das notações adotadas são descritas na Tabela 1. Com exceção das mensagens M2 e M3, as demais mensagens exibidas são típicas de uma publicação insegura com QoS 1 no MQTT. Portanto, de modo a evitar a transmissão de novas mensagens, o mecanismo *LegitimateBroker* introduz os parâmetros necessários para seu funcionamento nas próprias mensagens envolvidas no processo de conexão e publicação. Dessa forma, uma vez concluído o processo de autenticação, opcionalmente, as mensagens M5 e M6 podem ser transmitidas múltiplas vezes dentro da mesma conexão. A seguir o processo completo é descrito.

Tabela 1. Notação.

Pi	Identificação do *i-ésimo* Publicador	Br	Identificação do *Broker*
R_{Pi}	*Nonce* do Publicador	R_{Br}	*Nonce* do *Broker*
H	*Hash* Criptográfico Unidirecional	K	Chave de Sessão
N	Elemento de Composição da Chave de Sessão	E	Encriptação
D	Decriptação	M	Mensagem
C_M	Contador de Mensagens	C_P	Contador de Publicações
$K^{'}$	Nova Chave de Sessão	$N^{'}$	Novo Elemento de Composição da Chave de Sessão
$\|\|$	Concatenação	$\bigoplus$	Ou-Exclusivo (XOR)

O mecanismo assume que a Chave de Sessão (K_i) e o Elemento de sua Composição (N_i) são previamente carregados na memória dos dispositivos, sendo K_i e N_i no Publicador e somente N_i no *Broker*. Assim, o *Broker* terá armazenado apenas os Elementos de Composição de Chave de Sessão compatíveis com cada Publicador.

Na Mensagem (M1), o Publicador inicia a conexão e, consequentemente, a Fase de Autenticação Mútua e Composição da Chave de Sessão. Para tal, as informações que deverão ser utilizadas pelo *Broker* para efetuar as operações necessárias de composição da Chave de Sessão, bem como um desafio de autenticação, são enviados juntamente com a mensagem *CONNECT* do protocolo MQTT. Essas informações consistem em: Identificação do Publicador (P_i), um número aleatório (*nonce*) gerado antes do envio da mensagem (R_{Pi}) e o resultado da operação XOR entre o Elemento de Composição da Chave de Sessão com a Chave de Sessão presente no Publicador ($N_i \bigoplus K_i$). Dada a sensibilidade das informações envolvidas no cômputo desse XOR, é importante que haja confidencialidade na comunicação. Portanto, essa informação é encriptada por meio do algoritmo *Advanced Encryption Standard* (AES) 128 bits, utilizando como chave criptográfica o Elemento de Composição da Chave de Sessão ($E_{Ni}(N_i \bigoplus K_i)$). O *Broker*, por sua vez, deverá realizar a decriptação do conteúdo da mensagem. Uma vez de posse das informações encriptadas, o mesmo condições de compor a Chave de Sessão, efetuando uma operação XOR em seu Elemento de Composição correspondente, presente na memória do Broker ($N_i \bigoplus K_i \bigoplus N_i = K_i$). Assim, ambas partes conhecerão uma Chave de Sessão para uso no restante da conexão.

Na Mensagem 2 (M2), o *Broker* utiliza as informações recebidas do Publicador, juntamente de sua identificação e um *nonce*, para o cômputo de um valor MAC resultante

