HOW TO AUTHENTICATE MQTT SESSIONS WITHOUT CHANNEL- AND BROKER SECURITY

A PREPRINT

Reto E. Koenig
Department of Computer Science
Bern University of Applied Sciences
Switzerland
reto.koenig@bfh.ch

April 2, 2019

ABSTRACT

This paper describes a new but state-of-the-art approach to provide authenticity in mqtt sessions using the means of zero-knowledge-proofs. This approach completely voids session hijacking for the mqtt protocol and provides authenticity without the need for any network-security nor channel-security nor broker-based predefined ACLs. The presented approach does not require the broker to keep any secrets for session handling, what so ever. Moreover, it allows the clientID, which represents the identification for a session, to be publicly known. The presented approach allows completely anonymous but authentic sessions, hence the broker does not need any a priori knowledge of the client-party. As it is especially targeted for applications within the world of IoT, the presented approach is tuned to require only the minimum in extra power in terms of energy and space. The approach does not introduce any new concept, but simply fusions a state-of-the-art cryptographic zero knowledge proof of identity with the existing MQTT-5 specification. Thus no protocol extension is required in order to provide the targeted security properties. The described approach is completely agnostic to the application layer at the client side and is only required during mqtt-session establishment.

Keywords:
- mqtt
- anonymous authentication
- session hijacking
- zero knowledge proof
- Schnorr identification scheme
- channel security
- network security
- tls
- vpn
- Schnorr identification scheme

1 Introduction

MQTT [1] is designed as a robust, session-oriented protocol especially suitable for the world of IoT, where the clientID plays the central role for session management. The MQTT specification requires the clientID to be provided within the first data frame of the protocol during session establishment. The semantics of the clientID is to provides the unique way a session can be (re-)established between a client and the broker, without any further information. So the clientID is required to be unique per broker over time, hence, no collision of clientIDs should ever happen. As the clients are not aware of each other, but usually provide their own clientID, it must be drawn from large set of possible clientIDs so the probability of a collision of clientIDs is negligible. The protocol specification defines a minimum clientID space of order $2^{63}$. Hence, allows the theoretical security parameter of $2^\lambda$ with an approximate $\lambda = 137$.

The specification of mqtt does not directly address the immanent possibility of active session hijacking. This attack-vector allows an adversary to take over the session by simply (re-)establishing a connection using the clientID of the victim. Only indirectly, the mqtt specification tries to weaken that immanent attack-vector by providing the possibility for channel-security via TLS or even worse, by securing a whole network via vpn, and of course the heavy security requirement of the broker to be fully trusted in terms of secrecy of the clientID. However, the mqtt protocol specification gives room for customised authentication and even provides protocol intrinsic authentication solution via simple username password or via access control lists for ID-based session handling. These ID-based approaches do however
pose several disadvantages in the context of client management and security, as they require a separate and out-of-band on-staging-phase comprising the broker in order to manage clients by the identification scheme. Moreover, all these methods require the broker to keep secrets, shared amongst different clients and the broker.

The presented approach provides a full anonymous identification scheme based on the Schnorr non-interactive identification Scheme \[2\] without any of the drawbacks mentioned above. It makes use of well known cryptographic properties and allows the security model of the broker to be lowered from "fully trusted" to "honest but curious... without any privacy constraints". Again, there is no new concept introduced, but well known concepts are simply combined in order to provide the desired property.

The following sections will introduce the security- and adversary model this approach is operating in. This way, a discussion about the targeted adversary and the trust- and security assumptions is possible. Then the cryptographic- and protocol prerequisites are provided in order to understand the implementation that follows. The paper concludes by providing some quantitative and qualitative analysis of the approach.

## 2 Security Model

The main security parameter is denoted by \( \lambda \in \mathbb{N} \). We write \( a \leftarrow A(x) \) if \( a \) is assigned to the output of algorithm \( A \) with input \( x \). An algorithm is efficient if it runs in probabilistic polynomial time (ppt) in the length of its input. For the remainder of this paper, all algorithms are ppt if not explicitly mentioned otherwise. If \( S \) is a set, we write \( a \leftarrow_R S \) to denote that \( a \) is chosen uniformly at random from \( S \). For a message \( m = (m[1], m[2], \ldots, m[l]) \), we call \( m[i] \) a block, while \( l \in \mathbb{N} \) denotes the number of blocks in a message \( m \). For a list we require that we have a unique, injective, and efficiently reversible, encoding, which maps the list to \([0, 1]^{*}\).

