Design and Implementation of Efficient IoT Authentication Schemes for MQTT 5.0

Hung-Yu Chien[*], *Pin-Ping Ciou*

Department of Information Management, National Chi Nan University, Taiwan
hychien@ncnu.edu.tw, s107213055@mail1.ncnu.edu.tw

Abstract

MQTT (Message Queue Telemetry Transport) is one of the most popular Internet of Things (IoT) communication protocols, owing to its lightweight and easiness to use. The previous MQTT standards (including version 3.1 and its precedents) do not provide proper security functions; instead, they assume the adoption of SSL/TLS in the underlying layer. However, it not only incurs larger overhead but also hinders the development of suitable authentication/confidentiality protection to suit various MQTT deployment scenarios. The newest MQTT standard called MQTT 5.0 responds to these challenges by supporting the Enhanced Authentication framework in which designers can design and implement any secure authentication mechanisms within the framework.

This paper designs and implements two efficient authentication protocols, using the MQTT 5.0 Enhanced Authentication framework. One is simple challenge-response authentication scheme, and the other is an anonymous challenge-response authentication scheme. We extend HiveMQ platform to implement the schemes and evaluate the performance. The results show that the proposed schemes demand only hundred few more milliseconds to achieve much robust authentication, compared to the simple identity-password authentication.

Keywords: Internet of Things, MQTT, Authentication, Challenge and response, Anonymity

1 Introduction

The IoT technologies is a very promising area for collecting and analyzing data from potential many IoT devices. To facilitate the easy transmission of IoT data and signals where IoT devices are usually resources-batteries-limited, an IoT communication protocol is expected to be more efficient, compared to conventional networking protocols. Among many IoT communication protocols, MQTT has become one of the most popular IoT communication protocols, owing to its high efficiency and simplicity to use.

Previous MQTT standards [1-3], focusing on being lightweight, do not provide enough security protection like authentication, integrity, and confidentiality; instead, the security of the MQTT deployments are assumed be dependent on the use of Secure Sockets Layer (SSL)/ Transport Layer Security (TLS) [4] in the underlying layer, which demands quite a few cryptographic computations and communication overhead of the certificates [5] and lacks the feasibility of supporting anonymity. Anonymity is crucial for some IoT scenarios where the identities could be used to refer one's movement and possible locations. The weaknesses of the previous MQTT standards and improper deployments have exposed these MQTT systems to serious threats. Many mis-configured MQTT servers and the security weakness have been extensively reported or evaluated [6-7].

There exist many proposals (like [8-20]) to enhance the security of MQTT systems. Some of these security-enhancement proposals provide hardware/architecture support for security protocols, some propose new symmetric-key-based/asymmetric-key-based security protocols, and some aim at designing authenticated key agreement schemes that are compatible with the previous MQTT packet formats and Application Interfaces (API). Chien [20-21] proposes general and efficient client-server authentication and device-to-device authentication.

In light of various security concerns and desirable function extensions, the new MQTT standards called MQTT 5.0 have been ratified [23-24]. Regarding the security, MQTT 5.0 supports the Enhanced Authentication framework in which new authentication API (called AUTH API) and new parameters (like Authentication method and Authentication data) could be used to design/implement authentication and key agreement schemes within the MQTT 5.0 context. With the new security framework and new functions, MQTT 5.0 is expected to greatly enhance the security functions and to support new application use cases.

However, none of the previous MQTT enhancements like [8-20] target on the MQTT 5.0 systems. In this paper, we design and implement two efficient authenticated key schemes using the MQTT 5.0 AUTH APIs. We implement the two schemes using the HiveMQ's MQTT platform [25]. The evaluation of the prototypes show that the proposed schemes ensure much robust authentication security with only insignificantly additional more authentication latency, compared to the simplest identity-password authentication.

2 Related Work and Background

A MQTT system consists of a set of clients and a broker who acts as an intermediary among the clients (publishers and subscribers). The message exchange among clients, via brokers, is based on the concept of "topic". A broker would forward the published data of a topic from a publisher to those subscribers who subscribe the same topic.

*Corresponding Author: Hung-Yu Chien; E-mail: hychien@ncnu.edu.tw

DOI: 10.53106/160792642023052403012

There exist many proposals and publications concerning the security of MQTT systems. The works like [6-8] report the security weaknesses and evaluate the security threats. The MQTT-security-enhancement proposals could be classified from several perspectives. From the architecture's perspective, Lesjak et al. [9] integrate an IoT device with a TLS-capacity-embedded hardware to relieve the IoT device from the burden of handling TLS connections; Rizzardi et al. [10] and Neisse et al. [11] respectively focus on the key management framework and the policy management framework of MQTT systems so that the messages could be flexibly and dynamically encrypted and accessed according to the flexible policies; Niruntasukrat et al. [12] augment MQTT authentication and authorization by integrating with OAuth mechanism. For those customized authenticated key agreement schemes for MQTT, the previous proposals could be classified into two categories- public-key-based schemes or symmetric-key-based schemes. In general, symmetric-key-based schemes like [7, 10-18] might own the benefits of lightweight computations and lightweight communication; some public-key-based schemes like [19-20] might own the benefits of better scalability in terms of key management; Singh et al. [20] augmented existent MQTT protocols with Key/Ciphertext Policy-Attribute Based Encryption the Key/Ciphertext Policy-Attribute Based Encryption (KP/CP-ABE) , which would demand high computational cost and poor scalability, even for only a small number of attributes. Chien et al.'s two-phase authenticated key agreement scheme [16], via a side channel (for example a socket connection), establishes an authenticated session key between a client and a broker in the first phase; and it uses the hashed value of the session key as the password field in the MQTT 3.1 CONNECT API in the second phase; this arrangement can easily integrate any secure key agreement scheme with the MQTT 3.1 API. Chien et al. [18] propose a hierarchical MQTT architecture with edge computation to improve the message latency. Chien et al.'s implementations are based on the Mosca platform [26].

