TLS
Internet-Draft
Intended status: Standards Track
Expires: November 6, 2015

BZ. Wu
Alibaba Inc.

May 5, 2015

Transport Layer Security (TLS) Client Keyshare Extension
draft-bzwu-tls-client-keyshare-01

Abstract

This document defines an extension that allows a TLS client to carry Diffie-Hellman (DH) keyshares in ClientHello message, replacing the ClientKeyExchange message in the second round-trip, to reduce the full handshake latency to one network round-trip time (RTT).

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 http://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 November 6, 2015.

Copyright Notice

Table of Contents

1. Introduction

A full TLS handshake as specified in [TLSv1.2] requires 2-RTT, mostly because of the ClientKeyExchange message in the second round-trip, which is used for the key exchange. TLS 1.3, which works in progress, will offer a 1-RTT mode by sending DH keyshare immediately after the ClientHello in the first round-trip, called ClientKeyShare message. However, it will take a long time to finalize the draft and deploy TLS 1.3.

This document defines a TLS extension that allows a client using TLS 1.2 to carry DH keyshares in the ClientHello message with the first round-trip. The client and server complete key exchange by the keyshares and native ServerKeyExchange message, thus the ClientKeyExchange message is not necessary and could be omitted. This leads to a latency reduction of one round-trip.

This extension is intended for TLS 1.2 only, but no previous versions [TLSBCP]. This extension only supports Ephemeral DH, but no static DH [TLSBCP]. This extension supports Elliptic Curve (EC) and Finite Field (FF) keyshare types. Only NamedCurves [TLSECC] (for EC type) and NegotiatedParameters (work in progress) (for FF type) are supported, while generic parameters are not supported for safety and simplicity. This extension dose not work if client certificates are involved, which needs the second round-trip too.

2. Requirements Notation

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 RFC 2119 [KEYWORDS].

3. Client Keyshare Extension

3.1. Extension Type

This document defines a new extension type (client_keyshare(TBD)), which is used in the ClientHello and ServerHello messages. The extension type is specified as follows.

```
enum {
     client_keyshare(TBD), (65535)
} ExtensionType;
```

3.2. Extension-data Specification

The extension_data field of this extension, when included in the ClientHello message, MUST contain the ClientKeyshare structure, which offers one or more ClientKeyShareOffer values, each representing a single set of DH key agreement parameters. The shares for each ClientKeyShareOffer MUST be generated independently. Clients MUST NOT offer multiple ClientKeyShareOffers for the same parameters. The shares SHOULD keep the same order as with elliptic_curves extension [TLSv1.2], to indicate client's preference.

```
struct {
     ClientKeyShareOffer offers<0..2^16-1>;
} ClientKeyShare;

struct {
     byte          type(3);
     NamedGroup    group_id;
     select (typeof(group_id)) {
         case FF:  ClientDiffieHellmanPublic;
         case EC:  ECPoint;
     } public_key;
} ClientKeyShareOffer;
```

type
: Since only NamedCurves and NegotiatedParameters are supported in this extension, this byte exists only for compatibility with ECCurveType in ECParameters [TLSECC], and its value MUST always be 3.

group_id
: Specifies the DH parameters associated with the public key. NamedGroup is extended from NamedCurve [TLSECC] by Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for TLS (work in progress) to support finite-field-based DH.

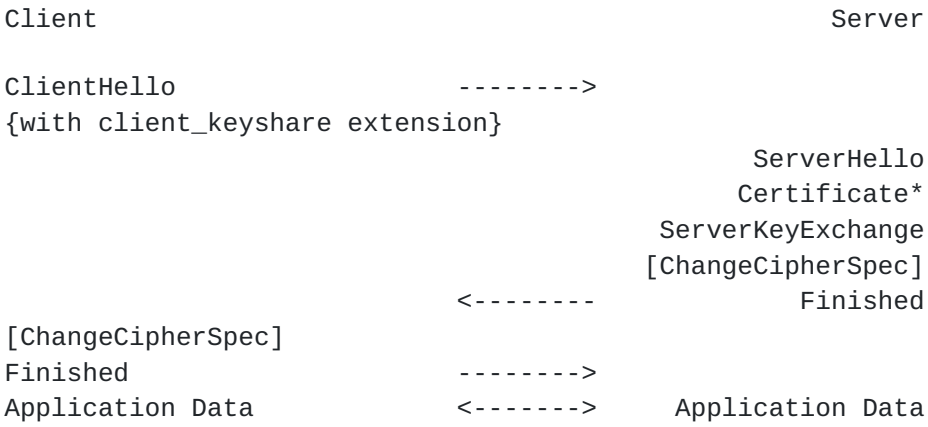
public_key
The ephemeral DH public key. It's in ClientDiffieHellmanPublic format [TLSv1.2] for FF type, or in ECPoint format [TLSECC] for EC type.

