TLS Working Group T. Putman
Internet-Draft Dyson Technology
Intended status: Informational November 30, 2017
Expires: June 3, 2018

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

Abstract

This document defines a new mutual authentication method for the Transport Layer Security (TLS) protocol version 1.2. The authentication method requires that the client and server are each pre-provisioned with a unique asymmetric Elliptic Curve Diffie-Hellman (ECDH) keypair and with the public ECDH key of the peer. The handshake provides ephemeral ECDH keys, and a premaster key is agreed using Double- or Triple-ECDH; confirmation of possession of this key provides mutual authentication. Multiple new cipher suites which use this authentication method are specified.

Status of This Memo

Copyright Notice

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 would be 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 is intended to replace the use

of symmetric pre-shared keys, so it should only be used in the same rather limited set of deployments. It assumes that an OOB method of pre-configuring the asymmetric keys into the endpoints exists, and this method is outside the scope of this document.

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

1.1. Rationale for Choice of Authentication Algorithm

In TLS 1.2 [RFC5246], all cipher suites 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 authenticated key-agreement methods introduced in this document are primarily for use by constrained devices, so only Elliptic Curve algorithms are considered.

In order to be usable by as many constrained devices as possible, this proposal uses only a single algorithm, namely the ECDH algorithm. 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.

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

In order to break PFS in Double-ECDH, the attacker must obtain the static ECDH private keys of both client and server. This is likely to be a difficult feat, so both Dual- and Triple-ECDH are specified in this document.

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-ECDH SHOULD be used for those deployments where the client devices are able to support the additional computation.

1.2. Specified Cipher Suites

This document specifies the new Double-ECDH and Triple-ECDH authentication algorithms together with a number of existing AEAD cipher algorithms, namely AES-GCM [RFC5288], AES-CCM [RFC6655] and

ChaCha20-Poly1305 [RFC7905], as well as with the NULL cipher from TLS 1.2 [RFC5246]. A summary of these cipher suites is shown below.

Cipher Suite	Authenticated Key Agreement
TLS_2ECDH_WITH_AES_128_GCM_SHA256	Double-ECDH
TLS_2ECDH_WITH_AES_256_GCM_SHA384	Double-ECDH
TLS_2ECDH_WITH_AES_128_CCM_8_SHA256	Double-ECDH
TLS_2ECDH_WITH_AES_128_CCM_SHA256	Double-ECDH
TLS_2ECDH_WITH_NULL_SHA256	Double-ECDH
TLS_2ECDH_WITH_NULL_SHA384	Double-ECDH
TLS_2ECDH_WITH_CHACHA20_POLY1305_SHA256	Double-ECDH
TLS_3ECDH_WITH_AES_128_GCM_SHA256	Triple-ECDH
TLS_3ECDH_WITH_AES_256_GCM_SHA384	Triple-ECDH
TLS_3ECDH_WITH_AES_128_CCM_8_SHA256	Triple-ECDH
TLS_3ECDH_WITH_AES_128_CCM_SHA256	Triple-ECDH
TLS_3ECDH_WITH_NULL_SHA256	Triple-ECDH
TLS_3ECDH_WITH_NULL_SHA384	Triple-ECDH
TLS_3ECDH_WITH_CHACHA20_POLY1305_SHA256	Triple-ECDH

1.3. Applicability Statement

The cipher suites defined in this document are 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 PSK [RFC4279] or ECDHE_PSK [RFC5489] cipher suites may be more appealing. If the main goal is to avoid Public-Key Infrastructures (PKIs), then the use of raw public keys [RFC7250] may be preferrable.

If the relationship between client and server is not close, for example if the client enrols with the server, then a password-based cipher suite such as SRP [RFC5054] or Dragonfly [I-D.harkins-tls-dragonfly] may result in lower management overheads.

1.4. Client Impact Compared to Alternatives

Compared with PSK, the use of asymmetric keys doubles the number of keys that a client must be configured with: each client must have a unique private key and must also have a corresponding server public key. In addition, the ECDH keys are approximately twice as large as the symmetric keys of equivalent cryptographic strength.