Most of the existent authenticated key agreement schemes for MQTT concern the link security- the security between a client (a publisher or a subscriber) and a broker, and only very few works tackle the end-to-end security (from a publisher to a subscriber). The end-to-end security can provide some desirable benefits: (1) the broker could get rid of the loading of decrypting a message from a publisher and then re-encrypting the message several times for all the subscribers; (2) the content of MQTT messages could be kept secret from a curious broker. Mektoubi et al. [19] propose a topic-centric key distribution in which the system assigns a specific certificate for each topic, a publisher encrypts its messages using the public key of the topic certificate, and a legitimate subscriber who owns the corresponding private key can decrypt the messages. Chien et al. [17] propose a MQTT group communication scheme in which a broker automatically generates a topic for topic-centric group key updating for any specific topic, and all the legitimate clients are automatically registered with this group-key updating topic; a trusted daemon periodically updates the group keys and securely delivers the newest group keys to all legitimate publishers and subscribers, via the broker; a publisher encrypts its messages using the group key, and the subscribers of this topic decrypt the received messages using the same group key.

Chien [21-22], based on hashing and hash composition, propose an anonymous client-to-server authentication and an anonymous device-to-device authentication. In this paper, we modify and extend the anonymous client-to-server authentication in [20-21] to design and implement an anonymous client-to-broker authentication for MQTT 5.0. The differences between this work and several MQTT precedents are summarized here. (1) The precedents [16-18] are based on MQTT3.1 and the Mosca platform [25] while this work being based on MQTT 5.0 and the HiveMQ platform [26]. (2) The precedents handle the management works (like user management, topic management, and key management) and the challenge-and-response-based authentication using the two-phase technique to be compatible with MQTT 3.1 CONNECT APIs; but this work designs and implements two hash-based authentication schemes (one is anonymous and the other is not) within the MQTT 5.0 Enhanced Authentication framework and APIs.

The contributions of this work are outlined as follows.

(1) We design and implement two hash-based client-to-broker authentication schemes using the MQTT 5.0 Enhanced Authentication framework and APIs.
(2) We design and implement the first anonymous client-to-broker authentication for MQTT 5.0 to meet the identity privacy protection for many IoT application scenarios.
(3) We specify the proposed schemes using the High-Level Protocol Specification Language (HLPSL) [28], and verify the key security properties of the proposed schemes using the formal verification tools- the Automated Validation of Internet Security Protocols and Applications (AVISPA) [29].
(4) We analyze the performance of the protocols, and evaluate the performance of the implementations.

3 Two Efficient Authentication Schemes for MQTT 5.0

In this section, we introduce our two MQTT5.0-compatible authentication schemes where one provides identity privacy protection (anonymity) and the other does not. Before describing the protocols, we first give a simple review of MQTT 5.0 Enhanced Authentication framework in Section 3.1.

3.1 MQTT 5.0 Enhanced Authentication Framework

The MQTT 5.0 Enhanced Authentication Framework provides new AUTH APIs (and packets) and new authentication-related fields called Authentication method and Authentication data (they will be referred as Auth_id and Auth_data for the rest of this paper for short). With this framework, designer and implementors can design and implement any secure authenticated key agreement schemes within Mthe MQTT 5.0 context. A simplified protocol stack with these APIs is shown in Figure 1.

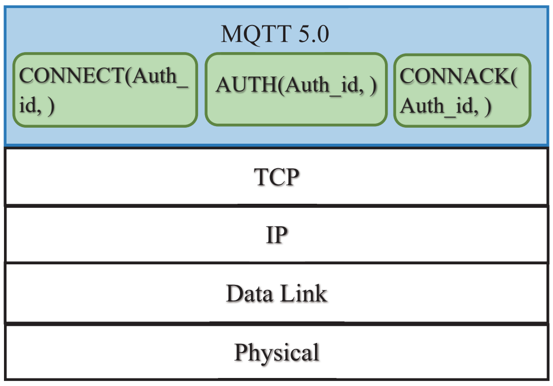


Figure 1. MQTT 5.0 enhanced authentication protocol stack

A client (a publisher or a subscriber) initiates its connection by sending a CONNECT request in which the Auth_id and Auth_data are specified to notify the broker which authentication method is chosen. During the authentication process, they might exchange several AUTH messages which convey the Auth_data for the specified AUTH_id. Finally, CONNACK is sent to notify the result of the process. During the whole process, AUTH_id should be included in each message to ensure the right method is referred to.

3.2 The Proposed Schemes in the MQTT 5.0 Context

Based on the MQTT 5.0 Enhanced Authentication framework, we introduce a hash-based Challenge-Response authentication scheme in Section 3.2.1 and a hash-based anonymous authentication in Section 3.2.2. The notations are summarized in Table 1.

3.2.1 A Hash-based Challenge-Response Authentication within MQTT 5.0 Context

In the initialization, a client C_i registers itself at a broker, and gets two keys- a *device_key* and a *topic_key*; The *device_key* is specific to each device; the *topic_key* is shared among all legitimate clients of that topic. In the following computations and protocol flow, the two keys would be concatenated as one unified key as :=*device_key*||*topic_key*. This arrangement has the merits that (1) when a client is authenticated using the unified key, both the device and its authorization to access the specific topic are verified at the same time.

The flow of the first scheme within the MQTT 5.0 context is depicted in Figure 2. In the first step, a client sends its CONNECT request in which the chosen authentication AUTH_id="CR" denoting the specific authentication method we develop.

In the CONNECT message, the client also sends its identity, the topic, and the first challenge denoted as *C1*. When the broker receives the request, it first retrieves the corresponding *device_key* and *topic_key* from its database using the received identity, lets K_i: = *device_key* ||*topic_key*, computes the response $R1 = h_1(K_i, C1)$, randomly chooses its challenge *C2*, and sends back AUTH(Auth_id=" CR", *R1*, *C2*) in the second step.

Table 1. The notations

C, B. ID_i,; ID_B	Client; Broker. ID_i: ID_B identities of a client and the broker respectively.
$h_1()$, $h_2()$, *Mac*()	Two cryptographic hashing functions. *Mac*(): message authentication code function, which could be implemented, using HMAC [27].
device_ key; *topic_ key*	The secret key between a device and the broker; the secret key shared among all clients and the broker for a specific topic.
C_i,; K_i	C_i: *i*th device; K_i: = *device_key; topic_key* the secret key shared between C_i and *B*.
N_i	N_i: C_i*'s seed* (a random value) for generating the chain of seed values. Each seed value is used to generate the corresponding pseudonym $PN_{i,k}$.
$h_1{}^j(N_i)$	$h_1{}^j(N_i) = h_1(h_1(\ldots h_1(N_i)))$ means h_1 being applied *j* times.
$PN_{i,0}$, $PN_{i,k}$	$PN_{i,0}$ denotes C_i' *s pseudonym* $PN_{i,0} = h_2(N_i)$. $PN_{i,k} = h_2(h_1{}^k(N_i))$ denotes the *k*th *pseudonym* successor of $PN_{i,0}$
PV, PV_i	*PV*: *Pseudonym Vector*; $PV_i = \{PN_{i,0}, PN_{i,1}, \ldots, PN_{i,k}, PN_{i,w1}\}$ is the pseudonym vectors pre-calculated by *B* for C_i.
w_1	the window size of the precalculated pseudonym vectors.
G_{ECC}, *P*, *q*	G_{ECC} is a cyclic multiplicative group of an order *q*, where the Computational Diffie–Hellman Problem (CDHP) is hard; *P* is a generator of G_{ECC}. Here, we let G_{ECC} be a group in the elliptic curve setting for the short key size.
Auth_id, Auth_data	The specified authentication method in the AUTH API, the authentication data within AUTH API.