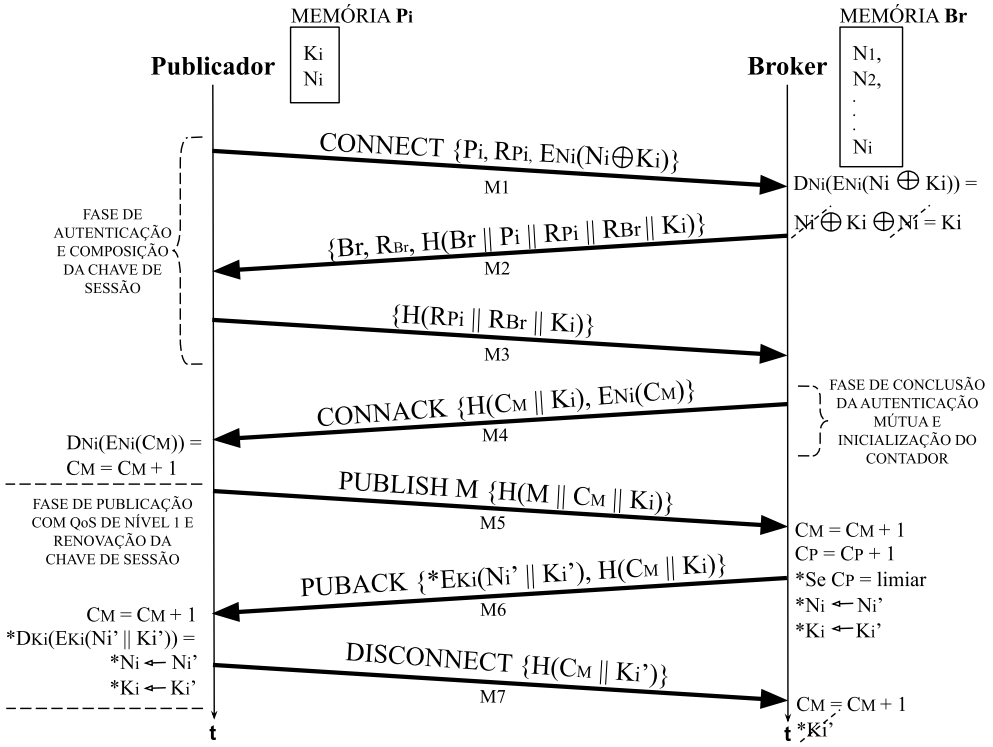


Figura 1. Mecanismo *LegitimateBroker*.

de um *Hash* criptográfico dessas informações concatenadas à Chave de Sessão. Feito isso, o *Broker* está habilitado para responder o desafio e ao mesmo tempo desafiar a outra parte, a fim de comprovar sua autenticidade.

Na Mensagem 3 (M3), se a autenticidade do Broker se confirmar, o Publicador computa um novo MAC através do *Hash* dos *nonces* de ambas as partes, concatenados à Chave de Sessão. Assim, o Publicador é capaz de responder ao desafio e comprovar sua autenticidade perante o Broker. Com efeito, o Broker deverá verificar o MAC recebido, concluindo a autenticação mútua.

Na Mensagem 4 (M4), dá-se início à Fase de Inicialiação do Contador de Mensagens. Para tal, o Broker confirma o fim da Fase de Autenticação, enviando a mensagem *CONNACK* do protocolo MQTT juntamente com o Contador codificado com a Chave de Sessão e um MAC gerado de um Hash Criptográfico sobre o Contador e a Chave de Sessão. O valor inicial do contador partirá da geração de um número aleatório. Desta forma, ao receber a mensagem 4, o Publicador deve decodifiar o valor do contador, incrementá-lo e armazená-lo para verificação das próximas mensagens recebidas.

A partir da Mensagem 5 (M5), inicia-se a Fase de Publicação com Nível de QoS 1 e Renovação da Chave de Sessão. Conforme abordado anteriormente, a escolha do nível de QoS 1 se deve à garantia de entrega das mensagens, e também pela não necessidade de nível de QoS 2, já que a duplicação das mensagens é evitada pelo esquema de Contador de Mensagens. A Mensagem 5 é responsável pelo envio da carga útil, ou seja, o dado coletado pelo Publicador, representado por *M* na sequência da mensagem *PUBLISH* do protocolo MQTT. Um MAC gerado através da computação do Hash $H(M\|C_M\|K_A)$

também é enviado para autenticar a mensagem. Ao receber a mensagem *PUBLISH*, o Broker deve incrementar o Contador de Mensagens, verificá-lo para constatar que não se trata de uma mensagem reproduzida, e também verificar a autenticidade da mensagem através do MAC recebido. Se tudo ocorrer bem, o dado *M* é armazenado e um Contador de Publicações (C_P) é incrementado. Se C_P atingir seu limiar pré-definido, ocorre a Renovação da Chave de Sessão ($K^{'}$) e do Elemento de Composição ($N^{'}$).

Na Mensagem 6 (M6), está presente a mensagem *PUBACK* do protocolo MQTT. Tal mensagem trata de confirmar o recebimento da mensagem de publicação anterior, enviada pelo Broker. A mesma somente apresenta-se nos níveil de QoS 1. Caso o limiar de publicações seja atingido, o Broker deve enviar juntamente com a mensagem *PUBACK*, a nova Chave de Sessão e seu Elemento de Composição, codificados através de um algoritmo simétrico, utilizando como chave criptográfia a Chave de Sessão vigente ($E_{Ki}(N_i'\|K_i')$). Ao final, um MAC também é gerado e adicionado à mensagem ($H(C_P\|K_i)$), utilizando o valor do Contador de Mensagens atualizado e a Chave de Sessão vigente. O Publicador, ao receber a M6, incrementa seu Contador de Mensagens e verifica a legitimidade da mesma. Também verifica o MAC recebido, a fim de confirmar sua autenticidade. Se confirmado, o Publicador utiliza o mesmo algoritmo simétrico para decodificar a Nova Chave de Sessão e o Novo Elemento de sua Composição. Assim, os valores antigos são substituídos pelos novos e a Renovação de Chaves é concluída.

