Continuing to reflect on TLS 1.3 with external PSK

Liliya Akhmetzyanova[*], Evgeny Alekseev[†],
Ekaterina Smyshlyaeva[‡], Alexandr Sokolov[§]
Crypto Pro LLC

Abstract

The TLS protocol is the main cryptographic protocol of the Internet. The work on its current version, TLS 1.3, was completed in 2018. This version differs from the previous ones and has been developed taking into account all modern principles of constructing cryptographic protocols. At the same time, even when there are security proofs in some fairly strong security model, it is important to study the possibility of extending this model and then clarifying the security limits of the protocol.

In this paper, we consider in detail the restriction on the usage of post-handshake authentication in connections established on external PSK. We clarify that the certain vulnerability appears only in the case of *psk_ke* mode if more than a single pair of entities can possess a single PSK. We provide several practical scenarios where this condition can be easily achieved. Also we propose appropriate mitigation.

1 Introduction

The current version of the TLS protocol, the main cryptographic protocol of the Internet, is described in the document [9], which was completed in 2018. This protocol was created taking into account all known vulnerabilities of previous versions and the protocol design principles that allow to obtain proven security bounds. This came out, for example, in the structure of the key schedule: «The key derivation functions have been redesigned. The new design allows easier analysis by cryptographers due to their improved key separation properties» [9]. As a result, several reductions and, consequently, lower security bounds were obtained for the TLS 1.3 protocol (see, for example, [2, 5, 8]) in various security models.

As noted in [4], in addition to constructing reductions and thus obtaining lower security bounds, it is important to consider the adequacy and completeness of the used security model. Examples of attacks against «provably» secure cryptosystems, which were possible due to the irrelevance of the used security model, can be found, for example, in [1, 3]. For the TLS 1.3 protocol, the issue on analyzing the model relevance is extremely difficult, since this protocol provides a number of

[*] lah@cryptopro.ru
[†] alekseev@cryptopro.ru
[‡] ess@cryptopro.ru
[§] sokolov@cryptopro.ru

non-trivial operation modes (for example, authentication with a pre-shared secret or sending data to the unauthenticated party), which can affect each other in a rather complicated way.

Despite the fact that the TLS 1.3 protocol was designed with all modern cryptographic principles taken into account, a new attack has been already proposed. The paper [6] describes a reflection attack called «Selfie» on TLS 1.3 based on a pre-shared key (PSK) distributed externally. The idea behind this attack is as follows: an adversary opens two parallel connections where the same entity runs both a client and a server, and reflects messages from one connection back to the sender in another connection. As a result, the entity opens a session with itself. This attack is applied if the entities sharing external PSK can be both a client (an initiator) and a server (a responder) simultaneously and the parties do not explicitly check the identity of the partners. The paper also provides the several attack scenarios which illustrate the posed threats.

One mitigation of the Selfie attack proposed in [6] is to use external PSKs together with certificates. Therefore, a built-in post-handshake authentication can be potentially used for protection. Indeed, the entity that has made a session with itself can detect this by receiving its own certificate in the server side connection.

However, the situation with permission of using post-handshake authentication with external PSKs in [9] is quite unclear. On the one hand, [9] allows to use post-handshake authentication, if the client indicated the corresponding extension (see Section 4.3.2 [9] for details):

> `Servers which are authenticating with a PSK MUST NOT send the Certificate Request message in the main handshake, though they MAY send it in post-handshake authentication provided that the client has sent the post_handshake_auth extension.`

On the other hand, in [9] (see Appendix E, p.144) one can find the following statement:

> `Implementations MUST NOT combine external PSKs with certificate-based authentication of either the client or the server unless negotiated by some extension.`

The latter statement is explained in [9] in the following way: «It is unsafe to use certificate-based client authentication when the client might potentially share the same PSK/key-id pair with two different endpoints», — and refers to [8] for more details. However, in [8] the only mentioning about this threat is the statement (see the footnote on p.13): «We exclude the case where the attacker has the same PSK key shared with both C and S in which case a transcript replication attack is unavoidable.»

To clarify the situation described above we investigate the post-handshake authentication with external PSKs more precisely. We provide the explicit description of an impersonation attack caused by the vulnerability mentioned in [8]. This attack is applied if more than a single pair of entities can possess a single PSK. Although this condition is quite strong and seems to be impractical, we show that it can be easily achieved on practice, providing two dummy systems where external PSKs are used (see Section 4 for details) .