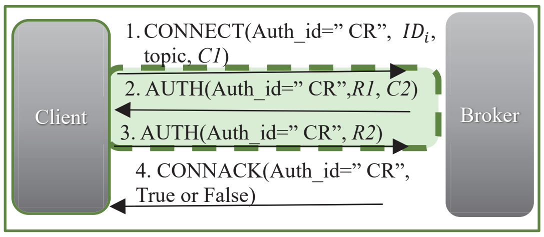

Client
Broker
1. CONNECT(Auth_id=" CR", ID_i, topic, *C1*)
2. AUTH(Auth_id=" CR",*R1*, *C2*)
3. AUTH(Auth_id=" CR", *R2*)
4. CONNACK(Auth_id=" CR", True or False)

Figure 2. The protocol flow of a hash-based Challenge-Response authentication within MQTT 5.0 context

The client verifies the received *R1*; if the verification succeeds, then it responds AUTH(Auth_id=" CR", *R2*) in the 3rd step, where $R2 = h_1(R1, C2, K_i)$. Finally, the broker verifies the received *R2* and responds "True(Success)" or "False(failure)" accordingly. The final session key is defined as K_{sess}: = *hash*(K_i, C1, C2, ID_i, *topic*). After the authentication process, the client and the broker use the session key to encrypt/decrypt messages.

3.2.2 An Anonymous Hash-based Challenge-Response Authentication within MQTT 5.0 Context

Here, we aim at designing an anonymous client-broker authentication within the MQTT 5.0 framework. The channels between clients and brokers are susceptible to various attacks. The brokers are trusted, but the clients might be compromised. The scheme consists of two phases: the registration–initialization phase and the authentication phase.

3.2.2.1 The Registration-initialization Phase

The scheme is based on the composited hashing depicted in Figure 3. Like the scheme in Section 3.2.1, each client registers itself at a broker, and gets two keys- a *device_key* and a *topic_key*, and the unified key as =*device_key*||*topic_key*. Figure 3 shows C_i*'s* seed and pseudonyms; C_i repetitively applies $h_1()$ on its seed N_i to update N_i, and each active N_i is applied $h_2()$ to have the active pseudonym $h_2(N_i)$.

Figure 4 shows the three algorithms used by a client and a broker, where the algorithm PN_Seed_Update(N_i) is run by C_i to update the seed and to generate the current *PN* in each session. The algorithm PV_Iinitialize(C_i, N_i) is run by a broker to initialize a client's pseudonym vector. When a broker (say *B*) accepts C_i*'s* registration, it initializes the client's pseudonym vectors $PV_i = \{PN_{i,0}, PN_{i,1}, \ldots, PN_{i,k}, \ldots, PN_{i,w1}\}$, where $PN_{i,j} = h_2(h_1{}^j(N_i))$, $j = 0{\sim}w_1$. This design of the pseudonym vector consisting of the current pseudonym and the future w_1 pseudonyms could properly tackle the possible out-of-synchronization issues caused either by the unreliable connection or the possible DoS attacks. When either the un-reliable connection or the DoS attacks happen, a client's requests might be blocked or be lost, and the client would repetitively update its seed and pseudonym, and retry its requests. In such situations, the sent pseudonym would be still in the range of the broker-prepare additional w_1 pseudonyms. Of course, the size of the window w_1 is determined according to tradeoff between the cost of the additional storage and the possible of the threats.

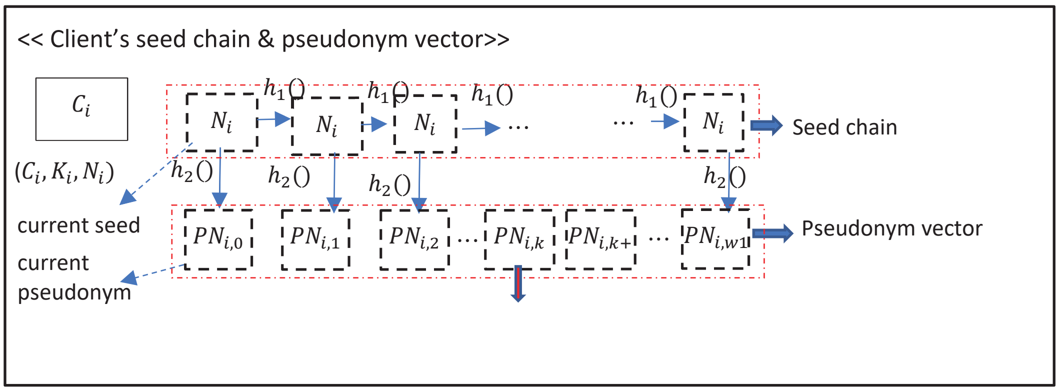
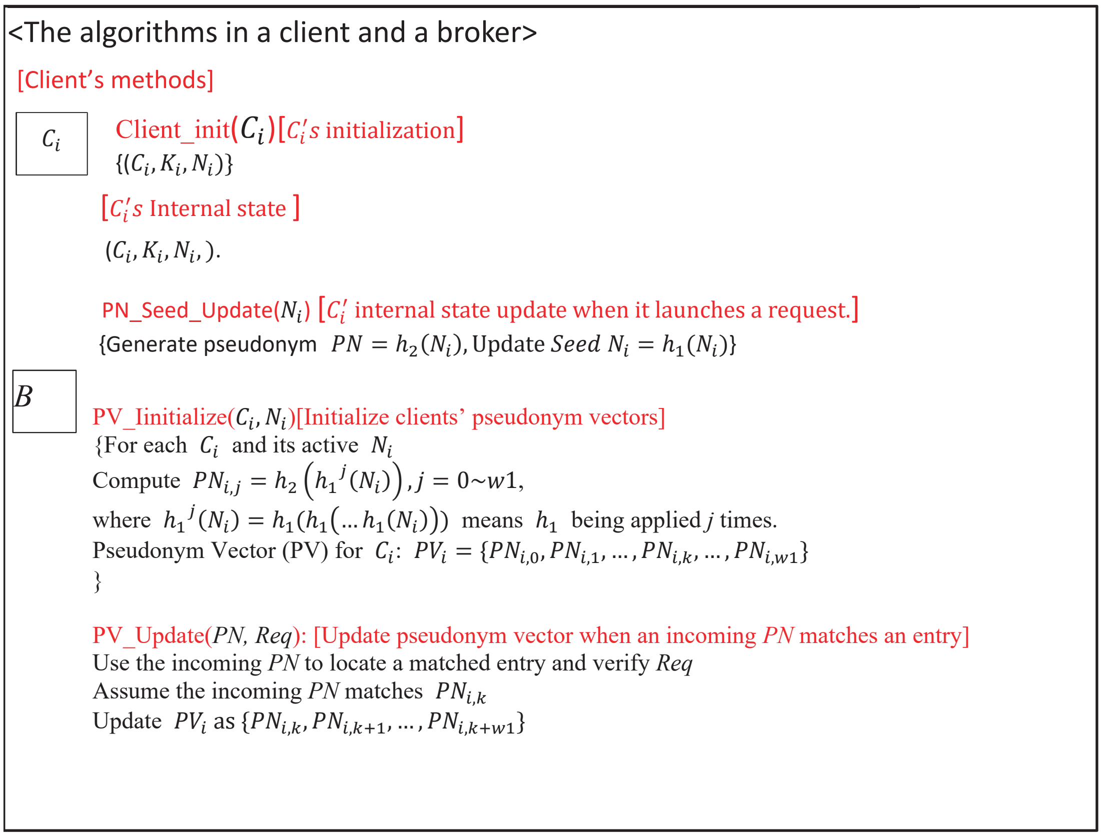