Na Mensagem 7 (M7), o Publicador, de posse da Nova Chave de Sessão, finaliza a conexão enviando a mensagem *DISCONNECT* do protoco MQTT, juntamente com um MAC gerado a partir do Contador de Mensagens atualizado e a Nova Chave de Sessão ($H(C_P\|K_i')$). O Broker, ao receber a mensagem *DISCONNECT*, incrementa o Contador de Mensagens e verifica a legitimidade da mesma. Também trata de verificar o MAC recebido, o qual teve a nova Chave de Sessão utilizada para sua geração. Por fim, se for uma mensagem autêntica, o Broker apaga a Chave de Sessão gerada e mantém armazenado apenas o Novo Elemento de Composição da Chave de Sessão.

4. Avaliação

Nesta Seção, o mecanismo *LegitimateBroker* é avaliado. Na Subseção 4.1, são descritos os cenários construídos a fim de avaliar a proposta. Já na Subseção 4.2, são apresentados os resultados obtidos junto a uma discussão sobre os mesmos.

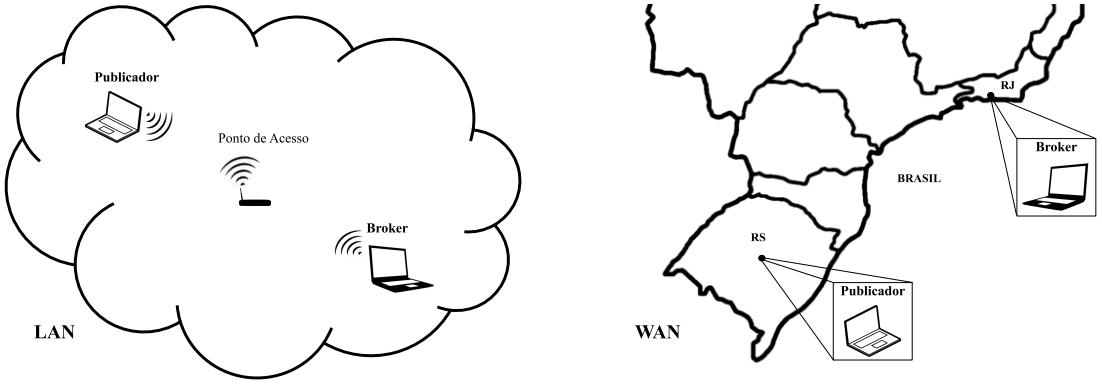
4.1. Descrição dos Cenários de Experimentação

Inicialmente, cabe destacar que foram implementados dois cenários de experimentação. O primeiro cenário foi construído a partir de uma rede *Local Area network* (LAN). No segundo cenário, o publicador e o *Broker* comunicam-se através de uma *Wide Area Network* (WAN). A Figura 2 ilustra tais cenários. Para fins de experimentação com o mecanismo proposto, foram adotados os algoritmos AES de 128 bits, para criptografia simétrica, e *Message Digest 5* (MD5), para geração de *hash* criptográfico e de código de autenticação de mensagem. Tais mecanismos foram escolhidos por serem relativamente leves, porém, o mecanismo não é dependente desses algoritmos, podendo-se optar por outros de acordo com os requisitos de segurança e capacidade computacionais dos dispositivos envolvidos.

No cenário LAN, ilustrado na Figura 2 (a), o *Broker MQTT* foi implementado

na linguagem *Python*, usando a biblioteca *Paho*[1] — a qual possui suporte à versão 5 do protocolo MQTT. Essa implementação foi instalada em um Notebook com processador Intel Dual Core, com 6 GB de memória RAM e sistema operacional Linux (Ubuntu 16.4). Nesse cenário, o dispositivo publicador foi implementado na linguagem Java, através da biblioteca *HiveMQ*[2], a qual oferece suporte à mesma versão do MQTT adotada no *Broker*. Essa implementação foi instalada em um Netbook com processador Intel Atom, com 2 GB de memória RAM e sistema operacional Linux (Ubuntu 16.4). A rede LAN foi construída através de um ponto de acesso sem fio *D-Link* 802.11g/2.4 GHz.

