TLS Working Group
Internet-Draft
Intended status: Informational
Expires: August 4, 2018

T. Putman
Dyson Technology

January 31, 2018

Authenticated Key Agreement using Pre-Shared Asymmetric Keypairs for (Datagram) Transport Layer Security ((D)TLS) Protocol version 1.3 draft-putman-tls13-preshared-dh-00

Abstract

This document defines an authenticated key agreement method for the Transport Layer Security (TLS) protocol version 1.3. The authentication method requires that the server (and optionally client) is pre-provisioned with a unique long-term static asymmetric Finite Field Diffie-Hellman (DH) or Elliptic Curve Diffie-Hellman (ECDH) keypair; the peer must be able to obtain the public key of the endpoint via an out-of-band mechanism (e.g. pre-provisioning). The handshake provides ephemeral (EC)DH keys, and a common key schedule is agreed using Double- or Triple-(EC)DH. Confirmation of knowledge of the key schedule provides server (and optionally client) authentication.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 4, 2018.

Table of Contents

1. Introduction

Often, constrained devices use pre-shared keys (PSK) for authentication and key agreement. A drawback of this is that if the server database of pre-shared keys is compromised, then this means that not only can the server be impersonated to the clients, but also the symmetric nature of the keys means that the clients can be impersonated to the server.

In consequence, a large-scale database compromise can result in large-scale client impersonation. This is very hard to recover from

because any remote update to the clients risks providing the updated information to an adversary.

This document describes the use of asymmetric pre-shared keys to address this data-loss scenario. It assumes that an out-of-band method of pre-configuring the asymmetric keys into the endpoints exists, and this method is outside the scope of this document. The primary intended usage is for constrained devices, but the method may be used in other circumstances where the peer's public key may be securely obtained out-of-band (e.g. via DNSSEC).

Another advantage of asymmetric pre-shared keys over symmetric pre-shared keys is that it is easier to protect a single server private key using hardware-based security than it is to protect a database of shared symmetric keys.

1.1. Rationale for Using Authenticated Key Agreement

In [I-D.ietf-tls-tls13], all key exchange methods except those based on PSK require perfect forward secrecy (PFS), which in turn requires (at present) either Diffie-Hellman or Elliptic Curve Diffie-Hellman. The number of constrained devices which can be accommodated is increased if the key agreement also provides authentication, so that no other public key algorithms are required.

Even if the device contains code for other public key algorithms (e.g. EdDSA for code update signature checking), these may be coded to use a slow variant of the algorithm to conserve code and data space. In addition, hardware support for ECDH is likely to be introduced in many constrained devices, as it is required for wireless communications (e.g. Bluetooth Mesh), whereas signing algorithms are not.

Double-(EC)DH [Blake-Wilson] could be used for authenticated key agreement, but this would not provide PFS. PFS can be provided by using Triple-(EC)DH [Kudla] with no change to the protocol messages; it only adds to the cost of computing the keying material by adding one additional (EC)DH computation.

In order to break forward secrecy in Double-(EC)DH, the attacker must obtain the static (EC)DH private keys of both client and server. This is likely to be a difficult feat, so both Double- and Triple-(EC)DH are specified in this document as this increases the number of constrained devices which are able to use this method.

Perfect Forward Secrecy (PFS) is a strongly recommended feature in security protocol design and is mandatory to implement in both HTTP/2 [RFC7540] and (for non-PSK deployments) CoAP [RFC7252]. Therefore,

Triple-(EC)DH SHOULD be used for those deployments where the client devices are able to support the additional computation.

1.2. Applicability Statement

The authentication methods defined in this document are primarily intended for a narrow set of applications, where there is a well-established relationship between clients and servers and where, in addition, there are severe constraints on the client capabilities. Even in such deployments, other alternatives may be more appropriate.

If the loss of server data is not of concern, then the use of symmetric pre-shared keys may be more appealing. If, on the other hand, the main goal is to avoid Public-Key Infrastructures (PKIs), then the use of raw public keys [RFC7250] may be preferrable.

A secondary use of this authentication method is to support 0-RTT traffic in a situation where no (current) session tickets exist. The server in this situation will typically not have knowledge of the client, so a method which supports only server authentication is also supported.