In the first system the main reason for this vulnerability is the absence of control on how the external PSK was generated. Indeed, if there is no specific mechanism of distributing PSKs, an «honest» client cannot detect whether a server provides a pre-shared key generated uniformly and independently or obtained from another entity. In the second system being the group chats prototype, the entities legally possess a single external PSK which aims to provide authentication of a group membership. The practicality of distributing an external PSK to a group of entities is also mentioned in [7].

The described attack is applied only to the *psk_ke* mode and is not applied to the *psk_dhe_ke* mode. Therefore, the restriction on the usage of external PSKs with post-handshake authentication stated in [9] seems to be fair for one mode only. Consequently, we claim that the post-handshake authentication allows to securely protect from the Selfie attack for the *psk_dhe_ke* mode only (without leading to new vulnerabilities).

The paper is organized as follows. In Section 2 we remind the PSK based key exchange modes of TLS 1.3 and the post-handshake authentication mechanism. In Section 3 we describe an impersonation attack. Section 4 discusses the implications of the attack on practice and in Section 5 the mitigations are discussed.

2 TLS 1.3 protocol

TLS 1.3 allows to establish a secure connection between client and server using PSK which is known to the parties before the current connection and is obtained as follows:

1. Distributed between the parties out of the protocol TLS 1.3: external PSK.
2. Obtained in the previous connection via the NewSessionTicket mechanism: resumption PSK.

Further we describe `psk_ke` and `psk_dhe_ke` modes where external PSKs are used and post-handshake authentication.

2.1 Modes based on PSK

Notation. For simplicity, client and server handshake and application traffic secrets are denoted by $CHTS$, $SHTS$ and $CATS$, $SATS$ correspondingly. The negotiated hash is denoted by $Hash$. The concatenation of all messages sent during the main handshake is denoted by HM.

`psk_ke` and `psk_dhe_ke` modes. In order to establish a connection based on PSK the client initiates the communication by sending $ClientHello$ message where it indicates:

- a list of supported cipher suites (block cipher, AEAD and Hash algorithms);
- `pre_shared_key` extension which contains the data corresponding to each PSK and formed as described below;
- `psk_key_exchange_modes` extension which contains the value *psk_ke* and/or *psk_dhe_ke*.

In the case of the *psk_dhe_ke* mode, the *ClientHello* message must also contain the following extensions:

- `supported_groups` extension which contains a list of DH/ECDH groups supported by a client;
- `key_share` extension containing the DH/ECDH ephemeral keys corresponding to each group indicated in `supported_groups` extension.

The `pre_shared_key` extension contains a list of identities and a list of binders which correspond to each PSK proposed by a client. Every identity contains a unique identifier and `obfuscated_ticket_age`, which is equal to zero in case of external PSKs.

Server sends the *ServerHello* message in response, where it indicates a specific ciphersuite, a specific PSK from the list and (in the case of the *psk_dhe_ke* mode) an ephemeral key corresponding to a group from the client's list in `supported_groups` extension. Then it sends *EncryptedExtensions* and *Finished* messages, which are encrypted under the key material derived from $SHTS$. The client verifies the data received in server *Finished* message and sends its own *Finished* message, which is encrypted under the key material derived from $CHTS$ and is verified by the server.

2.2 Post-Handshake Authentication

If client desires to authenticate explicitly after the main handshake, it can optionally indicate the `post_handshake_auth` extension in *ClientHello* message. In that case, server sends *CertificateRequest* message encrypted under the key material derived from $SATS$. Client sends in response $Certificate$, $CertificateVerify$ and *Finished* messages encrypted under the key material derived from $CATS$. Server receives these messages from the client, verifies the data in client $CertificateVerify$ and *Finished* messages and then considers the client to be authenticated.

Figure 1 represents `psk_ke` and `psk_dhe_ke` modes, based on external PSKs and described in 2.1, and post-handshake authentication.