Figure 3. C_i's Seed and pseudonyms


<The algorithms in a client and a broker>

[Client's methods]

C_i

Client_init(C_i)[C_i'*s* initialization]
$\{(C_i, K_i, N_i)\}$

[C_i'*s* Internal state]
$(C_i, K_i, N_i,)$.

PN_Seed_Update(N_i) [C_i' internal state update when it launches a request.]
{Generate pseudonym $PN = h_2(N_i)$, Update *Seed* $N_i = h_1(N_i)$}

B

PV_Iinitialize(C_i, N_i)[Initialize clients' pseudonym vectors]
{For each C_i and its active N_i
Compute $PN_{i,j} = h_2\left(h_1{}^j(N_i)\right), j = 0{\sim}w1$,
where $h_1{}^j(N_i) = h_1(h_1(\ldots h_1(N_i)))$ means h_1 being applied *j* times.
Pseudonym Vector (PV) for C_i: $PV_i = \{PN_{i,0}, PN_{i,1}, \ldots, PN_{i,k}, \ldots, PN_{i,w1}\}$
}

PV_Update(*PN*, *Req*): [Update pseudonym vector when an incoming *PN* matches an entry]
Use the incoming *PN* to locate a matched entry and verify *Req*
Assume the incoming *PN* matches $PN_{i,k}$
Update PV_i as $\{PN_{i,k}, PN_{i,k+1}, \ldots, PN_{i,k+w1}\}$

Figure 4. The algorithms used by a client and a broker

3.2.2.2 The Anonymous Authentication Phase

Figure 5 shows the authentication flow of the anonymous scheme within the MQTT 5.0 context. Basically, C_i sends its current pseudonym $PN = h_2(N_i)$ for B (the broker) to identify the client, and they use the secret key K_i to perform a Challenge-Response authentication where the exchanged Diffie-Hellman keys are used to generate the session keys. The details are described as follows.

Step 1. $C_i \rightarrow B$: CONNECT(AUTH_id = "anon", PN, ID_B, X, mac_1) C_i executes PN_Seed_Update(N_i) to generate the pseudonym PN and to update N_i ; C_i chooses $x \in_R Z_q^*$, and calculates $X = xP$ and $mac1 = Mac(K_i, PN, X)$. C_i sends the CONNECT(AUTH_id = "anon", PN, ID_B, X, mac), where AUTH_id = "anon" denotes the anonymous authentication method.

Step 2. $B \rightarrow C_i$: AUTH(AUTH_id = "anon", ID_B, PN, Y, $mac2$) B calls PV_Update(PN, $mac1$), which uses the received PN to search a matched entry in its pseudonym vectors to identify the client, and to update the client's pseudonym vectors. B chooses $y \in_R Z_q^*$, and calculates $Y = yP$ and its response $mac2 = Mac(K_i, PN, ID_B, X)$, which is a response to C_i's challenge PN *and* X). S wraps the messages in AUTH API specified in Step 2.

Step 3. $C_i \rightarrow B$: AUTH(AUTH_id = "anon", PN, ID_B, $mac3$) C_i verifies the received $mac2$, calculates $mac3 = Mac(K_i, PN, ID_B, Y)$, and computes the session key $K_{sess} = h_1$("KEY", ID_B, PN, K_i, xyP). It sends the AUTH packet specified in Step 3. Upon receiving C_i's response, B verifies $mac3$, and sends back the verification result in Step 4. If the verification succeeds, then it computes the session key.

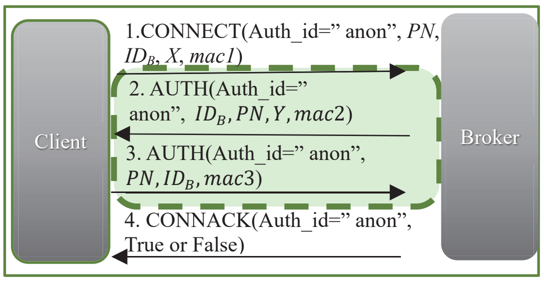

Client
1.CONNECT(Auth_id=" anon", PN, ID_B, X, $mac1$)
2. AUTH(Auth_id=" anon", $ID_B, PN, Y, mac2$)
3. AUTH(Auth_id=" anon", $PN, ID_B, mac3$)
4. CONNACK(Auth_id=" anon", True or False)
Broker

Figure 5. The protocol flow of an anonymous PN-based authentication within MQTT 5.0 context

4 The Implementations and Performance

We implement our proposed schemes by extending the opensource MQTT 5.0 platform provided by HiveMQ [25]. In the implementation, when the broker initializes, it loads the Enhanced Authentication Provider from the AuthExtensionMain module; when the broker receives a connection request, it retrieves the Authentication_method field from the request, and it instantiates the corresponding Authenticator from the Enhanced Authentication Provider to handle the authentication process specified in Section 3.2.1 and Section 3.2.2 respectively.