1.3. Comparison of 2/3(EC)DH with Other Methods

The 2/3(EC)DH authenticated key agreement modes offer advantages over other schemes, but there are limitations as well. A summary of advantages and disadvantages are given in the following table:

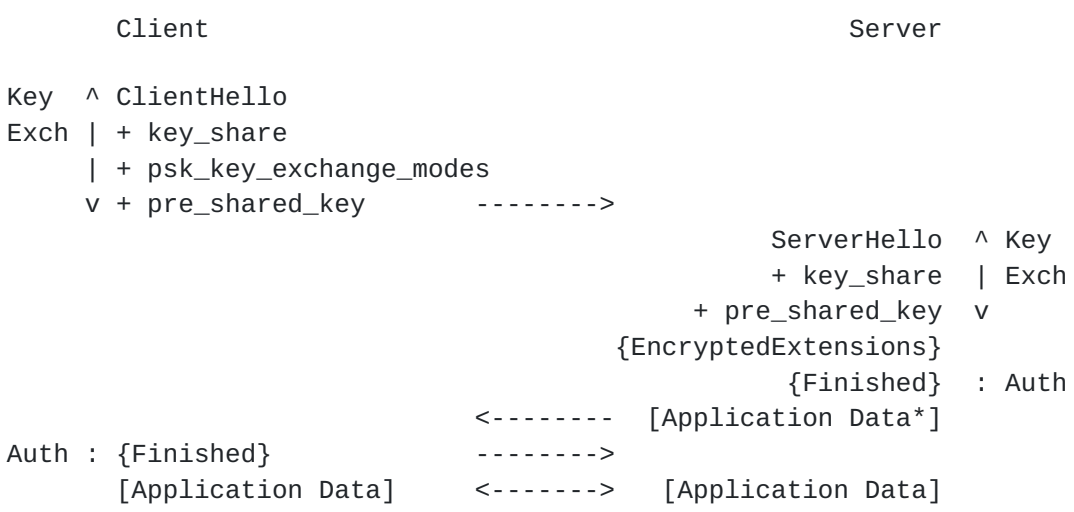
Alternative Authentication Mode	2/3(EC)DH Advantage	2/3(EC)DH Disadvantage
External PSK	Server breach does not permit client impersonation; hardware protection for server key possible; client identity is confidential	Public-key computations needed
Raw Public Keys	Supports 0-RTT messages; only one public-key algorithm used; shorter messages	Separate server keypair needed if it also supports PKI
Certificate Authentication	Supports 0-RTT messages; only one public-key algorithm used; no certificate parsing; much shorter messages	Out-of-band public key distribution needed (e.g. pre-provisioning, DNSSEC)

1.4. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

2. 2(EC)DH and 3(EC)DH Key Exchange

This section defines the key exchange mechanisms which are used for 2(EC)DH and 3(EC)DH. The message exchange is identical to that used for PSK with (EC)DHE key establishment, and is reproduced below in Figure 1. Although the message exchange is identical, the pre-shared keys have extra semantics applied and the key schedule is different.



+ Indicates noteworthy extensions sent in the previously noted message.

* Indicates optional or situation-dependent messages/extensions that are not always sent.

{} Indicates messages protected using keys derived from a [sender]_handshake_traffic_secret.

[] Indicates messages protected using keys derived from [sender]_application_traffic_secret_N

Figure 1: Message flow for 2/3(EC)DH TLS Handshake

Details of the PSK identity format which is used with 2/3{EC}DH is given in section Section 3. To understand the key exchange explained here, it only necessary to know that the PSK identity is the concatenation of a server PSK identity with an optional encrypted client PSK identity. The server PSK identity is uniquely associated with an (EC)DH private/public keypair, and the client PSK identity is present if and only if a key share on the named curve corresponding to the server PSK identity is present in the ClientHello message.

The client indicates its willingness to use 2/3(EC)DH authenticated key exchange by including at least one compatible identity in its "pre_shared_key" extension. Support for 2(EC)DH versus 3(EC)DH is indicated by including the values psk_ke or psk_dhe_ke respectively in the "psk_key_exchange_modes" extension; as with other pre-shared key methods, psk_ke does not support PFS, whereas psk_dhe_ke does. The client MUST NOT include more than one identity which is associated with the same server PSK identity: doing so would compromise the security of the encrypted client PSK identity. The client MUST also include in the "supported_groups" extension all

(EC)DH groups which are needed to establish a key exchange for any of the provided PSK identities: 2/3(EC)DH key exchange requires that all static and ephemeral (EC)DH keys are in the same named group.

If the TLS server accepts the use of 2(EC)DH or 3(EC)DH, then it selects a PSK identity which corresponds to the chosen key exchange mode from the list of offered identities in the "pre_shared_key" extension of the ClientHello message. As with other PSK modes, the server MUST ensure that the selected PSK and cipher suite are compatible, and it may do this by selecting the cipher suite first then excluding all incompatible PSKs.