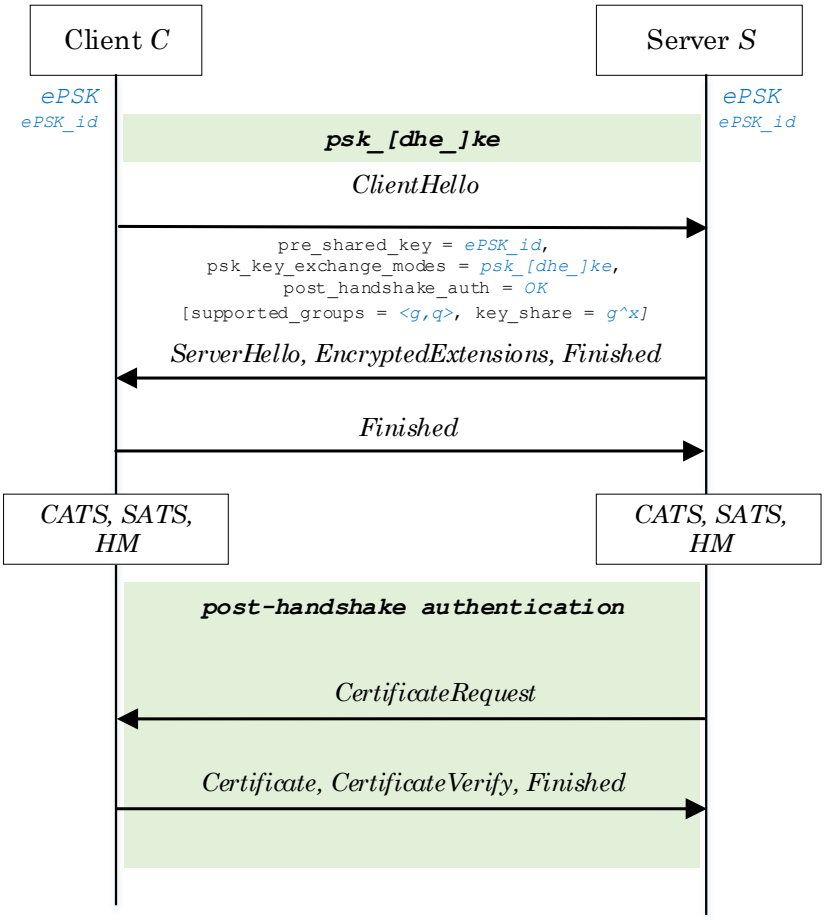

Figure 1: Modes based on external PSK with post-handshake authentication

3 Impersonation attack

Consider an adversary E and two honest parties C and S that always run as a client and as a server correspondingly. Let the adversary be a legitimate server for the client C and be a legitimate client for the server S. Also let C be a legitimate client for S and use the same certificate for authentication to S and E.

We assume that the connections between C and E and between E and S are established according to the TLS 1.3 protocol in the *psk_ke* mode with the usage of externally distributed PSK which is chosen by the server. Also we assume that after the connection establishment the server side can additionally request the post-handshake authentication with the usage of certificates.

The threat considered here is the impersonation of the client C by the adversary E for the server S.

The attack. Further by (A, B) we denote a connection established between parties A and B, where A is a client and B is a server. In order to impersonate C, the adversary acts as follows:

1. The server S externally shares with its client E (the adversary) the PSK value. Having obtained this value, the adversary now shares the same PSK with its own client C.

2. The client C initiates the interaction with E in order to establish the connection in the `psk_ke` mode using the previously obtained PSK. Also, the client C allows the server to implement the post-handshake authentication by sending a `post_handshake_auth` extension in the $ClientHello$ message. At the same time, the adversary E initiates the connection establishment with S in the same psk_ke mode using the same PSK.

3. The connections (C, E) and (E, S) are established simultaneously, where E just echoes messages from C to S and vice versa. As a result, these connections will be established successfully and the key application traffic secrets $CATS$, $SATS$ and the protocol transcript will be identical in both connections.

4. Further the server S requests the authentication of E by sending the encrypted $CertificateRequest$ messages in the connection (E, S). Receiving this message, the adversary sends the same message to its own client C in the connection (C, E).

5. In response the client C sends messages $Certificate$, $CertificateVerify$ and $Finished$ to E which echoes these messages to S in its connection. Note that the obtained authentication messages are verified successfully on the server S side, and, consequently, the S view of peer's identity is C.

The idea behind the attack is presented in Figure 2.

Consider why the proposed attack works in more details. Consider the data that is signed by the client C when forming the $CertificateVerify$ message in the connection (C, E). It depends on the whole transcript of the main handshake (HM) and the messages $CertificateRequest$ and $Certificate$. Since the adversary just echoes the messages from C to S and vice versa, the signed transcript-hash in both connections will be the same. Therefore, the $CertificateVerify$ message remains valid for S too.