The client also needs its public key for premaster secret generation. This can either be preconfigured or it may be generated from the private key and the generator (the latter may reside in code space). This is a time-space tradeoff.

The client using Double- or Triple ECDH must perform two (resp. three) public key operations: this is two (resp. three) more than is used for PSK authentication; it is one (resp. two) more than is used for ECDHE_PSK authentication, and it is one fewer (resp. approximately that same) as is used for a client-authenticated ECDHE exchange for other schemes.

As the keys are pre-shared, there is no need for the Certificate or CertificateVerify handshake messages to be sent. This saves a considerable amount of data in the handshake exchange which helps to make the protocol more robust in deployments which have unreliable network connectivity.

1.5. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] and [RFC8174].

2. 2ECDH and 3ECDH Key Exchange Algorithm

This section defines the key exchange algorithm and associated cipher suites which are used for 2ECDH and 3ECDH. It is assumed that the reader is familiar with the ordinary TLS handshake, shown below. The elements in parenthesis are not included when the 2ECDH or 3ECDH key exchange algorithm is used. The message exchange is identical to that used for the DHE_PSK handshake defined in Pre-Shared Key Ciphersuites for TLS [RFC4279], though the message contents differ a little.

```
      Client                                               Server
      ------                                               ------

      ClientHello                  -------->
                                                      ServerHello
                                                    (Certificate)
                                                ServerKeyExchange
                                             (CertificateRequest)
                                   <--------      ServerHelloDone
      (Certificate)
      ClientKeyExchange
      (CertificateVerify)
      ChangeCipherSpec
      Finished                     -------->
                                                 ChangeCipherSpec
                                   <--------             Finished
      Application Data             <------->     Application Data
```

The client indicates its willingness to use 2ECDH or 3ECDH authentication by including one or more corresponding cipher suites in the ClientHello message. If the TLS server also wants to use 2ECDH or 3ECDH, it selects one of the corresponding cipher suites, places the selected cipher suite in the ServerHello message, and includes an appropriate ServerKeyExchange message. The Certificate and CertificateRequest payloads are omitted from the response.

The server will have to establish sessions with multiple clients and so will have multiple client ECDH public keys. The client indicates which key to use by including an encrypted "PSK identity" in the ClientKeyExchange message. To help the client in selecting which identity to use, the server can provide an unencrypted "PSK identity hint" in the ServerKeyExchange message.

The client may have static ECDH keypairs on more than one curve associated with the same PSK Identity. For example, during a transition to a more secure curve, there will likely be a period when both curves are supported. The client MUST have only a single static ECDH keypair per curve for a given PSK Identity.

The server, which here means a server endpoint hosting the TLS functionality, may similarly have static ECDH keypairs on more than one curve. The server MUST have only a single static ECDH keypair per curve. This does not prevent a single physical device from having multiple static ECDH keypairs on a single curve, but if this is done then each MUST be associated with a different TLS endpoint.

The client SHOULD include the Supported Elliptic Curves Extension in the ClientHello message, listing all curves for which it holds a

static ECDH private key. If it lists other ECDH-related cipher suites in the list of supported cipher suites, then there is a risk that the server will select a 2ECDH- or 3ECDH- cipher suite using a curve which does not correspond to a key the client holds. In this case, the client should restart the handshake, omitting the selected curve from the Supported Elliptic Curves extension. Such an occurrence is expected to be rare, as there is no reason to suggest other cipher suites if the client knows that the server supports the 2ECDH- or 3ECDH- cipher suites.

In order to comply with [I-D.ietf-tls-rfc4492bis], the client and server MUST only use the uncompressed format for ECDH public keys. The client and server SHOULD include the Supported Point Formats Extension in the ClientHello (resp. ServerHello) message indicating support for only the uncompressed format.

The cipher suites in this document apply only to TLS 1.2. The server MUST NOT select any of these cipher suites if a different TLS version is being negotiated.

The ServerKeyExchange and ClientKeyExchange messages also include the Elliptic Curve Diffie-Hellman parameters for the ephemeral keys. Note that the ECDH curve parameters MUST be those of the selected pre-shared asymmetric key.