Prior to accepting PSK key establishment, the server MUST validate the corresponding binder value (see Section 4.2.11.2 of [I-D.ietf-tls-tls13]). If this value is not present or does not validate, the server MUST abort the handshake. Servers SHOULD NOT attempt to validate multiple binders; rather they SHOULD select a single PSK and validate solely the binder that corresponds to that PSK.

The server MUST support 3(EC)DH on each of its server PSK identities, and may support 2(EC)DH on any subset of identities (including all or none). The client MAY be willing to establish key exchange either with or without PFS, indicated by including both the psk_ke and psk_dhe_ke values in the "psk_key_exchange_mode" extension. If, in this situation, the server chooses to use a 2/3(EC)DH identity, then it SHALL select the 3(EC)DH key exchange method. The client MUST support this behavior by constructing the corresponding binder using the 3(EC)DH binding key; if it does not, then the handshake will fail.

The selected PSK identity identifies a server static (EC)DH private/public keypair and implicitly also a named group. If the ClientHello message does not contain a key share for this group, then the server SHALL request the key share by sending a HelloRetryRequest which indicates the selected PSK identity in the "pre_shared_key" extension. If the ClientHello message does contain a key share for this group but the selected identity does not contain an encrypted client identity, then the server SHALL abort the handshake with an "illegal_parameter" alert.

A client SHOULD remember which server PSK identity a server preferred in previous sessions and SHOULD include the corresponding key share in the first ClientHello message when establishing future sessions. This will reduce the number of HelloRetryRequest messages from the server and will speed up session establishment.

If the ClientHello is sent in response to a HelloRetryRequest, then it MUST only contain the PSK identity selected by the server. It MUST also contain in the "key_share" extension an ephemeral (EC)DH key which is on the named curve associated with the server PSK identity. Consequently, the PSK identity will contain both the server PSK identity and the encrypted client PSK identity.

If 2/3(EC)DH PSK key exchange mode is selected, then the server decrypts the client identity and, if it is not an anonymous client, obtains the corresponding client static public (EC)DH key. If it accepts anonymous clients or if it accepts the identified client, then the server responds with a ServerHello message; this message MUST include a "pre_shared_key" extension which indicates the selected PSK identity and an ephemeral key share in the same named group as the selected static (EC)DH key. It SHALL NOT send Certificate or CertificateRequest messages.

Both peers MUST use the selected ephemeral and static (EC)DH keys to derive the key schedule using either 2(EC)DH or 3(EC)DH, depending on the selected key exchange mode, as specified in section Section 5.1.

If the 2/3(EC)DH PSK key exchange mode is selected and the decrypted client identity is not recognised, or if it indicates an anonymous client and the server is configured to reject anonymous clients, then the server SHALL select a valid unused public key to use for the purposes of computing the key binding, and only reject the handshake when the binding check fails. This is needed to prevent an oracle attack on client identity confidentiality where the attacker replays a ClientHello but modifies the encrypted client identity in a controlled way.

3. Format of 2/3(EC)DH Identity

The format of a PSK identity defined in [I-D.ietf-tls-tls13] is opaque<1..2^16-1>, but additional structure is needed to support confidentiality of the client PSK identity; this structure remains opaque to the protocol, but must be agreed between the end-points using an out-of-band mechanism.

An identity which is associated with 2/3(EC)DH key exchange SHALL be the concatenation of a server PSK identity with an optional encrypted client PSK identity.

A server which supports 2(EC)DH and/or 3(EC)DH MUST have one or more identities associated with it for the purposes of (EC)DH PSK key exchange. Each identity is associated with a single static (EC)DH private/public keypair and Hash algorithm; these identities are

provided to the clients via an out-of-band mechanism (e.g. pre-provisioning) which is outside the scope of this specification.

To simplify processing, the server PSK identities SHOULD all be the same size; if the server has a single PSK identity, then it MAY be NULL (zero length), but this is NOT RECOMMENDED as this complicates the addition of new identities. Also, the server SHOULD ensure that PSK identities which are established by the session ticket mechanism are always distinguishable from 2/3(EC)DH PSK identities; for example, if the session ticket is constructed as described in section 4 of [RFC5077], then the opaque "key_name" should be chosen so that it does not collide with any of the server PSK identities.

The encrypted client PSK identity MUST be present if the corresponding key share is present in the "key_share" extension, and MUST NOT be present otherwise. If the server has a NULL identity, then the minimum length of the PSK identity requires that the client PSK identity must be present; this in turn means that the corresponding key share must be present in the "key_share" extension.

A single server MAY support multiple static (EC)DH keys across one or more named curves by having multiple server PSK identities. This allows rotation of static keys on a single curve, or migration from a weaker named curve or hash algorithm to a stronger one. The policies required to rotate keys or change curves/hash algorithms are outside the scope of this specification, but if such a change is instigated then the overlap period during which both keys are supported should be sufficiently long to ensure that all clients have been updated.