Remark 3.1. Note that in the case when an external PSK is replaced by a resumption PSK in one of the connections, the proposed attack does not work due to the usage of the different labels `«ext_binder»` and `«res_binder»` for computing a *binder key*. If the same labels are used, the attack will be still applied since the `obfuscated_ticket_age = 0` field of the `pre_shared_key` extension will be valid for the side waiting resumption PSK with a high probability.

4 Implications

Consider the following scenarios to demonstrate the damage the described attack poses.

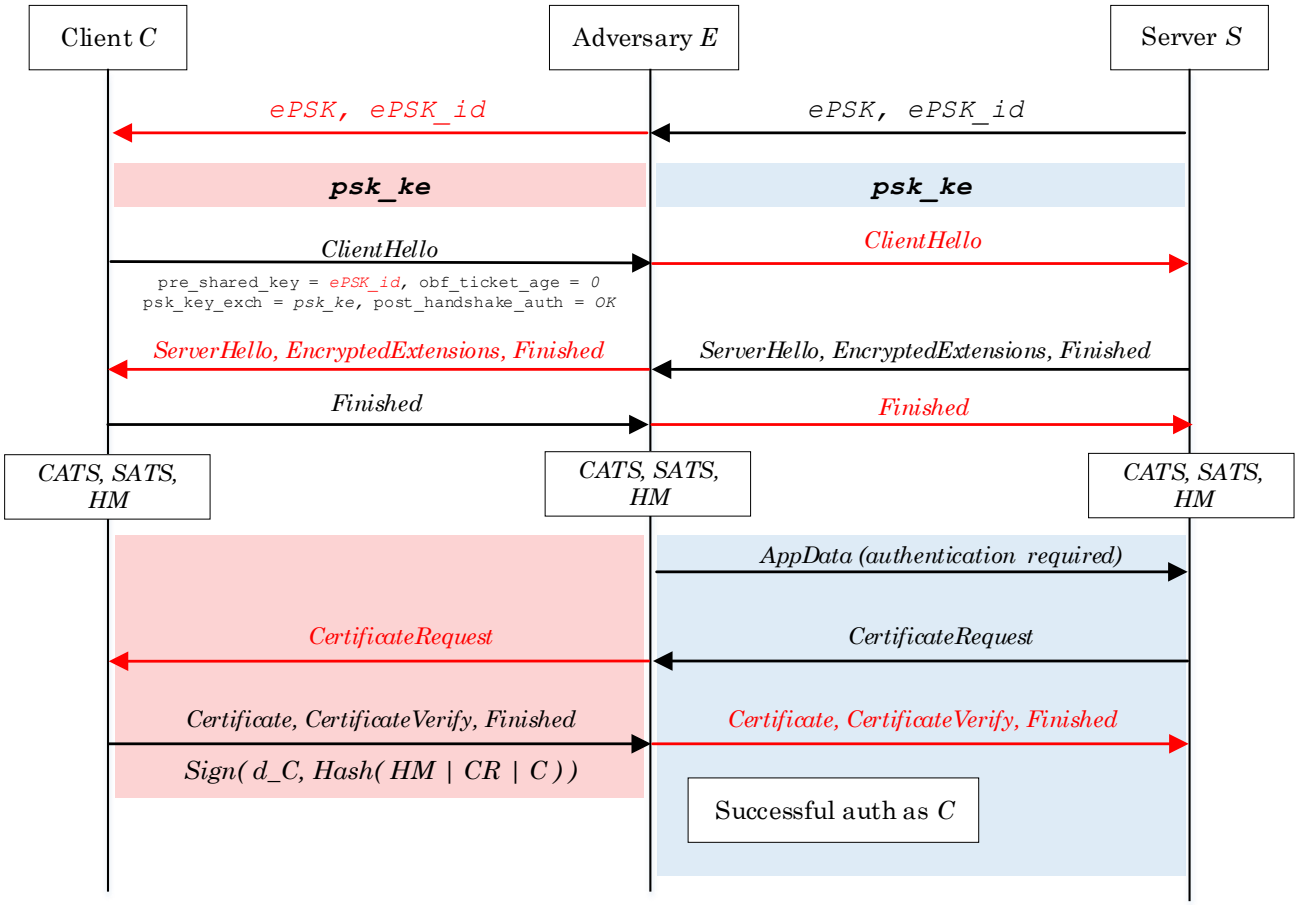

Figure 2: The adversary E impersonates the client C for the server S.

