This is the author's preprint version of the camera-ready article. The definitive version is published in the proceedings of the 2020 Medical Informatics Europe Conference (MIE2020). The final authenticated version is available online at `https://doi.org/10.3233/shti200177`

MQT-TZ: Secure MQTT Broker for Biomedical Signal Processing on the Edge

Carlos SEGARRA [a,b,1], Ricard DELGADO-GONZALO [a] and Valerio SCHIAVONI [b]
[a] *CSEM, Neuchâtel, Switzerland*
[b] *Université de Neuchâtel, Neuchâtel, Switzerland*

Abstract. Physical health records belong to healthcare providers, but the information contained within belongs to each patient. In an increasing manner, more health-related data is being acquired by wearables and other IoT devices following the ever-increasing trend of the *Quantified Self*. Even though data protection regulations (*e.g.*, GDPR) encourage the usage of privacy-preserving processing techniques, most of the current IoT infrastructure was not originally conceived for such purposes. One of the most used communication protocols, MQTT, is a lightweight publish-subscribe protocol commonly used in the Edge and IoT applications. In MQTT, the broker must process data on clear text, hence exposing a large attack surface for a malicious agent to steal/tamper with this health-related data. In this paper, we introduce MQT-TZ, a secure MQTT broker leveraging Arm TRUSTZONE, a popular Trusted Execution Environment (TEE). We define a mutual TLS-based handshake and a two-layer encryption for end-to-end security using the TEE as a trusted proxy. We provide quantitative evaluation of our open-source PoC on streaming ECGs in real time and highlight the trade-offs.

Keywords. wearables, mHealth, secure broker, MQTT, mosquitto, TrustZone

1. Introduction

Personalized health and medicine has the potential of being the next revolution in healthcare. It is also referred as the P4 medicine (Predictive, Preventive, Personalized, and Participatory), and provides the opportunity to benefit from more targeted and effective diagnoses and treatments [1]. One of the driving forces behind this tendency is the increasing medicalization of wearable technology [2] and mobile health (mHealth) apps [3]. In order to enable these technologies, complex processing IoT pipelines are gradually being deployed or repurposed. When the data-in-motion are vital signs, protecting user's privacy becomes a topic of crucial importance. Recent data protection regulations (*e.g.*, GDPR [4]) stress the importance of protecting sensitive information against malicious attackers or untrusted cloud providers.

Message Queuing Telemetry Transport (MQTT) [5] is one of the most commonly-used communication protocols in IoT. In spite of that, it is not included in some of the most extended Medical-Grade data exchange standards [6,7]. It follows a publish-subscribe architecture specially designed for environments with limited memory and re-

[1]Corresponding Author: Carlos Segarra; E-mail: carlossegarragonzalez@gmail.com.

This is the author's preprint version of the camera-ready article. The definitive version is published in the proceedings of the 2020 Medical Informatics Europe Conference (MIE2020). The final authenticated version is available online at `https://doi.org/10.3233/shti200177`

duced network bandwidth. In such circumstances, MQTT has proven to be more adapted to the IoT than classical protocols such as HTTP [8]. In MQTT, a *client* publishes data to a *topic* and the *broker* forwards it to each client that previously subscribed to it. The protocol is currently used in a variety of settings: data generation by sensors, pre-processing on the edge, and forwarding to the cloud. Examples include live heart-rate data [9,10], smart-grids [11], or building management systems [12]. Most MQTT implementations support TLS for transport security in the client-broker link, preventing malicious actors from spoofing application data. However, the broker itself still exposes a great attack surface [13].

In order to protect the privacy of health-related data, we present MQT-TZ, a secure implementation of the MQTT broker leveraging Arm TRUSTZONE, a Trusted Execution Environments (TEE), widely available on edge devices [14]. TRUSTZONE is a security feature available in recent Arm processors that enables system-wide hardware isolation for trusted software [15]. Our prototype builds atop `mosquitto` (`https://mosquitto.org`), a popular MQTT broker implementation, and includes persistent storage of client's keys in Arm's tamper-proof secure storage, as well as TEE-protected re-encryption of application data. These security enhancements make our ecosystem compliant with the *"Services Secure Interface"* [6] described by the Personal Connected Health Alliance, and address several attack vectors listed [7] by the IHE. We also consider linking our secure broker to a larger storage utility where data-at-rest is encrypted and its origin authenticated by MQT-TZ.