Diferentemente do cenário anterior, que reflete uma aplicação de menor escala, o cenário WAN é adaptado com a finalidade de avaliar o mecanismo *LegitimateBroker* em aplicações onde o publicador encontra-se fisicamente distante do *Broker*. Para tanto, foram consideradas as mesmas implementações do cenário anterior, porém, com um *Broker* mais robusto e uma conexão de múltiplos saltos (*e.g.*, nove saltos), via Internet. Dessa forma, o mesmo publicador utilizado no cenário anterior comunica-se com um *Broker* situado à uma distância de aproximadamente 1.646 km, instalado em um Notebook com processador Intel i7, com 16 GB de memória RAM e sistema operacional Windows 10.



(a) Cenário Interno (LAN). (b) Cenário Externo (WAN).

Figura 2. Cenários de Experimentação.

Adicionalmente, a fim de avaliar o consumo energético e tempo de execução do mecanismo *LegitimateBroker* a partir de um dispositivo comumente utilizado em aplicações da IoT, os experimentos em ambos os cenários foram repetidos tendo publicador um *Smartphone Honor* 8X, com 4 GB de memória RAM e sistema operacional *Android* versão 9. Uma vez que o *Android* oferece suporte nativo à linguagem Java, a mesma implementação utilizada no publicador *Netbook* foi instalada nesse dispositivo. Cabe ressaltar, no entanto, que os experimentos envolvendo TLS utilizam as tecnologias de armazenamento de chaves típicas de cada plataforma. Enquanto, no *Netbook*, é utilizado uma tecnologia provida nativamente pela Oracle, chamada *Java Keystore* (JKS), no *Android* é utilizado o formato *BouncyCastle Keystore* (BKS) — o qual é provido por uma biblioteca de terceiros chamada *BouncyCastle*. Para os experimentos com TLS (versão 1.2) foi utilizado o conjunto de algoritmos (*ciphersuite*): RSA com chave de 4096 bits, *Elliptic-curve Diffie–Hellman* (ECDHE), SHA-256 e criptografia AES 128 bits.

[1]https://github.com/eclipse/paho.mqtt.testing

[2]https://www.hivemq.com/blog/mqtt-client-library-encyclopedia-eclipse-paho-java/

A partir desses cenários, foram coletadas amostras de tempo para a execução de 500 publicações, cada qual com um tamanho fixo de 32 *Bytes* de carga útil. É importante observar que cada publicação se dá através de uma nova conexão. Desse modo, foi considerada a média aritmética simples dos intervalos entre o envio da primeira mensagem até a última mensagem (*i.e.*, da mensagem *CONNECT* até *DISCONNECT*).

Nos experimentos envolvendo o *Smartphone*, o consumo energético foi obtido através da conversão do valor em *miliampère* por hora (*mAh*) — reportado no gerenciador de aplicações do *Android* — para *Joules* (J). Para tanto, primeiramente é necessária a multiplicação do valor da corrente, em amperès (A), pelo tempo, em segundos. Com isso, é obtida a Carga (C), em *coulombs*, conforme a Equação 1. Em seguida, esse valor é multiplicado pela Tensão (V) da bateria do *smartphone*, resultando no valor em *Joules* da energia acumulada consumida, conforme a Equação 2.

$$Carga\,(C) \;=\; Corrente\,(A) \;\cdot\; Tempo\,(s) \tag{1}$$

$$Energia\,(J) \;=\; Carga\,(C) \;\cdot\; Tensao\,(V) \tag{2}$$

Portanto, conforme denota a Equação 3, cada *mAh* equivale 3,6 *culombs*. Então, uma vez que a bateria do *smartphone* utilizado na experimentação possui 3,82 *volts* [Empetel 2019], o valor do consumo energético em *Joules* é dado pela a Equação 4. É importante observar que, para cada experimento, o valor medido representa o consumo em *mAh* acumulado ($consumo_{mAh}$) para a execução do experimento, ou seja, para as 500 publicações realizadas em cada sistema avaliado.

$$Carga\,(C) \;=\; 0,001\;A \;\cdot\; 3600\;s \;=\; 3,6\;C \tag{3}$$

$$Energia\,(J) \;=\; (consumo_{mAh} \;\cdot\; 3,6) \;\cdot\; 3,82 \tag{4}$$

Durante a execução dos experimentos, o dispositivo publicador estabelece 500 conexões em um intervalo de dois segundos entre cada conexão. Para cada conexão, ocorre a publicação de um conjunto de dados que totalizam 32 *Bytes* de carga útil. Após a publicação, a conexão é encerrada. A execução dos experimentos práticos tem por objetivo a comparação do mecanismo *LegitimateBroker* com as abordagens sem criptografia e *Transport Layer Security* (TLS). Para cada uma dessas abordagens, os três níveis de QoS do MQTT são experimentados.