Internet of Things. Consider the system where low-power devices (for example, IoT-type devices) act as clients. In order to quickly establish a connection using the TLS 1.3 protocol, secret values PSK are used, which are hardcoded into the device memory at the system initialization stage. Here, PSK is aimed to provide authentication of a central server and authentication of client membership to the system. To perform rare system specific operations, requiring a high security level, the server can request resource-intensive client authentication with the usage of a certificate after successfully establishing the connection on PSK.

Note that a device owner can potentially use a signing key and a corresponding certificate which have been received from some third party (e.g., using CA) and are aimed to be used for different purposes, e.g. for authentication in other more reliable systems (for example, providing banking services).

Thus, a malicious server of some IoT system can share (with a device) a PSK secret that has been distributed in another system between its honest server and some malicious agent being its

client. Consequently, using the described attack, the malicious IoT server can impersonate the device owner, e.g, in a banking system.

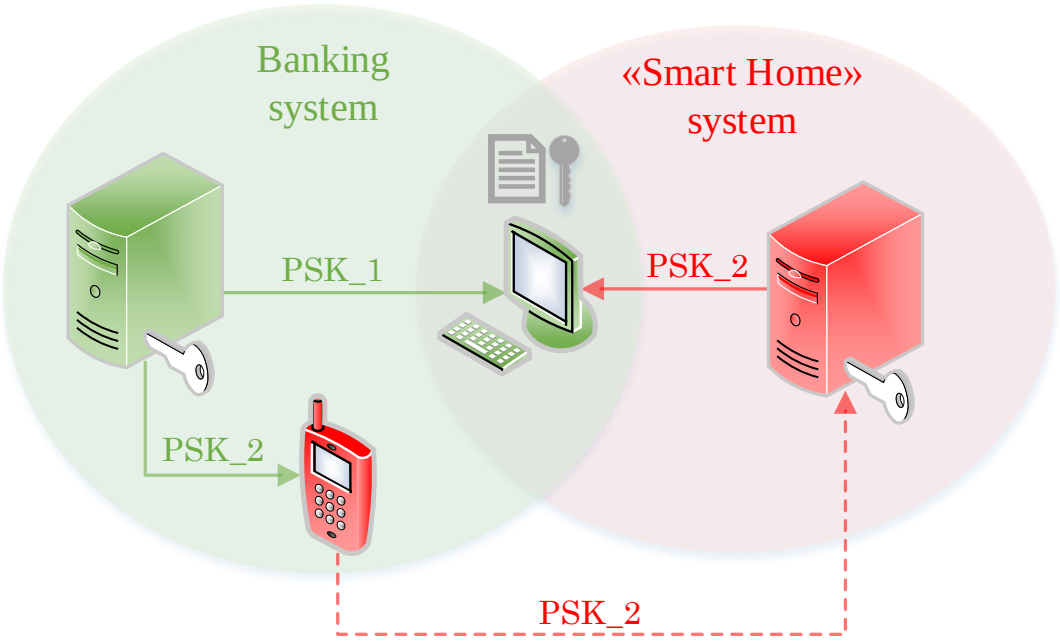

Figure 3: The distribution of external PSKs leading to the impersonation attack. Here the malicious server of IoT system has its own malicious agent being a legitimate client in a banking system.

Group chats. Consider the group chats where the membership to the group is confirmed by the possession of PSK distributed externally among the group participants. In order to communicate, entities establish peer-to-peer connections according to the TLS 1.3 protocol in the *psk_ke* mode. In these connections PSK values respond for authentication of each other's membership to the group. In order to identify and authenticate the exact member the post-handshake authentication (if needed) is used. Thus, using the described attack, one malicious member of the group can impersonate any «honest» member (see Figure 4).

Remark 4.1. Note that the current OpenSSL implementation of TLS 1.3 supports the usage of external PSKs in the *psk_ke* mode (see the flag `SSL_OP_ALLOW_NO_DHE_KEX`) with post-handshake authentication.