The format of the ServerKeyExchange and ClientKeyExchange messages is shown below.

```
struct {
    select (KeyExchangeAlgorithm) {
        /* other cases for rsa, diffie_hellman, etc. */
        case 2ec_diffie_hellman:  /* NEW */
            opaque psk_identity_hint<0..2^16-1>;
            ServerECDHParams params;
        case 3ec_diffie_hellman:  /* NEW */
            opaque psk_identity_hint<0..2^16-1>;
            ServerECDHParams params;
    };
} ServerKeyExchange;

struct {
    select (KeyExchangeAlgorithm) {
        /* other cases for rsa, diffie_hellman, etc. */
        case 2ec_diffie_hellman:   /* NEW */
            opaque encrypted_psk_identity<0..2^16-1>;
            ClientECDiffieHellmanPublic public;
        case 3ec_diffie_hellman:   /* NEW */
            opaque encrypted_psk_identity<0..2^16-1>;
            ClientECDiffieHellmanPublic public;
    } exchange_keys;
} ClientKeyExchange;
```

The structures ServerECDHParams and ClientECDiffieHellmanPublic are the same as defined in [I-D.ietf-tls-rfc4492bis]. The semantics of psk_identity_hint is given in section Section 3.3. The contents of encrypted_psk_identity is defined in Section 3.1 and Section 3.2.

3. Conformance Requirements

It is expected that different types of 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 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).

3.1. PSK Identity Encoding

The PSK identity MUST be first converted to a character string, and then encoded to octets using UTF-8 [RFC3629]. For instance,

- o IPv4 addresses are sent as dotted-decimal strings (e.g. "192.0.2.1"), not as 32-bit integers in network byte order.
- o Domain names are sent in their usual text form [RFC1035] (e.g. "www.example.com" or "embedded\.dot.example.net"), not in DNS protocol format.
- o X.500 Distinguished Names are sent 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.

3.2. PSK Identity Protection

The PSK Identity MUST be encrypted with the selected authenticated encryption algorithm using the client_write_key derived from the pskid_master_secret described in section 4.2. The GenericAEADCipher.nonce_explicit is a sequence of zero octets of the length appropriate for the cipher and there is no "additional data".

3.3. Identity Hint

In the absence of an application profile specification specifying otherwise, servers SHOULD NOT provide an identity hint and clients MUST ignore the identity hint field. Applications that do use this field MUST specify its contents, how the value is chosen by the TLS server, and what the TLS client is expected to do with the value.

3.4. Requirements for TLS Implementations

TLS implementations supporting these cipher suites MUST support arbitrary PSK Identities up to 128 octets in length, and MUST support P-256. Supporting longer identities and other ECDH curves is RECOMMENDED.

3.5. 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 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 elliptic curve or curves which are supported together with the corresponding generator(s).

- Entering pre-shared private and public keys in uncompressed form with each co-ordinate being up to 32 octets in length in hexadecimal encoding. 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 that a public key is a point on the curve in the correct subgroup.

4. Cryptographic Operations

Two different premaster secrets and their corresponding master secrets are computed for this handshake. The first is to encrypt/decrypt the PSK Identity and the second is for use in the actual TLS session. Both of these use similar algorithms, though they differ in the algorithm parameters.

4.1. Computing the Premaster Secrets

The structure for all premaster secrets for this document is the same, namely:

```
struct {
    opaque ephemeral_ecdh<0..2^16-1>;
    opaque client_static_ephemeral_ecdh<0..2^16-1>;
    opaque server_static_ephemeral_ecdh<0..2^16-1>;
    opaque client_static_ecdh<0..2^16-1>;
    opaque server_static_ecdh<0..2^16-1>;
};
```

The elements of this struct are populated differently, depending on both the cipher suite which has been selected and on the premaster key which is being constructed.

All ECDH computations are carried out as described in section 5.10 of [I-D.ietf-tls-rfc4492bis]. The public key validation described in section 5.11 of [I-D.ietf-tls-rfc4492bis] MUST always be carried out; for X25519 and X448, the receiving party check MUST be applied to each ECDH computation, not just to the overall premaster secret.