Tradicionalmente, ataques de personificação são baseados no uso de informações de dispositivos legítimos, em que o atacante é capaz assumir a identidade da vítima. Desse modo, nos cenários estudados, para que um atacante personifique um *Broker* são necessárias duas etapas básicas: (i) acesso à informação confidencial (e.g., *por violação física ou força bruta*), e (ii) implantação de um *Broker* atacante. Dado que o mecanismo *Legitimate Broker* é resistente à exploração de chaves obsoletas, é importante então a definição de uma periodicidade de renovação de chaves de modo a mitigar a ação de um atacante por meio da expiração de chaves antigas, porém, sem sobrecarregar os dispositivos com limitações energéticas e computacionais através de renovações desnecessárias.

Por fim, um experimento envolvendo a introdução de um dispositivo publicador atacante em cada um dos cenários, de modo a calcular o tempo de ação do atacante para que o ataque seja executado, é executado. Com isso, a frequência de renovação de chaves no mecanismo *LegitimateBroker* pode ser calibrada. O ataque em questão consiste na abertura de múltiplas conexões, esgotando o número de *threads* do *Broker*, impedindo que conexões de publicadores legítimos sejam estabelecidas — causando um ataque de negação de serviço, similar ao ataque SYN-Flood [Degirmencioglu et al. 2016], porém, a nível de camada de aplicação. Com isso, especialmente, *Brokers* que limitam a quantidade de *threads* disponíveis podem, rapidamente, ficar indisponíveis. No entanto, a definição desse limite é uma recomendação dos fornecedores para evitar a redução de performance. O guia do desenvolvedor do *Broker HiveMQ*, por exemplo, estabelece esse tipo de recomendação [HiveMQ 2019], tendo como padrão o limite de 1024 *threads*. Portanto, o ataque experimentado é uma ameaça real aos *Brokers* atuais.

4.2. Resultados e Discussão

A etapa de inicialização do mecanismo *LegitimateBroker* conta com a geração de números pseudo-aleatórios e operações criptográficas leves. Em ambientes IoT reais, assume-se que essa etapa ocorre no ato de implantação do sistema. A partir da mensagem M1, o procedimento ocorre a cada vez em que uma nova conexão é estabelecida. Conforme ilustrado na Figura 3, é possível notar um maior atraso para a etapa de conexão e autenticação mútua, o que é aceitável devido a quantidade de informações transmitidas e cômputos realizados pelas partes nesta fase do protocolo. Após sua conclusão, as etapas seguintes podem ocorrer de forma segura. O detalhamento dos tempos gastos para cada etapa do mecanismo, no cenário LAN, são ilustrados na Figura 3. É importante notar que, alternativamente, múltiplas publicações podem ser ocorrer dentro de uma mesma conexão. Nesse caso, o tempo médio para cada publicação é de 9,511 segundos.

Conforme esperado, a sobrecarga temporal introduzida pelo mecanismo *LegitimateBroker* é consideravelmente inferior à abordagem envolvendo TLS. Mais especificamente, considerando o mesmo nível de QoS (QoS 1), a abordagem TLS apresenta um aumento de 424,3% em relação ao tempo médio exigido pelo *LegitimateBroker* no cenário

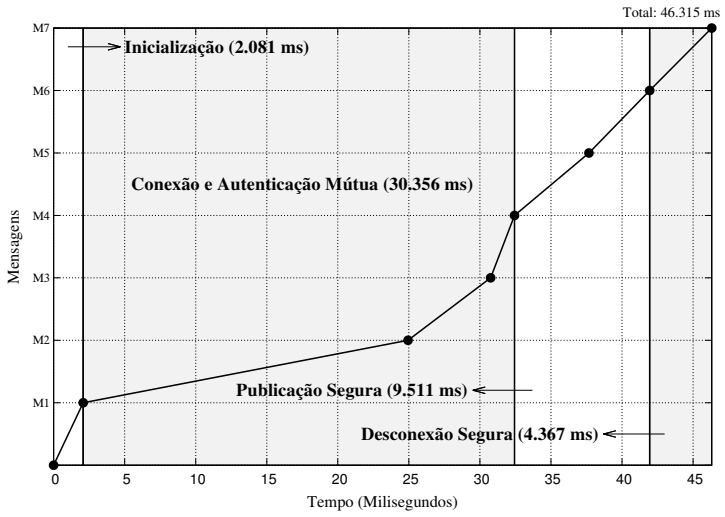
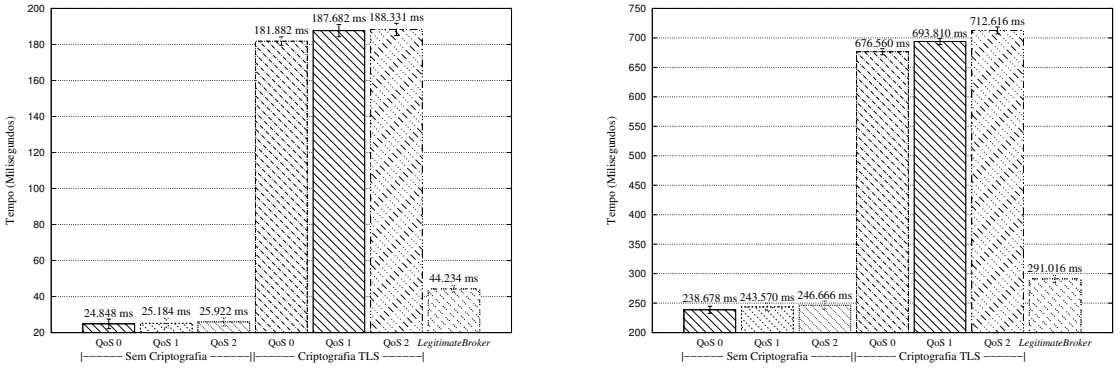