5 Mitigations

As posed in [6], the certificates should be used to mitigate the Selfie attack. To protect also against the impersonation attack, the connections based on external PSK should be established

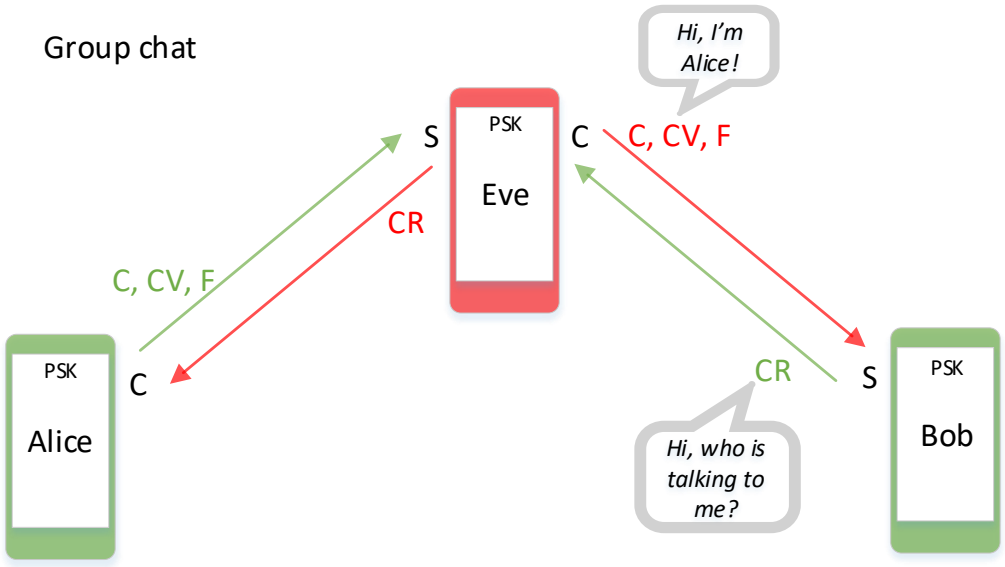

Figure 4: The demonstration of the attack for group chats.

only in the *psk_dhe_ke* mode. Indeed, if the connections are established with additional usage of `key_share` extension, then the adversary which just echoes messages cannot reveal the traffic key material (as long as the used group is secure). If the adversary changes the messages then the transcript-hash value will be different on the «honest» sides and the signature verification will fail. Thus, the following restriction can mitigate both attacks: «External PSKs MUST be used together with certificates in `psk_dhe_ke` mode only».

Note that the proposed mitigation provides certificate-based authentication of the client only. If such an authentication of the server is also needed (e.g. in group chats) the `tls_cert_with_psk` extension proposed in [7] can be used.

In the future work we are going to investigate an enhanced MSKE model [6] where more than a pair of parties can possess the same PSK and then analyze the certificate-based authentication for such a network configuration by obtaining a security proof.

Acknowledgements. We thank Dmitry Belyavsky, Vasily Nikolaev, and Stanislav Smyshlyaev for useful discussions and comments during this work.

References

[1] Albrecht M.R., Paterson K.G., Watson G.J.: *Plaintext recovery attacks against SSH*. In 2009 IEEE Symposium on Security and Privacy, pp. 16–26 (2009).

[2] Bhargavan K., Delignat-Lavaud A., Fournet C., Kohlweiss M., Pan J., Protzenko J., Rastogi A., Swamy N., Zanella-Beguelin S., Zinzindohoue J.K., «Implementing and Proving the TLS 1.3 Record Layer». Cryptology ePrint Archive, Report 2016/1178, 2016.

[3] Canvel B., Hiltgen A., Vaudenay S., Vuagnoux M.: *Password Interception in a SSL/TLS Channel*. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 583–599 (2003).

[4] Degabriele J.P., Paterson K.G., Watson G.J.: *Provable security in the real world*. IEEE Security & Privacy 9(3), pp. 33–41 (2011).

[5] Dowling B.J., «Provable security of Internet protocols». Doctoral Thesis, Queensland University of Technology, `https://eprints.qut.edu.au/108960/`, 2017.

[6] Drucker N, and Gueron S., «Selfie: reflections on TLS 1.3 with PSK». Cryptology ePrint Archive, Report 2019/347, 2019.

[7] Housley R., «TLS 1.3 Extension for Certificate-based Authentication with an External Pre-Shared Key», version 02, September 26, 2018.

[8] Krawczyk H., «A Unilateral-to-Mutual Authentication Compiler for Key Exchange (with Applications to Client Authentication in TLS 1.3)». Cryptology ePrint Archive, Report 2016/711, 2016.

[9] Rescorla E., «The Transport Layer Security (TLS) Protocol Version 1.3», RFC 8446, DOI 10.17487/RFC8446, August 2018, `https://www.rfc-editor.org/info/rfc8446`.