Figure 6 shows the network setting of the experiments for evaluating the performance of the prototypes. We arrange the broker and the clients locating in two subnetworks of the same building; the arrangements let the communication between a broker and a client to be two-hop. The clients are running on a raspberry pi model 4. The brokers are running on a notebook "HP Laptop 14s-cf0xxx" with Windows 10.

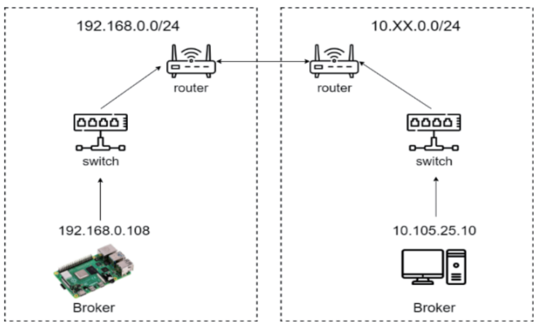


Figure 6. The network setting of the experiments

Table 2 lists the software of the clients and the broker. To evaluate the proposed schemes, we set up several authentication schemes in two different running environments: one is the programs compiled into executable commands, and the other is programs running in the Integrated Development Environment (IDE).

Table 2. The software settings of several client-broker authentications

Authentication method	Broker	Client
Simple identity-password	hivemq-ce-2021.1 (execution command)	MQTT CLI (execution command)
Simple identity-password	hivemq-mqtt-serverSimpleAuthentication (application in IDE)	hivemq-matt-client-SimpleAuthentication
Enhance Authentication - Challenge and Response	hivemq-mqtt-serverCRChallenge_ Anonymous (application in IDE)	hivemq-mqtt-clientPublisher& Subscriber-CRChallenge
Enhance Anonymous - Challenge and Response	hivemq-mqtt-serverCRChallenge_ Anonymous (application in IDE)	hivemq-mqtt-clientPublisher_ AnonymousAsync

In Table 2, the authentication method with "simple identity-password" is the simplest authentication which uses client's identity and password to authenticate the client. The method with "EnhanceAuthentication - Challenge and Response" is the implementation of our first scheme- the classic challenge-response authentication in the MQTT 5.0 context. The method with "EnhanceAnonymous - Challenge and Response" is the implementation of our anonymous challenge-response authentication in the MQTT 5.0 context. The "MQTT CLI" is one MQTT client execution command provided by HiveMQ team and it has been optimized to get better running performance. The other clients are those programs which we modify from the opensource. A program running as an executable command demands much less time than the same program running in the IDE, as an executable command has already bound and loaded the required libraries.

Table 3. Average authentication latencies (in ms) for several MQTT 5.0 authentication schemes

Authentication method	Average auth. delay (only normal cases)
Simple identity-password in command mode	0.15375
Simple identity-password in IDE mode	1.015
EnhanceAuthentication-Challenge-Response in IDE	1.254
EnhanceAnonymous-Challenge-Response in IDE	1.786

We use the simple identity-password authentication as the baseline, and compare the proposed schemes with this baseline. The reason for providing the latencies of the simple identity-password authentication in the two different running modes is because, even if we make the simple identity-password authentication from the IDE into the command-line version, our make version still cannot have the same performance as that provided by the HiveMQ team. We speculate that the command-line version provided by the HiveMQ has been further fine-tuned to optimize its performance. Therefore, we first compare the proposed schemes with the baseline to get the rough ratio of the comparison, and then further use the data to speculate the possible latencies if they have been commercially optimized.

Table 3 lists the latencies from the experiments. From the table, we can see that the same simple identity-password authentication scheme demands quite different authentication latencies in two different modes- the command mode and the IDE mode. The command mode requires only 0.153 seconds while the IDE mode requiring 1.015 seconds; that is, the IDE mode latency is almost 6.6 times the latency in the command mode for the simple identity-password authentication. We would like to evaluate the latency of the proposed schemes, compared to the simple identity-password scheme. In the IDE mode, the simple identity-password scheme demands 1.015 seconds, the classic challenge-response scheme requires 1.254 seconds, and the proposed anonymous challenge-response scheme takes 1.786 seconds. Compared to the simple identity-password scheme in the IDE mode, the classic challenge-response scheme demands only extra 239 ms, and the anonymous challenge-response scheme requires only extra 771 ms. Compared to the simple identity-password scheme, the proposed schemes demand only few more milliseconds, and it achieves much robust authentication security. If we use the scale factor 6.6 to estimate the latencies of the proposed schemes in the command mode, then they will be 0.19 seconds and 0.27 seconds respectively. Table 4. summarizes the security properties of the related schemes.

Table 4. Summary of related work

Properties Scheme	Target at MQTT 3.1 or 5.0	Goals & functions	Anonymity
[6-8]	3.1	Security weaknesses	No
[9]	3.1	TLS-embedded hardware	No
[10]	3.1	Key management	No
[11]	3.1	Policy management	No
[7, 10-20]	3.1	Customized key agreement	No
[21-22]	3.1	Customized key agreement	YES
Ours	5.0	Customized key agreement	YES

5 Security Analysis and Verification

This section examines the security of the proposed schemes. The proposed first scheme is a classic challenge-response authentication being implemented within the MQTT 5.0 Enhanced Authentication context; the session key of the first scheme is defined as K_{sess}: = $hash(K_i$, $C1$, $C2$, $device_{id}$, $topic)$, where the standard HMAC is suggested as the hash function. Therefore, its security is ensured as the classic challenge-response authentication and HMAC have been well studied and proved.