As suggested earlier, the server SHOULD ensure that the different PSK identities that it is prepared to process are readily distinguishable. If it mis-classes an identity then the binding check will fail and the session will not be established. The contents of the server PSK identity field are opaque, so this may be done by keeping all PSK identities of this server the same length or by preceding the server PSK identity with a length field. Each server PSK identity is associated with a single (EC)DH private/public keypair and hash algorithm, so the identity may be formed from a root name with the corresponding curve name (and optionally hash algorithm) appended.

When present, the client PSK identity is encrypted. The encryption key is derived as described in section Section 5.1 and depends on the client key share (among other things). For this reason, the client PSK identity cannot be present if the corresponding key share is not present and this forces the server to send a HelloRetryRequest if this server PSK identity is selected.

Any string of all zero bytes of any length is reserved for an anonymous client and MUST NOT be used for an actual client PSK identity. It is RECOMMENDED that all client PSK identities are the same length, in which case this length SHOULD be publicised so that an anonymous client can use an identity of the same length. Alternatively, if different lengths are supported for client PSK identities, then the client and server SHOULD support padding of the identity with leading zero bytes to prevent client identification by correlating the length of the encrypted PSK identity string in different handshake sequences.

4. Conformance Requirements

It is expected that different types of client and server identities are useful for different applications running over TLS. This document does not therefore mandate the use of any particular type of identity (such as IPv4 address or Fully Qualified Domain Name (FQDN)).

However, the TLS client and server clearly have to agree on the identities and keys to be used. To improve interoperability, this document places requirements on how the identity is encoded in the protocol, and what kinds of identities and keys implementations have to be supported.

The requirements for implementations are divided into two categories, requirements for TLS implementations and management interfaces. In this context, "TLS implementation" refers to a TLS library or module that is intended to be used for several different purposes, while "management interface" would typically be implemented by a particular application that uses TLS.

This document does not specify how the server stores the keys and identities, or how exactly it finds the key corresponding to the identity it receives. For instance, if the identity is a domain name, it might be appropriate to do a case-insensitive lookup. It is RECOMMENDED that before looking up the key, the server processes the client PSK identity with a PRECIS framework (see [RFC7564]) appropriate for the identity in question (such as [RFC5891] for components of domain names or [RFC8265] for usernames).

4.1. PSK Identity Encoding

The server and client PSK identities MUST be first converted to a character string, and then encoded to octets using UTF-8 [RFC3629]; the server PSK identity MAY be preceded by a single-octet length field and the client PSK identity MAY be preceded by a variable number of zero octets. For instance,

- IPv4 addresses are encoded as dotted-decimal strings (e.g. "192.0.2.1"), not as 32-bit integers in network byte order.

- Domain names are encoded in their usual text form [RFC1035] (e.g. "www.example.com" or "embedded\.dot.example.net"), not in DNS protocol format.

- X.500 Distinguished Names are encoded in their string representation [RFC4514], not as BER-encoded ASN.1.

This encoding is clearly not optimal for many types of identities. It was chosen to avoid identity-type-specific parsing and encoding code in implementations where the identity is configured by a person using some kind of management interface. Requiring such identity-type-specific code would also increase the chances for interoperability problems resulting from different implementations supporting different identity types.

4.2. Requirements for TLS Implementations

TLS implementations supporting 2/3(EC)DH key agreement MUST support arbitrary client PSK Identities up to 128 octets in length, and MUST provide a server PSK identity which uses a static ECDH key on the named curve P-256. Supporting longer identities and other ECDH curves is RECOMMENDED.

The server MUST provide information out-of-band regarding what format(s) of client PSK identity it supports for each server PSK identity, together with the server (EC)DH public key and hash function. This information includes the supported length(s) of the client PSK identity, the padding method (if any) and whether it accepts anonymous clients or not.

4.3. Requirements for Management Interfaces

In the absence of an application profile specification specifying otherwise, a management interface for entering the pre-shared private/public keys, and/or PSK Identity MUST support the following:

- Entering client and server PSK identities consisting of up to 128 printable Unicode characters. Supporting as wide a character repertoire and as long identities as feasible is RECOMMENDED.

- Entering the named curve and hash function which is associated with each server PSK identity, or selecting from a pre-set list of named curves and hash functions which are supported by the TLS implementation.

- Entering pre-shared private and public keys in the representation specified for that named curve in [I-D.ietf-tls-tls13], where each value/co-ordinate is in hexadecimal encoding up to the length required by the named curve. Supporting compressed forms and longer co-ordinates is RECOMMENDED. The interface SHOULD validate the key which was entered, if possible; that is, check that a private key is in the correct range and, for ECDH keys, that a public key is a point on the curve in the correct subgroup.