Figura 3. Tempo de execução do mecanismo *LegitimateBroker*.

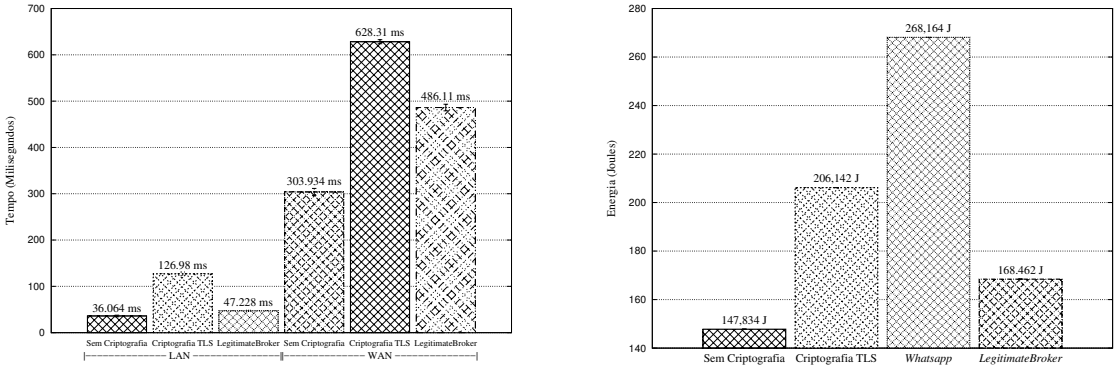
LAN e um aumento de 238,4% no cenário WAN — conforme ilustrado na Figura 4. Os resultados das medidas de tempo da abordagem envolvendo o uso de criptografia TLS, no publicador *Netbook*, demonstram uma alta sobrecarga em relação à abordagem sem criptografia. O uso de TLS demonstrou afetar significativamente o desempenho da aplicação, tanto no cenário interno, conforme a Figura 4 (a), quanto no cenário externo, conforme a Figura 4 (b). Portanto, a justificativa pela qual dispositivos com capacidade computacional limitada tendem evitar o uso dessa abordagem é elucidada.



(a) Cenário Interno (LAN). (b) Cenário Externo (WAN).

Figura 4. Tempo médio para publicação através do *Netbook*.

O mesmo comportamento é observado a partir do dispositivo *smartphone*. Nesse caso, os experimentos demonstram uma maior sobrecarga relacionada a abordagem envolvendo TLS, seja em termos de tempo para publicação, conforme ilustrado na Figura 5 (a), ou em consumo energético, conforme ilustrado na Figura 5 (b). Para o cômputo do consumo energético, foram consideradas as 500 conexões do cenário interno. Os resultados indicaram uma economia de 22,37% comparado ao uso de TLS no mesmo cenário e com o mesmo nível de QoS. Em adição, procurou-se comparar o sistema *LegitimateBroker* com um aplicativo largamente utilizado na atualidade, a saber: o *WhatsApp*, versão 2.19.156. Para a transmissão de 500 mensagens nesse aplicativo, observou-se o consumo de 268,164 *Joules*, que supera o *LegitimateBroker* em 62,82%, conforme a Figura 5 (b).



(a) Tempo Médio de Publicação. (b) Consumo Energético Acumulado.

Figura 5. Medições para publicação através do *Smartphone Android*.