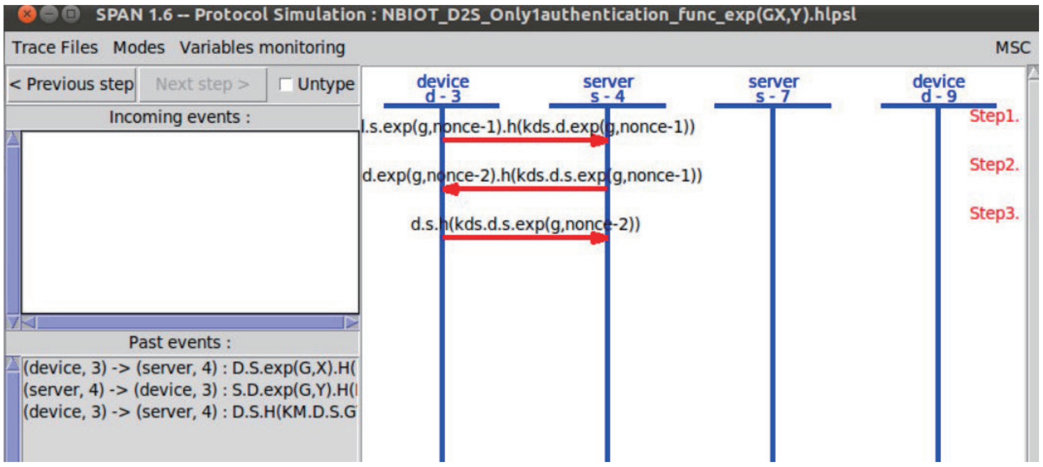
We therefore focus on verifying and analyzing the security of the second scheme. Here, we adopt two approaches. First, we specify the second scheme using the HLPSL specification/language, and then verify its *mutual authentication, session key privacy, and forward secrecy* using the AVISPA tool. Following that, we anlsyze the anonymity and the unlinkability because AVISPA tool cannot verify the two security properties.

We specify the second scheme using two HLPSL instances, where two roles- “server” for the broker and “device” for the client- are defined.

In the first specification, we specify one session in each role- server and device. The goal of this instance is to verify the properties of mutual authentication, session key privacy, and forward secrecy. Mutual authentication can be modeled using the predicate “authentication_on”, and session key privacy can be modeled using the predicate “secrecy_of”.

In the second specification, we model the forward secrecy property by specifying two successive sessions in both the server role and the device role in their respective HLPSL specifications to have two session keys. The specifications let the intruder have the secrets of the second session, and then verify whether the intruder can derive the secret keys of the first session. The goal is to verify whether, even if the intruder has the secrets of the 2nd session, the goals of authentication and session key privacy of the 1st session are still achieved in AVISPA verification. If the goals are satisfied, then it achieves the forward secrecy property. In AVISPA, the channel between the server and the device is modeled as the Dolev-Yao channels, where the intruder may divert sent messages and send new ones impersonating other agents. Figure

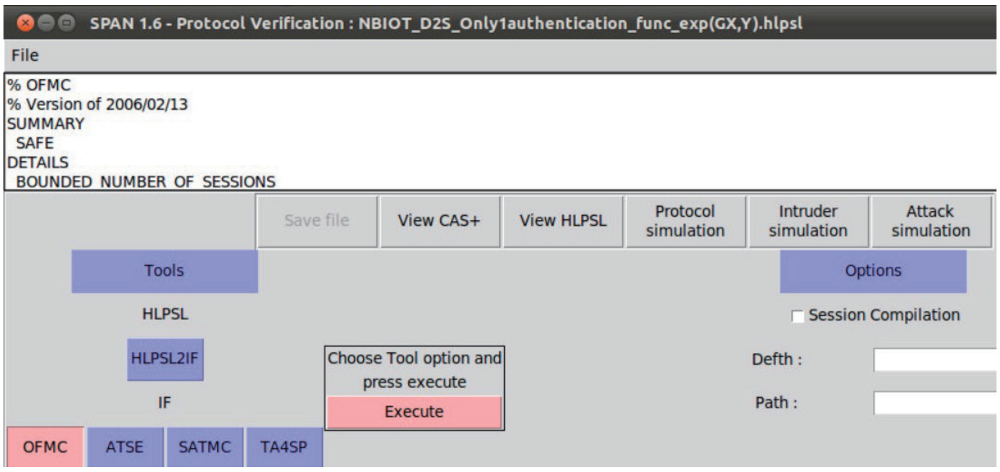
7 shows the message flows of the first specification where the device and the server achieve 3-step authentication process. Figure 8 shows the result of the OFMC verifier which is a formal security property verification tool in AVISPA; the OFMC reports "SAFE"—no successful attacks can be plotted within the protocol specification. It means the protocol achieves its goals of mutual authentication and session key privacy.



```
l.s.exp(g,nonce-1).h(kds.d.exp(g,nonce-1))
d.exp(g,nonce-2).h(kds.d.s.exp(g,nonce-1))
d.s.h(kds.d.s.exp(g,nonce-2))
```

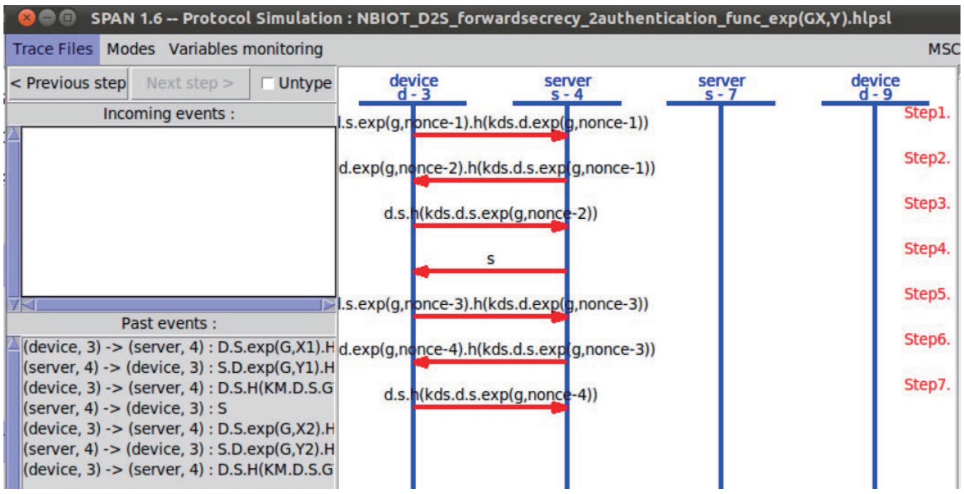
```
(device, 3) -> (server, 4) : D.S.exp(G,X).H(
(server, 4) -> (device, 3) : S.D.exp(G,Y).H(
(device, 3) -> (server, 4) : D.S.H(KM.D.S.G
```

Figure 7. The message flow of the 1st HLSPL specification of the 2nd scheme



```
% OFMC
% Version of 2006/02/13
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
```