4.4. Precomputation for Constrained Clients

A number of computations in this protocol require substantial resources from a constrained client. Some of these may be precomputed, which reduces the time taken during the actual session establishment, and so may improve the success rate of the handshake.

If the client only communicates with a small number of servers, then the static-static shared secret (C_s/S_s) may be computed once only and stored for all future communications, or may be computed offline and preprovisioned. If this is done, then the secret SHOULD be protected with the same level of security as is used to protect the client private key; exposure of this key would allow the impersonation of 0-RTT traffic. If the client does not store this shared secret, then it may compute it before starting the handshake.

Prior to starting each handshake, the client should accumulate sufficient entropy to generate all the session-specific secrets. These include the client ephemeral (EC)DH keypair and the ClientHello.random value. Accumulating entropy in a constrained device may take a great deal of time; if so, then this should be done as a background task.

Prior to starting each handshake, the client will usually generate the ephemeral keypair and use it to compute the ephemeral-static (C_e/S_s) shared secret. These are necessary to encrypt the client identity. If the client has no information about the server's preferences, then it may send an initial ClientHello without any key share values so that it can learn which curve and server PSK identity to use when generating the client ephemeral key.

If the above precomputations are performed, then during the handshake the client only needs to perform one (respectively, two) public key operations for 2(EC)DH (respectively, 3(EC)DH) authenticated key agreement. These values are reduced by one for anonymous clients, but constrained devices are not expected to use this mode.

5. Cryptographic Operations

Some changes are made to the key schedule which is specified in section 7.1 of [I-D.ietf-tls-tls13] when computing keys generated using the 2/3(EC)DH mode of key exchange. These changes introduce a new derived secret which is used as a basis for encrypting client PSK identities as well two new labels which are used for generating bindings for the new PSK identities. Also, the schedule inputs are expanded to introduce multiple (EC)DH shared keys, which are the basis for this mode's authenticated key agreement.

In addition, there is a new key calculation which is used to encrypt the client PSK identity.

5.1. Key Schedule

The key schedule shown below uses the same formatting conventions and functions as used in section 7.1 of [I-D.ietf-tls-tls13]. That is,

- HKDF-Extract is drawn as taking the Salt argument from the top and the IKM argument from the left.

- Derive-Secret's Secret argument is indicated by the incoming arrow. For instance, the Early Secret is the Secret for generating the client_early_traffic_secret.

Authenticated key agreement using 2(EC)DH or 3(EC)DH makes use of both static and ephemeral (EC)DH keys, and these keys may belong to either the client or the server. This results in four possible ways to combine a client key with a server key. These are shown in the key schedule as follows:

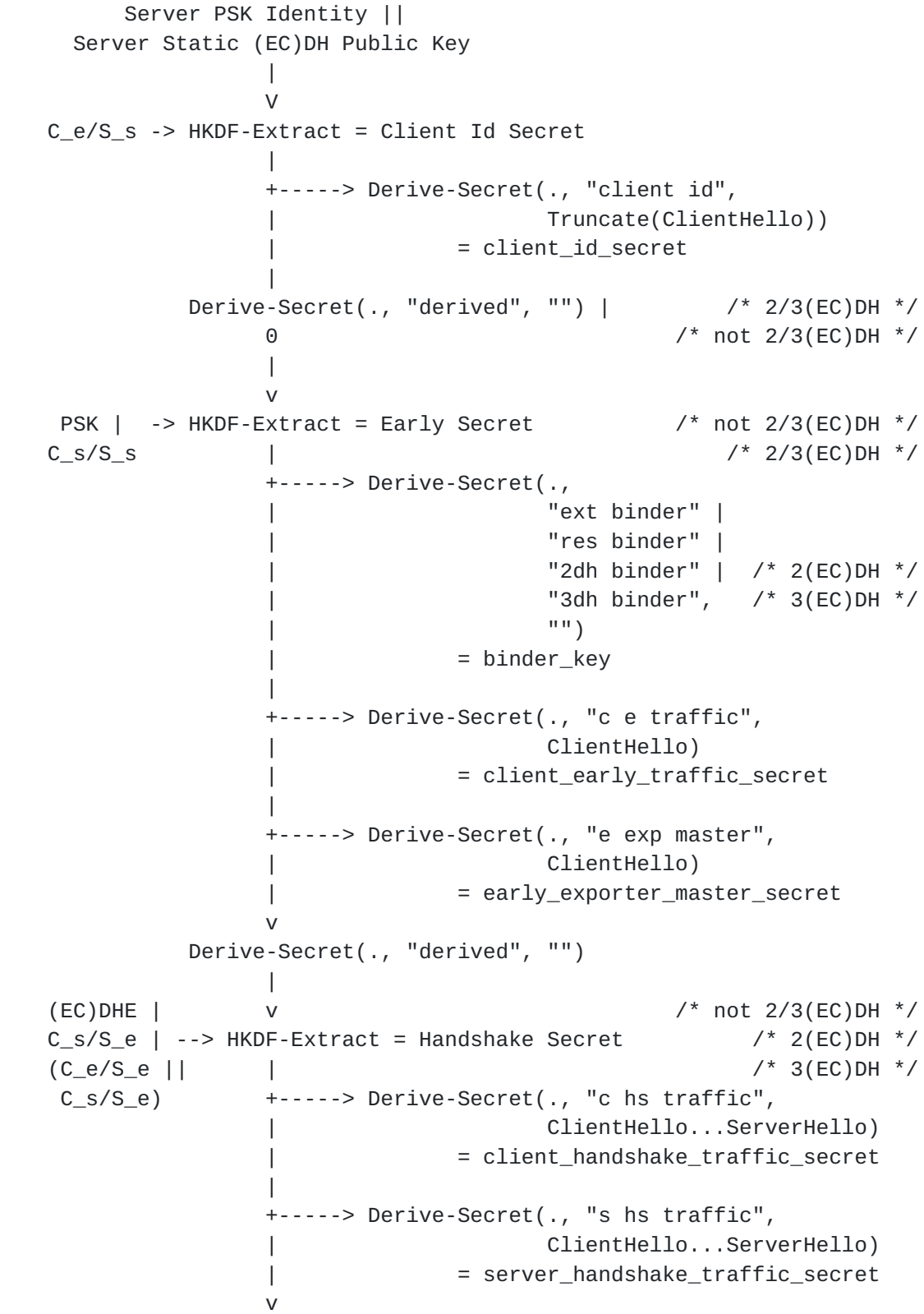
- C_s/S_s: Shared key generated using the client static (EC)DH key and the server static (EC)DH key; this is a block of zeroes if the client is anonymous.

- C_e/S_s: Shared key generated using the client ephemeral (EC)DH key and the server static (EC)DH key.

- C_s/S_e: Shared key generated using the client static (EC)DH key and the server ephemeral (EC)DH key; this is a block of zeroes if the client is anonymous.

- C_e/S_e: Shared key generated using the client ephemeral (EC)DH key and the server ephemeral (EC)DH key. This is the same as (EC)DHE used by other key agreement modes, but this terminology is used here for consistency with the other shared keys. It is not used by 2(EC)DH.

The concatenation of two octet strings is shown using || between the strings.



```
            Derive-Secret(., "derived", "")
                  |
                  v
       0 -> HKDF-Extract = Master Secret
                  |
                  +-----> Derive-Secret(., "c ap traffic",
                  |                     ClientHello...server Finished)
                  |             = client_application_traffic_secret_0
                  |
                  +-----> Derive-Secret(., "s ap traffic",
                  |                     ClientHello...server Finished)
                  |             = server_application_traffic_secret_0
                  |
                  +-----> Derive-Secret(., "exp master",
                  |                     ClientHello...server Finished)
                  |             = exporter_master_secret
                  |
                  +-----> Derive-Secret(., "res master",
                                        ClientHello...client Finished)
                                = resumption_master_secret
```

(EC)DH computations can leak information about the private key if the peer public key is not valid. Therefore, all (EC)DH computations in the key schedule MUST be preceded by a validity check on the peer public key as described in section 4.2.8.1 or section 4.2.8.2 of [I-D.ietf-tls-tls13]; additional validity checks may be added as new attacks are discovered. The endpoint MUST terminate the handshake if validation fails. Validation is required even for the static public keys, as the security of these keys is likely to be lower than for the static private key, and they may be manipulated by an attacker even if validation takes place during provisioning.

The first change to the key schedule defined in [I-D.ietf-tls-tls13] introduced to support 2/3(EC)DH is to prepend an HKDF-Extract computation to generate the Client Id Secret. The salt for this computation is the concatenation of the server PSK identity with the corresponding static public key; the IKM is the secret generated by the client ephemeral key and the server static key. For 2/3(EC)DH modes of key agreement, the salt for the next step of the key schedule is a secret derived from the Client Id Secret; for other modes the salt is a zero block. This means that this specification results in no change to the key schedule for all other modes.