Acredita-se que uma das razões para o comportamento de sobrecarga observado

no uso de TLS ocorra devido aos algoritmos envolvidos, que incluem criptografia assimétrica, os quais tipicamente apresentam custo computacional superior aos algoritmos de cifras simétricas [Wander et al. 2005]. Outro fator relevante é o incremento no número de mensagens trocadas, incluindo, mensagens para troca de cifras, acordo e compartilhamento de chaves simétricas, envio de certificados digitais, entre outras [Borgia 2014].

Na Figura 6 é ilustrado o tempo para a efetivação de um ataque DoS em um cenário interno e externo, o qual tem início no instante de 50 segundos. Baseado nesse gráfico, em um cenário externo, um *Broker* com limite de 1024 *threads* — que é padrão em *brokers* como o *HiveMQ* [HiveMQ 2019] — pode deixar de responder as requisições legítimas em apenas 205 segundos de duração de um ataque DoS. A situação é ainda mais crítica em um cenário interno, onde o *Broker* deixa de atender as requisições de publicadores legítimos no instante 65, em apenas 15 segundos após o início do ataque. Portanto, a partir dessas informações, é possível obter um Limiar para Renovação de Chaves (LRC[3]) adequado para cada cenário por meio da Equação 5. Essa equação leva em consideração o Tempo de Efetivação do Ataque DoS (TE_{DoS}) e o Intervalo entre Publicações (I_{pub}) em que o dispositivo publicador transmite as informações.

$$\lfloor LRC \rfloor = \frac{TE_{DoS}}{I_{pub}} - 1 \tag{5}$$

Suponha uma aplicação de rede de sensores onde a informação coletada por um sensor "dispositivo publicador" é transmitida em um intervalo de 2 segundos entre cada mensagem. Esse é um tempo factível para aplicações que necessitam de constante monitoramento (*e.g.*, sensoriamento de gases, como o hidrocarboneto [Dudar et al. 2018]). Desse modo, de forma a garantir a segurança da aplicação e impedir que a efetivação desse tipo de ataque facilite a personificação do *Broker* legítimo, o limiar para renovação de chaves é de 6 publicações no cenário interno e 101 publicações no cenário externo.

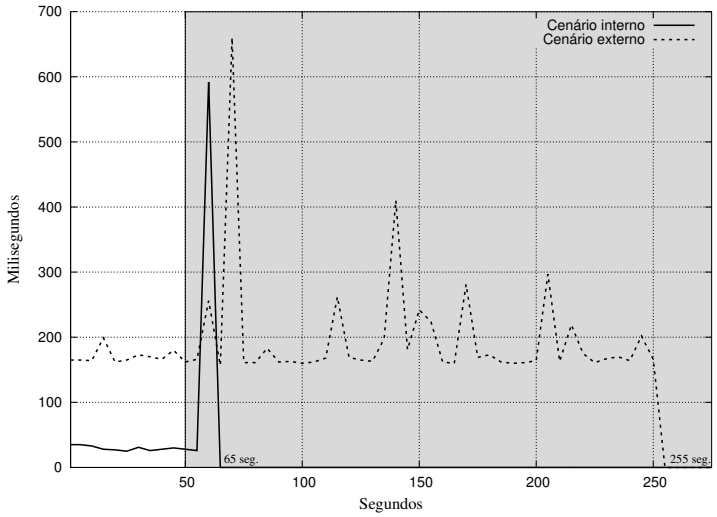


Figura 6. Tempo para efetivação do ataque de negação de serviço (DoS).

[3]O LRC corresponde ao "limiar" ilustrado na Figura 1, o qual é descrito na Seção 3.

5. Conclusão

O mecanismo *LegitimateBroker* consiste em uma ferramenta desenvolvida no intuito de combater a personificação de dispositivos que se comunicam por meio do protocolo MQTT. O mecanismo proposto apresenta uma alternativa de pouca sobrecarga computacional e energética. O mesmo pode ser adaptado de acordo com as restrições de cada cenário de implantação, podendo-se implementar diferentes algoritmos simétricos e tamanhos de chave. As experimentações práticas demonstram que o mecanismo *LegitimateBroker* é capaz de mitigar a personificação de dispositivos, ao passo que provê autenticidade, confidencialidade e evita ataques de reprodução de mensagens. Essas propriedades são alcançadas com uma redução de até 143,448 milissegundos do tempo de execução e consumindo 22,37% menos energia comparado ao uso de TLS.