The security model that covers the approach comprises of three actors, namely the channel, the broker and the client. There are no security assumptions required for the channel except of eventual availability. The channel is allowed to be modelled as an unreliable broadcast channel. The memory of the broker can be modelled as public readable memory, thus no secrecy is required on the broker side concerning the management of the clientIDs. The client is required to be able to keep a secret which might be either completely intrinsic to the client e.g. provided by a physical uncloneable function (puf), or injected via application level.

The security properties of the actors define the adversary model. The adversary which is assumed to be restricted to be in ppt only is allowed to have full knowledge about the channel and the memory of the broker at any time (full-take). The adversary, however, is not able to extract the secret from the client.

## 3 Prerequisites

In the following section the prerequisites are described in order to understand the solution. As this solution proposes a cryptographic approach within the given protocol specification, both aspects are described here.

### 3.1 Cryptographic Prerequisites

The cryptographic protocol used for this solution is the Schnorr identification scheme as described in the RFC-8235\[2\]. In order to render this paper self contained, a brief summary of the RFC-8235 is given here. Please rever to the full RFC-document for further details.

The Schnorr NIZK proof can be implemented over a finite field or an elliptic curve (EC). The technical specification is basically the same, except that the underlying cyclic group is different. For simplicity, this document describes the approach within the finite field, whereas in the RFC-document, both versions are described.

Let \( p \) and \( q \) be two large primes with \( q \mid p - 1 \). Let \( \mathbb{G}_q \) denote the subgroup of \( \mathbb{Z}_p^* \) of prime order \( q \), and \( g \) be a generator for the subgroup.

**Statement** \( \text{NIZKP}[(x) : y = g^x] \) Alice knows the discrete logarithm \( x \) of the value \( y \) to the base \( g \) (within \( \mathbb{Z}_p^* \)).

**Public input** \( p, q, g, \mathbb{Z}_p^*, \mathbb{G}_q, \) and the cryptographic hash function \( H(x) \).

**Prover’s (Alice) private input.** \( x \in \mathbb{Z}_q \) such that \( y = g^x \mod p \).

**P \((t,s)\) V** Alice chooses random \( \omega \leftarrow_R \mathbb{Z}_q \), calculates \( t = g^\omega \), calculates \( c = H(g|t||y||\text{public data}) \), calculates \( s = \omega + x \cdot c \mod q \)

**Verification.** Bob accepts the proof only if \( y \in \mathbb{Z}_p^* \) and \( g^s = t \cdot y^c \mod p \).
3.2 Protocol Prerequisites

The MQTT-Protocol describes the possibility for authentication exchange in section 3.15 and section 4.12. The authors of the MQTT-protocol were aware of the problem of session-hijacking and reply-attacks. Especially in section 5.4.5 of the protocol specification they give hints on how to mitigate the attack attempts. But all these mitigation approaches are not focusing on the problem at hand but augment the security assumptions of the channel or even network, by requiring the channel to be secured (TLS) or even by asking for a secure network environment (VPN).

As already advertised in the security model, the approach does not require such heavy security assumptions. It is based on a challenge response mechanism already foreseen by the specification, very much like the described Salted Challenge Response Authentication Mechanism (SCRAM) in the non-normative part of the protocol description. Thus the approach perfectly fits within the given protocol boundaries.

4 Protocol Description

The main idea of this protocol is to chose the clientID as a group element $y \in G_q$. In the notation for the multiplicative group, the client chooses $y = g^x$ with $x \leftarrow \mathbb{Z}_q$. In order to void replay-attacks, the broker provides the client with a nonce $n$ to be used as public data for the non-interactive proof. Only if the client is able to provide the proof $(t, s)$ within the same network connection as the provided challenge, the broker proceeds with CONNACK $rc=0$, in all other cases it disconnects with DISCONNECT and some reason code e.g. $rc=0x83$. This way, the client provides an anonymous identification each time it

5 Implementation

The implementation requires some addition to the client- and broker software, that provide the mqtt ability.