All computations use the same curve, which is indicated in ServerECDHParams, so they all result in an output string of the same known length. For those elements of the premaster secret struct which do not involve an ECDH computation, the element is the same length as the ECDH output and is filled with zero bytes.

For all cipher suites which use Double-ECDH, the ephemeral_ecdh element is constructed as a uint16 containing the length of the ECDH output followed by that number of zero bytes. For all cipher suites which use Triple-ECDH, the ephemeral_ecdh element is constructed as a uint16 containing the length of the ECDH output followed by the octet string which is the output of the ECDH computation using both ephemeral ECDH keys.

The client_static_ephemeral_ecdh element for the premaster secret which is used to derive keys to protect the PSK Identity is constructed as a uint16 containing the length of the ECDH output followed by that number of zero bytes. The client_static_ephemeral_ecdh element for the premaster secret which is used to derive keys for the TLS session is constructed as a uint16 containing the length of the ECDH output followed by the octet string which is the output of the ECDH computation using the client's static (pre-shared) ECDH key and the server's ephemeral ECDH key.

For all cipher suites specified in this document, the server_static_ephemeral_ecdh element is constructed as a uint16 containing the length of the ECDH output followed by the octet string which is the output of the ECDH computation using the server's static (pre-shared) ECDH key and the client's ephemeral ECDH key.

The client_static_ecdh and server_static_ecdh keys are not present in the message exchange, so they are included here to mix them into the session key. They are represented in the same way as all the other results, namely as the x-coordinate of the public key represented as an octet string (with leading zeros included). The server_static_ecdh key is alway present, but the client_static_ecdh key is replaced by a sequence of zeros in the premaster secret structure which is used to derive keys to protect the PSK Identity.

4.2. Computing the Master Secrets

The algorithm which is used to convert the pre_master_secret for protecting the PSK Identity into the corresponding master_secret uses a similar construction to that used for the TLS session master key.

```
pskid_master_secret = PRF(pskid_premaster_secret, "psk identity",
                          ClientHello.random + ServerHello.random)
                          [0..47];
```

The pskid_premaster_secret does not need to be deleted from memory once the pskid_master_secret has been computed: it may be retained for use in constructing the premaster_secret for the TLS session. It SHOULD be deleted either when the TLS session premaster_secret is deleted or if the handshake exchange is terminated early for some reason.

The algorithm which is used to convert the premaster_secret for the TLS session to the corresponding master secret is the standard TLS 1.2 method described in section 8.1 of [RFC5246].

5. Acknowledgements

The document structure is based heavily on [RFC4279] and a fair amount of text was copied from that, so the author would like to thank the authors of that RFC, namely Pasi Eronen, Hannes Tschofenig, Mohamad Badra, Omar Cherkaoui, Ibrahim Hajjeh and Ahmed Serhrouchni. The idea of encrypting the PSK Identity, though not the method, was borrowed from [I-D.harkins-tls-dragonfly], so the author would also like to thank Dan Harkins.

6. IANA Considerations

This document defines the following new cipher suites, whose values have been assigned in the TLS Cipher Suite Registry defined by [RFC5246].

```
TLS_2ECDH_WITH_AES_128_GCM_SHA256        = {0xTBD; 0xTBD}
TLS_2ECDH_WITH_AES_256_GCM_SHA384        = {0xTBD; 0xTBD}
TLS_2ECDH_WITH_AES_128_CCM_8_SHA256      = {0xTBD; 0xTBD}
TLS_2ECDH_WITH_AES_128_CCM_SHA256        = {0xTBD; 0xTBD}
TLS_2ECDH_WITH_NULL_SHA256               = {0xTBD; 0xTBD}
TLS_2ECDH_WITH_NULL_SHA384               = {0xTBD; 0xTBD}
TLS_2ECDH_WITH_CHACHA20_POLY1305_SHA256  = {0xTBD; 0xTBD}
TLS_3ECDH_WITH_AES_128_GCM_SHA256        = {0xTBD; 0xTBD}
TLS_3ECDH_WITH_AES_256_GCM_SHA384        = {0xTBD; 0xTBD}
TLS_3ECDH_WITH_AES_128_CCM_8_SHA256      = {0xTBD; 0xTBD}
TLS_3ECDH_WITH_AES_128_CCM_SHA256        = {0xTBD; 0xTBD}
TLS_3ECDH_WITH_NULL_SHA256               = {0xTBD; 0xTBD}
TLS_3ECDH_WITH_NULL_SHA384               = {0xTBD; 0xTBD}
TLS_3ECDH_WITH_CHACHA20_POLY1305_SHA256  = {0xTBD; 0xTBD}
```