Como trabalhos futuros, pretende-se estender o estudo realizado (i) envolvendo outros dispositivos da IoT, tais como, Raspberry Pi [Foundation 2012], sensores e atuadores; (ii) executando experimentos em larga escada, através de simulações; e (iii) estabelecer os valores adequados para renovação de chaves frente a outros ataques DoS, tais como, SYN Flood, Golden Eye, Hulk e SlowHTTPTest [Sharafaldin et al. 2018].

Agradecimentos

Os autores agradecem ao Fundo de Incentivo à Pesquisa (FIPE – Júnior/UFSM) pelo apoio no desenvolvimento deste trabalho.

Referências

Bhawiyuga, A., Data, M., and Warda, A. (2017). Architectural design of token based authentication of MQTT protocol in constrained IoT device. In *11th International Conference on Telecommunication Systems Services and Applications (TSSA)*, pages 1–4. IEEE.

Biju, S. and Shekokar, N. (2017). Security approach on MQTT based smart home. In *International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, pages 1106–1114. IEEE.

Bisne, L. and Parmar, M. (2017). Composite secure MQTT for Internet of Things using ABE and dynamic S-box AES. In *Innovations in Power and Advanced Computing Technologies (i-PACT)*, pages 1–5. IEEE.

Borgia, E. (2014). The internet of things vision: Key features, applications and open issues. *Computer Communications*, 54(1):1–17.

Degirmencioglu, A., Erdogan, H. T., Mizani, M. A., and Yılmaz, O. (2016). A classification approach for adaptive mitigation of SYN flood attacks: Preventing performance loss due to SYN flood attacks. In *Network Operations and Management Symposium (NOMS)*, pages 1109–1112. IEEE/IFIP.

Dudar, A. M., Martin, D. R., and Miller, K. J. (2018). Hydrocarbon sensor diagnostic. US Patent App. 15/435,741.

Empetel (2019). Battery Original Huawei Honor 8X. Disponível em: https://www.empetel.es/Battery-Honor8X. Acessado em: Maio/2019.

Esfahani, A., Mantas, G., Matischek, R., Saghezchi, F. B., Rodriguez, J., Bicaku, A., Maksuti, S., Tauber, M. G., Schmittner, C., and Bastos, J. (2019). A Lightweight Authentication Mechanism for M2M Communications in Industrial IoT Environment. *IEEE Internet of Things Journal*, 6(1):288–296.

Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A. (2003). The Many Faces of Publish/Subscribe. *ACM Computing Surveys*, 35(2):114–131.

Firdous, S. N., Baig, Z., Valli, C., and Ibrahim, A. (2017). Modelling and Evaluation of Malicious Attacks against the IoT MQTT Protocol. In *International Conference on Internet of Things (iThings) and Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and Smart Data (SmartData)*, pages 748–755. IEEE.

Foundation, R. P. (2012). Raspberry Pi. Disponível em: https://www.raspberrypi.org/. Acessado em: Maio/2019.

Gartner (2017). Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016. Disponível em: http://www.gartner.com/newsroom/id/3598917. Acessado em: Março/2019.

HiveMQ (2019). Plugin Developer Guide. Disponível em: https://www.hivemq.com/docs/3.4/plugins/ services.html. Acessado: em Maio/2019.

Jerkins, J. A. (2017). Motivating a market or regulatory solution to IoT insecurity with the Mirai botnet code. In *7th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 1–5. IEEE.

Khemissa, H. and Tandjaoui, D. (2016). A novel lightweight authentication scheme for heterogeneous wireless sensor networks in the context of Internet of Things. In *Wireless Telecommunications Symposium (WTS)*, pages 1–6. IEEE.

OASIS (2019). MQTT Version 5.0 OASIS Standard. Disponível em: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. Acessado em: Março/2019.

Santos, B. P., Silva, L., Celes, C., Borges, J. B., Neto, B. S. P., Vieira, M. A. M., Vieira, L. F. M., Goussevskaia, O. N., and Loureiro, A. (2016). Internet das coisas: da teoria à prática. *Livro de Minicursos do XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, 1:15–52.

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, pages 108–116.

Tanenbaum, A. S. and Wetherall, D. J. (2010). *Computer Networks*. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition.

Waher, P. (2015). *Learning Internet of Things*. Packt Publishing Ltd.

Wander, A. S., Gura, N., Eberle, H., Gupta, V., and Shantz, S. C. (2005). Energy analysis of public-key cryptography for wireless sensor networks. In *3dh International Conference on Pervasive Computing and Communications*, pages 324–328. IEEE.

Wells, L. J., Camelio, J. A., Williams, C. B., and White, J. (2014). Cyber-physical security challenges in manufacturing systems. *Manufacturing Letters*, 2(2):74 – 77.