Next, the IKM for the generation of the Early Secret for 2/3(EC)DH is the secret which is generated by the client static key and the server static key. This is a long-term secret which is equivalent to the PSK which is the IKM for other PSK modes. The inclusion of keying material based on the client static (EC)DH key provides data origin

authenication for 0-RTT traffic (if present). For anonymous clients, this IKM is a zero block; data origin authentication has no meaning in any case for anonymous clients.

Additional labels are defined for the generation of the binder_key which is used to bind the identity and key to the current handshake. The label "2dh binder" (respectively "3dh binder") is used if the selected mode is 2(EC)DH (respectively 3(EC)DH).

The last change to the key schedule is to the IKM used to derive the Handshake Secret. As defined in [I-D.ietf-tls-tls13], this is the secret generated by the client ephemeral key and the server ephemeral key (if available). For 2(EC)DH, this is the secret generated by the client static key and the server ephemeral key. For 3(EC)DH, this is the concatenation of the secrets generated by the client ephemeral key and the server ephemeral key, and by the client static key and the server ephemeral key in that order.

All (EC)DH computations are carried out as described in section 7.4 of [I-D.ietf-tls-tls13].

5.2. Client Id Key Calculation

Each offered client PSK identity within a handshake is associated with a different server PSK identity, which means that the key schedule ensures that a different Client Id Secret is computed for each identity. This means that each derived client_id_secret is different within a handshake.

However, it is possible that a client may re-use an ephemeral key in multiple handshakes (though this is bad practice). If the client pads the identity differently or manages multiple identities then this will leak identity information. Therefore, the truncated handshake context is included in the computation of the client_id_secret to mitigate possible ephemeral key reuse.

The Truncate() function which is used when computing client_id_secret is similar to that used in computing the binding values (see section 4.2.11.2 of [I-D.ietf-tls-tls13]), except that it excludes the entire "pre_shared_key" extension, instead of just removing the binders list. If the ClientHello is sent in response to a HelloRetryRequest then the Truncate() function only applies to the second ClientHello; the full first ClientHello message is included in the Transcript-Hash as defined in section 4.4.1. of [I-D.ietf-tls-tls13].

The encrypted identity is integrity-protected by the corresponding PSK binder, so only encryption is required. Therefore, the HKDF-Expand-Label function is used to generate a keystream of the same

length as the padded client PSK identity and the two are XORed together to produce the encrypted identity. The decryption procedure is identical.

```
client_id_key = PKDF-Expand-Label(client_id_secret,
                                  "client id",
                                  length(padded client PSK identity))
```

6. Acknowledgements

This document is based on [I-D.putman-tls-preshared-ecdh] and the author would like carry over the acknowledgements to this document; that is, to thank the authors of [RFC4279], namely Pasi Eronen, Hannes Tschofenig, Mohamad Badra, Omar Cherkaoui, Ibrahim Hajjeh and Ahmed Serhrouchni; and to thank the author of [I-D.harkins-tls-dragonfly], Dan Harkins, for the idea of encrypting the client identity.

7. IANA Considerations

This document does not define any new IANA considerations.

8. Security Considerations

The security considerations in [I-D.ietf-tls-tls13] apply to this document as well.

8.1. Security of 1-RTT Traffic

The Double- and Triple-Diffie-Hellman authenticated key exchange is not particularly new, but it has not seen wide usage. Triple-ECDH is used in the Signal protocol [Marlinspike] and a security proof of both, in a modified form of the Bellare-Rogaway model, is provided in [Kudla]; this latter paper also proves strong partnering in the random oracle model.

The TLS 1.3 protocol is more complex than the protocol used in the above proof, but no additional constraints are made on the components of the proof if the client is not anonymous. All the material which is used to compose the key in the proof is also used in constructing the Finished messages, with the exception of the pair of static keys: instead of including both static keys in the key schedule, the derived secret is included; this is equivalent and is done to give extra protection against weak ephemeral keys. The security proof therefore also holds for the protocol described in this document if the client is not anonymous.

If the client is anonymous, then there is no static client (EC)DH keypair. The security properties of this situation are similar to those of DHIES/ECIES [ABR], which also derives a key from a combination of an ephemeral and a static (EC)DH key. The inclusion of handshake context in the generation of all session keys protects against replay attacks, and the inclusion of the ephemeral-ephemeral key in the 3(EC)DH anonymous client situation additionally provides PFS in this exchange. However, none of this is rigorous, and a full security analysis of this should be undertaken.

8.2. Security of 0-RTT Traffic

The security of the 0-RTT traffic has the same weakened conditions as for 0-RTT traffic in other PSK situations, namely that there is no PFS and there is no guarantee of traffic uniqueness.