The paper is organized as follows. In Section 2, we describe the technical architecture and implementation of MQT-TZ. Then, in Section 3, we evaluate its performance and robustness at processing 1-lead ECGs in real time. Finally, in Section 4 we expose our main conclusions and propose further lines of research.

2. MQT-TZ: Securing the MQTT Broker

2.1. Architecture & Component Description

TRUSTZONE splits the system in a hardware-protected trusted part (the TEE) and an untrusted one (also called Rich Execution Environment, or REE). We add an encryption layer in MQTT's payload using client-specific keys stored in Arm's secure storage. This way, application data is only processed in clear inside the TEE. For the additional key-provisioning, we redefine the client authentication in the mutual TLS handshake to prevent the REE from gaining access to clients' keys.

The overall workflow looks as follows. Data travels two-fold encrypted from the client to the broker (Fig.1-❶). Once the client access is confirmed, Fig.1-❷, the subscribers for the given topic are retrieved and the payload forwarded (Fig.1-❸). Then, encrypted data is transferred to the TEE (Fig.1-❹). The origin and destination client keys are retrieved (❺-❼), the payload is re-encrypted, and sent back to the REE (Fig.1-❽) and to the subscriber (Fig.1-❾).

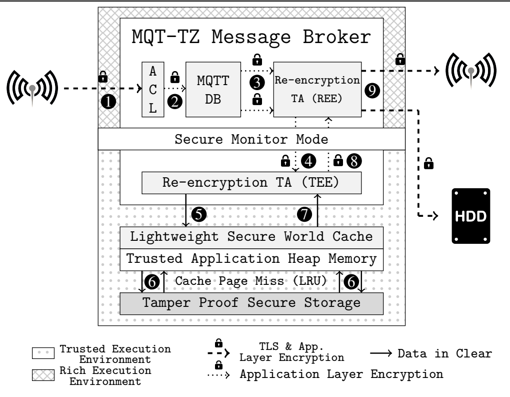
Two-Step Handshake. MQT-TZ defines and uses a two-step handshake that realizes broker and client authentication with end-to-end encryption from the client to the TEE. The handshake protocol requires minimal pre-provisioned cryptographic material. The broker (server in TLS nomenclature) authentication is done through TLS' handshake,

This is the author's preprint version of the camera-ready article. The definitive version is published in the proceedings of the 2020 Medical Informatics Europe Conference (MIE2020). The final authenticated version is available online at https://doi.org/10.3233/shti200177

supported by default in `mosquitto`. The client authentication is done through MQTT. It publishes its symmetric key, encrypted with the broker's TEE public key, to a specific write-only topic. This TEE key-pair is generated at device start-up time (secure boot) and derived from a Hardware Unique Key (HUK).

Layered Encryption & Access Control Mechanisms. Once the initial handshake is finished, MQT-TZ uses a two-layer encryption mechanism. First, the client-broker link is protected by TLS within MQTT. Second, MQTT's payload field is encrypted using the clients' symmetric key. Then, data is re-encrypted in the TEE (explained next) and sent to destination over MQTT-TLS. Doing so, we achieve end-to-end security relying on TRUSTZONE as a secure proxy.

Figure 1. MQT-TZ Architecture and data flow.


MQT-TZ Message Broker
A C L
MQTT DB
Re-encryption TA (REE)
Secure Monitor Mode
Re-encryption TA (TEE)
Lightweight Secure World Cache
Trusted Application Heap Memory
Cache Page Miss (LRU)
Tamper Proof Secure Storage
HDD
Trusted Execution Environment
Rich Execution Environment
TLS & App. Layer Encryption
Application Layer Encryption
Data in Clear

Payload Re-encryption. The core secure functionality implemented in MQT-TZ is the payload re-encryption. We link MQTT with a Trusted Application (TA) running inside the TEE that transfers the encrypted data to the Secure World, retrieves the origin and destination keys from secure storage, and re-encrypts the information. Currently, topic subscription lists and MQTT metadata are stored in a dedicated database (MQTT DB) in the REE. We plan on shadowing these structures and keeping them in the TEE.

Lightweight Cache. MQT-TZ embeds a lightweight cache that keeps the most recent keys in the TA's heap memory, and evicts the least used to persistent secure storage.