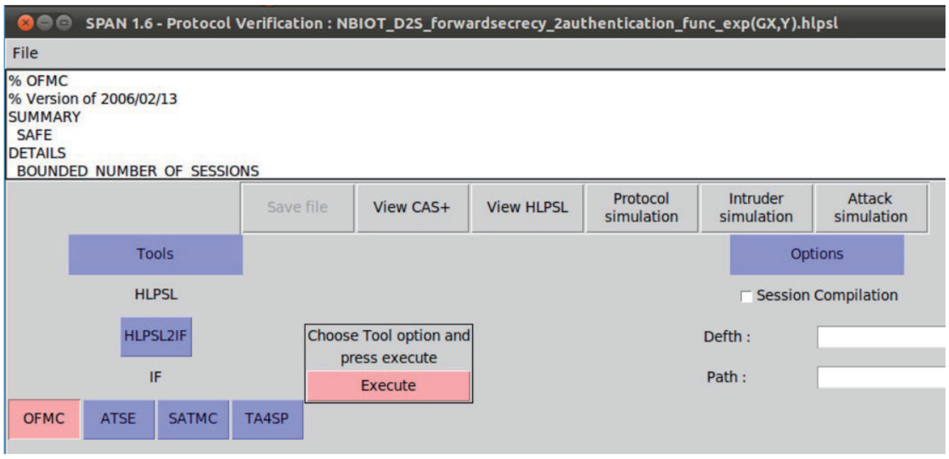
Figure 8. The result of On-the-Fly-Model-Checker (OFMC) on the 1st specification of the 2nd scheme



```
l.s.exp(g,nonce-1).h(kds.d.exp(g,nonce-1))
d.exp(g,nonce-2).h(kds.d.s.exp(g,nonce-1))
d.s.h(kds.d.s.exp(g,nonce-2))
s
l.s.exp(g,nonce-3).h(kds.d.exp(g,nonce-3))
d.exp(g,nonce-4).h(kds.d.s.exp(g,nonce-3))
d.s.h(kds.d.s.exp(g,nonce-4))
```

```
(device, 3) -> (server, 4) : D.S.exp(G,X1).H
(server, 4) -> (device, 3) : S.D.exp(G,Y1).H
(device, 3) -> (server, 4) : D.S.H(KM.D.S.G
(server, 4) -> (device, 3) : S
(device, 3) -> (server, 4) : D.S.exp(G,X2).H
(server, 4) -> (device, 3) : S.D.exp(G,Y2).H
(device, 3) -> (server, 4) : D.S.H(KM.D.S.G
```

Figure 9. The message flows of two successive authentications accomplished by the server and the device



```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
```

Figure 10. OFMC reports "SAFE" on the 2nd specification of the second scheme

Now we discuss the simulation of the 2nd specification which concerns the forward secrecy. Figure 9 shows the message flow, where "device d-3" and "server s-4" finish two runs of authentications, which means that the scheme can successfully complete two authentication sessions. Figure 10 shows that OFMC verifier could not plot any successful attacks on the 1st session, even if it lets the intruder gets the secrets of the 1st session. The result was "SAFE" in Figure 10. This means it achieves the forward secrecy property.

In addition to the above formal verifications of some security properties, we now analyze the security of the schemes. The first challenge-response authentication is a classic authentication scheme which has been well studied; we, therefore, focus on the security properties of the second anonymous authentication scheme in the following.

Mutual authentication between a client and the broker: The rationale of the scheme is basically a classic challenge-response authentication where the broker uses the pseudonym *PN* to identify a client and its corresponding secret key. The challenge from the client is the fresh *PN* and the $X = xP$, and the challenge from the broker is $Y = yP$. As long as the challenges are fresh and the secret key is secret, then the two entities can achieve mutual authentication through the challenge-response process.

Session key security and perfect forward secrecy: The session key is defined as $K_{sess} = h_1(\text{“}KEY\text{”}, ID_B, PN, K_i, xyP)$, where xyP is the computational Diffie-Hellman values derived from the two Diffie-Hellman values from the client and the broker. Therefore, the security of the session key is ensured owing to the secret key K_i, the forward secrecy is achieved owing to the Diffie-Hellman value xyP.

Anonymity of the clients: The second scheme always use the updated pseudonyms *PN* in the communications to achieve the anonymity.

Unlinkability among devices' successful connections: A client generates its pseudonym as $PN = h_2(N_i)$ and updates its seed $N_i = h_1(N_i)$, where h_1 and h_2 are two independent one-way hashing functions. The preimage resistance property of the one-way hashes and the freshness of the seed N_i ensures the unlinkability of the client.

Resistance to desynchronization-based DoS attacks: A client (say C_i) keeps its current pseudonym with the past *w1* pseudonyms while the broker keeping its recorded C_i's active pseudonyms and the future *w1* pseudonyms. This facilitates the two parties the capacity of coping with the situations of *w1* successive out-of-synchronization sessions, due to possible attacks or unreliable connections. The robustness of desynchronization-based-DoS-resistance depends on the window size *w1*; the choice of the size should consider the threat severity, the desirable robustness, and the storage space.

6 Conclusions

MQTT 5.0 defines the Enhanced Authentication Framework to support the design of customized authentication schemes. It is expected to greatly enhance the security supports for various scenarios. In this paper, we have proposed and implemented two efficient authentication schemes for MQTT 5.0. The first scheme is a classic challenge-response authentication within the MQTT 5.0 context. The second scheme is a hash-based anonymous authentication scheme which protects the client's identity privacy. The security of the proposed schemes has been verified using the formal verification tool AVISPA. The authentication latencies of the implementations have been evaluated. The results show that the implementations with much robust security only require, even in the IDE mode, few hundred milliseconds, compared to the simple identity-password authentication.

Acknowledgement

This project is partially supported by the Ministry of Science and Technology, Taiwan, R.O.C. grant number MOST 111-2221-E-260 -009 -MY3.

References

[1] ISO/IEC 20922:2016, *Information technology -- Message Queuing Telemetry Transport (MQTT) v3.1.1*, https://www.iso.org/standard/69466.html, Accessed 25

March 2022.

[2] OASIS, *OASIS Message Queuing Telemetry Transport (MQTT) TC*, https://www.oasis-open.org/committees/mqtt/, Accessed 7 March 2022.

[3] D. Locke, *MQ Telemetry Transport (MQTT) V3.1 Protocol Specification*, IBM DeveloperWorks Technical Library, August 2010. http://www.ibm.com/developerworks /webservices/library/ws-mqtt/index.html.