Because the key exchange is made by the ClientKeyshare extension and ServerKeyExchange message, it's not necessary to parse values in extension_data if included in the ServerHello message. The server just echoes the extension with an empty extension_data to indicate support in the current this session.

3.3. Message Flow with This Extension

In brief, the full handshake works as follows with this extension. A client takes this extension with some DH keyshares in the ClientHello message. A server receiving this extension echoes it in the ServerHello message to indicate support within this session and sends a ServerKeyExchange to complete the key exchange (with the DH keyshare in client's extension). Since there is no ClientKeyExchange to wait for, the server sends no ServerHelloDone, but ChangeCipherSpec and Finished message immediately, similar to an abbreviated handshake flow.

The message flow is illustrated in the this figure.



This works only if client and server both support the extension. For example, if a server which does not support this extension receives a ClientHello message with this extension, the server MUST ignore it.

This extension only works if the negotiated key exchange algorithm is Ephemeral Diffie-Hellman (FFDHE or ECDHE). Obviously, the client has to send a ClientKeyExchange message after getting the server's certificate if it is using RSA as key exchange. Thus it can not benefit from this extension. Although the client may get server's certificate before the handshake by the Cached Infomation extension

(work in progress), we do not support RSA key exchange for simplicity. TLS 1.3 will remove support for RSA key-exchange entirely and RSA as key-exchange is discouraged [TLSBCP]

Since the client does not know which DH types and parameters the server supports, it MAY takes more than one DH keyshare in this extension. The server picks one DH keyshare of the same type as with the key exchange algorithm (FF or EC) and acceptable parameters, generates a DH keyshare with the same parameters, sends it in ServerKeyExchange message, and completes the key exchange with these two keyshares. If there is no suitable keyshare in client's extension, the server MUST ignore this extension.

A client enables this extension only if the server echoes this extension in the ServerHello message. The client picks the ClientKeyShareOffer containing the same parameters as with ServerKeyExchange. If there is no such ClientKeyShareOffer, the client MUST abort the handshake with an illegal_parameter fatal alert.

If this extension is enabled, a server does not wait for the ClientKeyExchange, or send a ServerHelloDone message; instead it sends ChangeCipherSpec and Finished messages immediately, like with an abbreviated handshake. Accordingly a client dose not send ClientKeyExchange or wait for ServerHelloDone message.

A server does not enable this extension if it requests client's certificate, which needs the second round-trip too.

Finally, this extension only works in full handshake, while not in abbreviated handshake which does not need key exchange.

4. Interaction

Server sends ChangeCipherSpec and Finished messages after the ServerKeyExchange, if this extension is enabled. However there may be messages between the ServerKeyExchange and ChangeCipherSpec, e.g. NewSessionTicket, if the Session Ticket extension is used [TICKET].

With the Session Hash extension (work in progess) "handshake_messages" refer to all handshake messages up to and including the ClientKeyExchange message. There is no ClientKeyExchange if this client_keyshare extension is enabled. The "handshake_messages" should be changed to refer to all handshake messages up to and including the ServerKeyExchange message, without breaking the Session Hash extension.

5. Security Considerations

This extension brings client's DH keyshare forward, from the ClientKeyExchange message in the second round-trip, to ClientHello message in the first round-trip. TLS 1.3 (works in progress) also works like this. So there should not be any security problem introduced.

6. IANA Considerations

IANA is requested to add an entry to the existing TLS ExtensionType registry, defined in TLS [TLSv1.2], for client_keyshare(TBD) defined in this document.

7. Acknowledgements

Thanks to Ilari Liusvaara and Aaron Zauner for their valuable comments and suggestions on this draft.

8. References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[TICKET] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.

[TLSBCP] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7525, May 2015.

[TLSECC] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, May 2006.

[TLSv1.2] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

Author's Address

Bingzheng Wu
Alibaba Inc.

EMail: bingzheng.wbz@alibaba-inc.com