2.2. *Implementation Details*

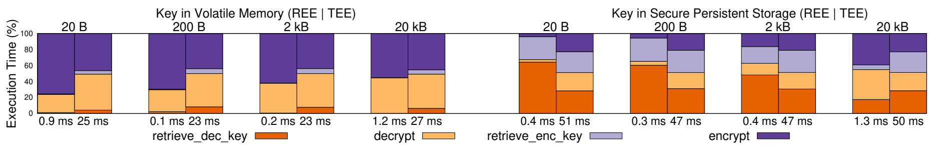
MQT-TZ is implemented in C. The current version of MQT-TZ adds 400 SLOC to `mosquitto` and the TA amounts to 1184 SLOC. The MQT-TZ TA relies on OP-TEE (`https://www.optee.org`), an open-source framework with native support for TRUSTZONE. Our implementation will be publicly available (`https://github.com/mqttz`).

Client and Server Authentication. The server-side authentication is done through vanilla TLS. We deploy MQT-TZ's secure broker in a device with a static IP address. Then, we bound the address to a domain name and use a certificate. We rely on *Let's Encrypt* (`https://letsencrypt.org/`) to get one and to authenticate the broker. The client-side authentication uses MQTT as communication layer, and `openssl` (v1.1.1a) for cryptographic primitives and operations. The integration with `mosquitto` exploits custom callbacks for each packet processing. In addition, we use MQTT Request/Response (RR) features (since v5) for the client's key exchange. To control access and R/W permissions to topics, we use `mosquitto`'s ACLs.

Trusted Application. We use OP-TEE to implement the payload re-encryption TA. Code developed within this framework has two parts: *(1)*, a host app that runs in the REE

This is the author's preprint version of the camera-ready article. The definitive version is published in the proceedings of the 2020 Medical Informatics Europe Conference (MIE2020). The final authenticated version is available online at `https://doi.org/10.3233/shti200177`

Figure 2. Re-encryption TA microbenchmark.



and acts as entry point and bridge to the TEE, and *(2)* a trusted API in the TEE that exposes different functions. MQT-TZ intercepts all MQTT packets being forwarded to the recipients, and feeds our host app with both client's IDs, and the encrypted data. We then perform the payload re-encryption using OP-TEE's storage and cryptographic libraries. TRUSTZONE not only provides isolation between worlds, but also between different TAs. Hence, we use the same secure API to store new keys during the handshake. For the key retrieval, we plan to implement a small LRU cache to store the most frequently used keys in the TA's heap, and the rest in persistent secure storage.

3. Evaluation and Results

In this section, we perform an evaluation of MQT-TZ. First, we benchmark the TA re-encryption with random data in order to understand the overhead introduced by the re-encryption; and then, we analyze the CPU and network throughput when monitoring vital signs in a real setting. For all experiments, we virtualize a Raspberry Pi 3 using QEMU-V8 (`https://www.qemu.org/`) running `mosquitto` v1.6.3 and OP-TEE v3.5.0.

3.1. TA Re-encryption

In Fig. 2, we show the breakdown of the time required to re-encrypt a single block of data for different sizes. The time is split in the time to retrieve each key (`retrieve_dec_key`, `retriev_enc_key`), `encrypt`, and `decrypt`. We can observe that AES is two orders of magnitude slower in the TEE. This is a consequence of OP-TEE not using hardware accelerators in contrast to `openssl` in the REE. Moreover, we observe sensible slowdowns when switching from in-memory to secure persistent storage.

3.2. Real-time ECG Processing

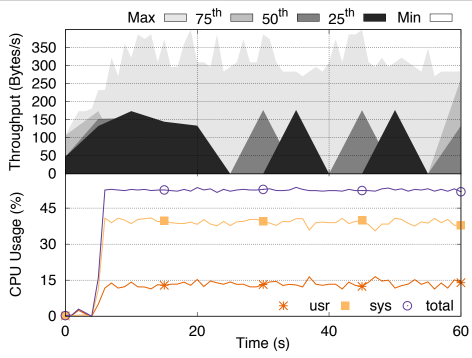
In this case, we test the resilience of MQT-TZ at sustaining the workloads that can be encountered in a hospital. For the experiments, we use the LTMS-S [16] platform developed by CSEM for the European Space Agency (ESA). In particular, we simulate 50 patients streaming in real-time 1-lead electrocardiograms (ECGs) at a frequency of 321.25 Hz. All ECGs are streamed toward a single MQT-TZ broker. In Fig. 3, we depict the outbound throughput generated by each publisher measured using `nethogs` (`https://github.com/raboof/nethogs`). We observe that at any given time only a subset of the publishers actually emits data. A single subscriber streams at 350 Bytes/s in the worst case, and the full collective generates between 3 to 5 kBytes per second. During the experiment, we recorded using `dstat` (`https://github.com/dagwieers/dstat`) the CPU load at the broker, shown in Fig. 3. We observe that after the initial peak, the overall CPU usage (both for `usr` and `sys` processes) stabilizes at 60%.