[4] Internet Engineering Task Force, *The Transport Layer Security (TLS) Protocol Version 1.3*, RFC 8446, August, 2018.

[5] Narens, *TLS and Mutual TLS handshake data overhead*, http://narendrasharma.blogspot.com/2018/01/tls-and-mutual-tls-handshake-data.html, Accessed 1 March 2022.

[6] Avast, *Avast research finds at least 32,000 smart homes and businesses at risk of leaking data*, https://press.avast.com/avast-research-finds-at-least-32000-smart-homes-and-businesses-at-risk-of-leaking-data, Accessed 7 March 2022.

[7] S. Andy, B. Rahardjo, B. Hanindhito, Attack Scenarios and Security Analysis of MQTT Communication Protocol in IoT System, *2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, Yogyakarta, Indonesia, 2017, pp. 1-5.

[8] S. N. Firdous, Z. Baig, C. Valli, A. Ibrahim, Modelling and Evaluation of Malicious Attacks against the IoT MQTT Protoco, *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber*, *Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Exeter, UK, 2017, pp. 748-755.

[9] C. Lesjak, D. Hein, M. Hofmann, M. Maritsch, A. Aldrian, P. Priller, T. Ebner, T. Ruprechter, G. Pregartner, *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, Cambridge, UK, 2015, pp. 1243-1250.

[10] A. Rizzardi, S. Sicari, D. Miorandi, A. Coen-Porisini, AUPS: An Open Source Authenticated Publish/Subscribe system for the Internet of Things, *Information Systems*, Vol. 62, pp. 29-41, December, 2016.

[11] R. Neisse, G. Steri, G. Baldini, Enforcement of security policy rules for the internet of things, *2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Larnaca, Cyprus, 2014, pp. 165-172.

[12] A. Niruntasukrat, C. Issariyapat, P. Pongpaibool, K. Meesublak, P. Aiumsupucgul, A. Panya, Authorization mechanism for MQTT-based Internet of Things, *2016 IEEE International Conference on Communications Workshops (ICC)*, Kuala Lumpur, Malaysia, 2016, pp. 290-295.

[13] S. H. Shin, K. Kobara, C. C. Chuang, W. C. Huang, A Security Framework for MQTT, *2016 IEEE Conference on Communications and Network Security (CNS): International Workshop on Cyber-Physical Systems Security (CPS-Sec)*, Philadelphia, PA, USA, 2016, pp. 432-436.

[14] S. H. Shin, K. Kobara, *Efficient Augmented Password-Only Authentication and Key Exchange for IKEv2*, IETF RFC 6628, Experimental, June, 2012.

[15] A. Bhawiyuga, M. Data, A. Warda, Architectural Design of Token based Authentication of MQTT Protocol in Constrained IoT Device, *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*, Lombok, Indonesia, 2017, pp. 1-4.

[16] H. Y. Chien, Y. J. Chen, G. H. Qiu, J. F. Liao, R. W. Hung, P. C. Lin, X. A. Kou, M. L. Chiang, C. H. Su, A MQTT-API-Compatible IoT Security-Enhanced Platform, *International Journal of Sensor Networks*, Vol. 32, No. 1, pp. 54-68, 2020.

[17] H.-Y. Chien, P. C. Lin, M. L. Chiang, Efficient MQTT Platform Facilitating Secure Group Communication, *Journal of Internet Technology*, Vol. 21, No. 7, pp. 1929-1940, December, 2020.

[18] H. Y. Chien, G. H. Qiu, R. W. Hung, A. T. Shih, C. H. Su, Hierarchical MQTT with Edge Computation, *the 10th International Conference on Awareness Science and Technology (iCAST 2019)*, Morioka, Japan, 2019, pp. 1-5.

[19] A. Mektoubi, H. Lalaoui, H. Belhadaoui, M. Rifi, A. Zakari, New approach for securing communication over MQTT protocol A comparison between RSA and Elliptic Curve, *2016 Third International Conference on Systems of Collaboration (SysCo)*, Casablanca, Morocco, 2016, pp. 1-6.

[20] M. Singh, M. A. Rajan, V. L. Shivraj, P. Balamuralidhar, Secure mqtt for internet of things (iot), *2015 Fifth International Conference on Communication Systems and Network Technologies*, Gwalior, India, 2015, pp. 746-751.

[21] H. Y. Chien, Highly Efficient Anonymous IoT Authentication Using Composite Hashing, *The 2021 IEEE Conference on Dependable and Secure Computing*, Aizuwakamatsu, Fukushima, Japan, 2021, pp. 1-7.

[22] H. Y. Chien, Two-Level-Composite-Hashing Facilitating Highly Efficient Anonymous IoT and D2D Authentication, *MDPI Electronics*, Vol. 10, No. 7, Article No. 789, April, 2021.

[23] OASIS, *MQTT Version 5.0*, 07 March 2019. https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[24] OASIS, *MQTT 5 Specification*, https://mqtt.org/mqtt-specification/.

[25] G. Held, *Enhanced Authentication*, HiveMQ, 6 March 2020, https://www.hivemq.com/blog/mqtt5-essentials-part11-enhanced-authentication/.

[26] Mosca, https://github.com/mcollina/mosca/, Accessed 7 November 2018.

[27] M. Bellare, R. Canetti, H. Krawczyk, Keying Hash Functions for Message Authentication, in: N. Koblitz (Eds.), *Advances in Cryptolo-gy—CRYPTO '96*, Springer, Berlin, Heidelberg, 1996, pp. 1-15.

[28] D. V. Oheimb, The high-level protocol Specification language HLPSL developed in the EU project AVISPA, *Proceedings of the APPSEM 2005 Workshop*, Frauenchiemsee, Germany, 2005, pp. 1-17.

[29] Avispa, *A tool for Automated Validation of Internet Security Protocols*, http://www.avispa-project.org, Accessed 20 February 2020.

Biographies

Hung-Yu Chien received the B.S. degree from NCTU, Taiwan, 1988, the M.S. degree from NTU, Taiwan, 1990, and the doctoral degree in applied mathematics at NCHU 2002. He is a professor of National Chi Nan University since 199808 His research interests include cryptography, networking, network security, ontology, and Internet-of-Things.

Pin-Ping Ciou received her bachelor degree in the Department of Information Management at the National Chi Nan University in Taiwan. Her research topic is Message Queuing Telemetry Transport and network security.