8.3. Security of Client Identity

The confidentiality of the client identity in a genuine handshake is protected by an ephemeral-static shared secret; this has similar properties to DHIES/ECIES [ABR]. The integrity of the key is protected by the binder.

The encrypted client PSK identity does not have any verified client information in the keying material. Therefore an attacker who knows the server public key may impersonate a client when sending a client key exchange message; this is no different to the other PSK key exchange modes and does not affect the security of the completed handshake.

Because a PSK Identity can be forged, the server should ensure that there are no PSK Identity retrievals which are more expensive than other operations in this protocol; this is to mitigate DoS attacks. Additionally, if there are differences in the lookup time of a PSK Identity (e.g. if recent lookups are cached), then an attacker may be able to obtain information about the PSK Identity of a recent handshake from timing attacks.

8.4. Additional Security Observations

This key exchange mode allows for the inclusion of 0-RTT traffic from a client which has no previous communications with the server, only an out-of-band method of obtaining the server PSK information (identity, public key, anonymous client support). This supports rapid client authentication at the application level, for example by using an authentication token. However, there is currently no way to convey this information to the TLS server, which will continue to consider the client to be anonymous. Further work would be needed

before this use case could be considered to be secure, and it is not considered further in this document.

As with the other PSK key exchange modes, these modes make use of hidden information in the construction of the keying material. This means that the cipher suites are quantum-safe in the event that the message exchange is stored for later attack, provided that the client and/or server static public keys (the pre-provisioned keys) remain unknown to the eavesdropper. In the anonymous client use case, the server public keys is likely to be widely known, so this observation does not apply.

The protocol description in this document only refers to finite field and elliptic curve Diffie-Hellman keys, but will work for any key exchange mechanism which uses public/private keypairs to establish a shared secret. The only change that is needed to support new methods (which may include quantum-safe key exchange algorithms) is to include their definition in the "key_share" extension. Any new mechanisms which are added to the "key_share" extension MUST include a consideration of their effect on this document in their security section.

9. References

9.1. Normative References

[I-D.ietf-tls-tls13]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-23 (work in progress), January 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[ABR] Abdalla, M., Bellare, M., and P. Rogaway, "DHIES: An Encryption Scheme based on the Diffie-Hellman Problem", 2001, <http://web.cs.ucdavis.edu/~rogaway/papers/dhies.pdf>.

[Blake-Wilson] Blake-Wilson, S., Johnson, D., and A. Menezes, "Key Agreement Protocols and their Security Analysis", Cryptography and Coding volume 1355 of LNCS, 1997.

[I-D.harkins-tls-dragonfly] Harkins, D., "Secure Password Ciphersuites for Transport Layer Security (TLS)", draft-harkins-tls-dragonfly-02 (work in progress), August 2017.

[I-D.putman-tls-preshared-ecdh] Putman, T., "ECDH-based Authentication using Pre-Shared Asymmetric Keypairs for (Datagram) Transport Layer Security ((D)TLS) Protocol version 1.2", draft-putman-tls-preshared-ecdh-00 (work in progress), November 2017.

[Kudla] Kudla, C. and K. Paterson, "Modular Security Proofs for Key Agreement Protocols", Advances in Cryptology ASIACRYPT 2005: 11th International Conference on the Theory and Application of Cryptology and Information Security, 2005, <http://www.isg.rhul.ac.uk/~kp/ModularProofs.pdf>.

[Marlinspike] Marlinspike, M. and T. Perrin, "The X3DH Key Agreement Protocol", November 2016, <https://www.signal.org/docs/specifications/x3dh/x3dh.pdf>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, DOI 10.17487/RFC4279, December 2005, <https://www.rfc-editor.org/info/rfc4279>.

[RFC4514] Zeilenga, K., Ed., "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names", RFC 4514, DOI 10.17487/RFC4514, June 2006, <https://www.rfc-editor.org/info/rfc4514>.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <https://www.rfc-editor.org/info/rfc5077>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <https://www.rfc-editor.org/info/rfc5891>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/info/rfc7250>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

[RFC7564] Saint-Andre, P. and M. Blanchet, "PRECIS Framework: Preparation, Enforcement, and Comparison of Internationalized Strings in Application Protocols", RFC 7564, DOI 10.17487/RFC7564, May 2015, <https://www.rfc-editor.org/info/rfc7564>.

[RFC8265] Saint-Andre, P. and A. Melnikov, "Preparation, Enforcement, and Comparison of Internationalized Strings Representing Usernames and Passwords", RFC 8265, DOI 10.17487/RFC8265, October 2017, <https://www.rfc-editor.org/info/rfc8265>.

Author's Address

Tony Putman
Dyson Technology
Malmesbury SN16 0RP
UK

Email: tony.putman@dyson.com