This is the author's preprint version of the camera-ready article. The definitive version is published in the proceedings of the 2020 Medical Informatics Europe Conference (MIE2020). The final authenticated version is available online at https://doi.org/10.3233/shti200177

4. Conclusion and Future Work

Motivated by the lack of secure-by-design communication protocols for the Edge, we presented MQT-TZ, our secure implementation of the MQTT broker using TRUSTZONE and showed its direct application in a in-hospital setting. The proposed system is robust and capable of managing 50 patients in real-time with a CPU usage of 60%. We plan to extend this work along the following directions. First, we will compare MQT-TZ against other publish-subscribe protocols and messaging queues. Second, we will study the performance overhead of MQT-TZ when deployed on large-scale scenarios. Finally, we intend to look into the energy footprint, an aspect of paramount relevance for edge deployments.

Figure 3. Workload test: Network throughput (top) and CPU usage (bottom)



References

[1] G. P. Cumming, "Connecting & collaborating - Healthcare for the 21st century," in *PAHI'2014*, 2014.

[2] J. Dunn, R. Runge, and M. Snyder, "Wearables and the medical revolution," *Pers. Med.*, vol. 15, no. 5, pp. 429–448, 2018.

[3] M.-P. Gagnon, P. Ngangue, J. Payne-Gagnon, and M. Desmartis, "m-Health adoption by healthcare professionals: A systematic review," *J. Am. Med. Inform. Assn.*, vol. 23, pp. 212–220, June 2015.

[4] The European Parliment and the Council of the European Union, "Regulation (EU) 2016/679," 2016.

[5] A. Banks and R. Gupta, "MQTT version 3.1.1," software, OASIS, Oct. 2014.

[6] Personal Connected Health Alliance, "Fundamentals of medical-grade data exchange," white paper, Continua, Sept. 2018.

[7] IHE PCD Technical Committee, "Medical equipment management (MEM): Medical device cyber security," white paper, IHE International, Inc., Oct. 2015.

[8] T. Yokotani and Y. Sasaki, "Comparison with HTTP and MQTT on required network resources for IoT," in *ICCEREC'2016*, pp. 1–6, Sept. 2016.

[9] K. Chooruang and P. Mangkalakeeree, "Wireless heart rate monitoring system using MQTT," *Procedia Comput. Sci.*, vol. 86, pp. 160–163, 2016.

[10] C. Segarra, R. Delgado-Gonzalo, M. Lemay, P.-L. Aublin, P. Pietzuch, and V. Schiavoni, "Using trusted execution environments for secure stream processing of medical data," in *Lect. Notes Comput. Sc.*, vol. 11534, pp. 91–107, 2019.

[11] A. Krylovskiy, M. Jahn, and E. Patti, "Designing a smart city internet of things platform with microservice architecture," in *FiCloud'2015*, pp. 25–30, Aug. 2015.

[12] Y. Lee, H. Hsiao, C. Huang, and S. T. Chou, "An integrated cloud-based smart home management system with community hierarchy," *IEEE. T Consum. Electr.*, vol. 62, pp. 1–9, Feb. 2016.

[13] Teserakt AG, "Is MQTT secure? (a report)," 2019.

[14] R. Liu and M. Srivastava, "VirtSense: Virtualize sensing through ARM TrustZone on Internet-of-Things," in *SysTEX'2018*, (New York, NY, USA), pp. 2–7, ACM, 2018.

[15] J. Amacher and V. Schiavoni, "On the performance of ARM TrustZone," in *DAIS'2019*, pp. 133–151, 2019.

[16] O. Chételat, D. Ferrario, M. Proença, J.-A. Porchet, A. Falhi, O. Grossenbacher, R. Delgado-Gonzalo, N. Della Ricca, and C. Sartori, "Clinical validation of LTMS-S: A wearable system for vital signs monitoring," in *EMBC'2015*, pp. 3125–3128, 2015.