NOTE TO THE RFC EDITOR: PLEASE REMOVE THIS PARAGRAPH. Please replace each instance of {0xTBD; 0xTBD} with the appropriate IANA-assigned values. All cipher suites are suitable for DTLS and none is IETF-recommended.

7. Security Considerations

The security considerations in TLS 1.2 [RFC5246], DTLS 1.2 [RFC6347], ECC Cipher Suites for TLS (including Curve25519 and Curve448) [I-D.ietf-tls-rfc4492bis], AES-GCM [RFC5288], AES-CCM [RFC6655] and ChaCha20-Poly1305 [RFC7905] apply to this document as well.

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 1.2 protocol is more complex than the protocol used in the above proof, but no additional constraints are made on the components of the proof. All the material which is used to compose the key in the proof is also used in constructing the Finished messages: the ephemeral public keys form part of the handshake messages and the remaining material is used in the construction of the premaster secret. Therefore, the security proof also holds for the protocol as described in this document.

The security of the PSK Identity is weaker than that of the completed protocol: it 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 PSK cipher suites 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.

The NULL cipher suites do not provide confidentiality, so these must not be used in situations where sensitive or private data (e.g. passwords) is tranmitted. The cipher suite used for protecting the PSK Identity is the same as that used for protecting the TLS session. If NULL encryption is chosen for the session, then the PSK Identity is not confidential either.

As with the PSK cipher suites, these protocols make use of hidden information in the construction of the keying material. This means that the cipher suites are quantum-safe in the event of storage of the message exchange for later attack, provided that the client and/ or server static public keys (the pre-provisioned keys) remain unknown to the eavesdropper.

The overall security level of the solution depends on the security of the cipher suite together with the security of the Elliptic Curve chosen for the pre-shared key. The curve and the cipher suite SHOULD be chosen to have approximately the same security level so that the processing load on the client is minimised for a required security level. For example, X25519 (128 bits of security) could be the chosen ECDH algorithm for use with the TLS_3ECDH_WITH_AES_128_CCM_8_SHA256 cipher suite.

8. References

8.1. Normative References

[I-D.ietf-tls-rfc4492bis]
Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", draft-ietf-tls-rfc4492bis-17 (work in progress), May 2017.

[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>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, DOI 10.17487/RFC5288, August 2008, <https://www.rfc-editor.org/info/rfc5288>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC6655] McGrew, D. and D. Bailey, "AES-CCM Cipher Suites for Transport Layer Security (TLS)", RFC 6655, DOI 10.17487/RFC6655, July 2012, <https://www.rfc-editor.org/info/rfc6655>.

[RFC7905] Langley, A., Chang, W., Mavrogiannopoulos, N., Strombergson, J., and S. Josefsson, "ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)", RFC 7905, DOI 10.17487/RFC7905, June 2016, <https://www.rfc-editor.org/info/rfc7905>.

[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>.

8.2. Informative References

[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.

[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>.

[RFC5054] Taylor, D., Wu, T., Mavrogiannopoulos, N., and T. Perrin, "Using the Secure Remote Password (SRP) Protocol for TLS Authentication", RFC 5054, DOI 10.17487/RFC5054, November 2007, <https://www.rfc-editor.org/info/rfc5054>.

[RFC5489] Badra, M. and I. Hajjeh, "ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)", RFC 5489, DOI 10.17487/RFC5489, March 2009, <https://www.rfc-editor.org/info/rfc5489>.

[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