**Client** Prior to the connection the client calculates its clientID by deriving a value $x'$ from its secret value (can be the result of a puf or an out-of-band injected value) and getting $y$. This then is mapped into the the allowed set of mqtt-clientIDs $clientID \leftarrow _{mqtt} y$. After the client established a connection to the broker, the client starts an mqtt session by sending the connect frame containing the authentication flag in expectancy of a broker authentication response, where a broke-challenge nonce $n \leftarrow \{0, 1\}^{\lambda}$ has to be provided. The client then calculates the non-interactive zero-knowledge proof.

The client then sends the proof as an authentication response within the same connection to the broker. If the client needs to re-establish the session, the very same procedure is executed as if the client would establish the session for the very first time.

**Broker** When there is a client connecting to the broker the provided clientID is only accepted, when the authentication succeeds within the same connection. The broker draws a random challenge nonce $n \leftarrow \{0, 1\}^{\lambda}$ as a response to the first authentication frame within a new connection. The response delivered from the client consists of the NIZKP. The broker checks this proof.

Only when this proof is accepted by the broker, the clientID is accepted and the broker behaves in its usual way. Except, of course, it does not have to keep the clientID secret as it had without this authentication scheme.

6 Analyses

First a fair warning: Due to the 'short' maximum length of the clientID covered by the mqtt specification in relation to cryptographic constraints, it is important to implement the proof using the mathematics of elliptic curves. This way, the security parameter $\lambda$ provided by the mqtt specification provides a maximum $\lambda = 70$. According to cryptographic sources [3], this is already a very low security. But using finite fields would lower $\lambda$ by at least another factor of 20 and would not sustain an attack of any computational moderate potent adversary. Therefore, as an advice, the client and the broker must be enabled to accept a clientID set of order $\geq 2^{256}$. Using the alphabet supported by the mqtt specification this results in a minimum of 43 characters for the clientID in order to get a minimum $\lambda = 128$.

6.1 Energy- and Space Requirements

In terms of computing power, the client must be able to execute two operations on an elliptic curve and one cryptographic hash per connection. The expensive calculations on the elliptic curve can even be lowered down to a single
operation if the clientID derived from the according secret can be stored in the system. The computing power on the broker side must allow to calculate a minimum of two operations on the elliptic curve and one cryptographic hash per connection. Furthermore, the broker requires to provide a high entropy challenge nonce \( n \) per connection. Either, the broker gets true randomization, which is a very difficult task, or the broker uses an initialization vector (secret iv) seed to initialize a pseudo random generator function. It can keep seeding the pseudo random generation function by the answers it receives from the client connections. This way, the entropy of the client secret can be used to influence the available entropy on the broker, where natural sources of entropy are sparse by nature.

6.2 Passive Adversary

The provided implementation allows the broker to reside in a very relaxed security model. No secret whatsoever is required anymore concerning the clientID. The provided implementation further does not require any special treatment of the channel used for communication, as no secret is sent over the channel at all. As long as the underlying network protocol provides authenticity in terms of source- and destination during connection, the mqtt session cannot be hijacked at all. The only remaining party within a strict security model is the client. It must be able to keep the secret safe from being extracted.

6.3 Active Adversary

As long as the adversary is not in possession of the secret \( x \) kept by the client, it cannot provide a valid proof of knowledge to the broker. However, if the adversary model is relaxed, it can start a man-in-the-middle attack. This allows the adversary to impersonate the client without the client nor the broker to gain any knowledge of that. Thus, the client is required to be sure about the public identity of the desired broker. So we are back to TLS (and all its pros and cons) with server certificate, in order to provide server-identity ‘only’ in order to prevent the man-in-the-middle attack. So, one might argue, that the title of this paper might be misleading, but it must be stressed out, that TLS is not required to prevent clientID stealing but for broker-identity in order to prevent the active adversaries man-in-the-middle attack.

References

[1] Ken Borgendale Andrew Banks, Ed Briggs and Rahul Gupta. mqtt-v5.0. OASIS Standard, mqtt-v5.0-cos02:1–137, 2019.

[2] Feng Hao. Schnorr non-interactive zero-knowledge proof. RFC, 8235:1–13, 2017.

[3] Damien Giry. Keylength – cryptographic key length recommendation, 2008. http://www.keylength.com/ (accessed 2019-03-28).