TLS Working Group
Internet Draft
Intended status: Experimental
Expires: January, 2011

Pascal Urien
Telecom ParisTech

July 2010

TLS Tripartite Diffie-Hellman Key Exchange
draft-urien-tls-dh-tripartite-00

Abstract

Most of privacy exchanges over the Internet rely on the TLS protocol. According to this protocol two entities the client and the server computes a master secret from which are deduced cryptographic keys used for data privacy and security. Digital transactions may deal with critical information (payments ...) that need to be recorded for traceability issues or for legal requirements. However messages are secured by the TLS protocol, so it is not possible for a third party that logs packets to perform decryption operations upon legitimate requests.

The goal of this draft is to support a Trusted Third Party (TTP) that could recover the protected information when needed. The proposed protocol uses the Tripartite Diffie-Hellman (tdh) algorithm based on bilinear pairings over elliptic curves.

Requirements Language

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 [RFC2119].

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 January 2011.

Table of Contents

1 Overview

Most of privacy exchanges over the Internet rely on the TLS protocol. According to this protocol two entities the client and the server computes a master secret from which are deduced cryptographic keys used for data privacy and security.

Digital transactions may deal with critical information (payments ...) that need to be recorded for traceability issues or for legal requirements.

However messages are secured by the TLS protocol, so it is not possible to a third party that logs packets to perform decryption operations upon legitimate requests.

The idea of this proposal is to support a Trusted Third Party (TTP) that could recover the protected information when needed.

The Tripartite Diffie-Hellman (tdh) idea was initially introduced in [TDH].

Toady cryptography provides efficient software tools [PBC] computing bilinear pairing over elliptic curve. The [TLS-ECC] standard already supports cryptographic operations based on elliptic curves, mainly for key exchange based on ECDH (Elliptic Curve Diffie Hellman) or the ECDSA (Elliptic Curve Digital Signature Algorithm) signature.

The Weil pairing for example works with classical elliptic curves; server and client are equipped with DH public and private keys.

The classical TLS pre-master-secret is computed according to the relation

pre-master-secret = abP (or g^ab with multiplicative notations)

Where P is a point of an elliptic curve point (a generator) defined over a Fq field, aP and bP are respectively the server and client DH public keys, and a,b their private keys.

If the client and the server agree to use a TTP, identified by an attribute ttp-name and owning a DH public key cP, the pre-master-secret is computing according to the following relation:

Sclient = e(aP,cP)^b
Sserver = e(bP,cP)^a
Sttp = e(aP,bP)^c,

S = Sserver = Sclient = Sttp = shared-secret

S is an element of F(q^k), with k integer

pre-master-secret = h(S), h a hash function producing p bits

In other words each entity (client, server, TTP) computes a shared secret S by using its private DH key and the two other public DH keys.

The master-secret is thereafter produced:

```
master_secret = PRF(pre-master-secret,
                    "master secret",
                     ClientHello.random + ServerHello.random).
```

The master secret is computed by the client and the server without a dialog with the TTP. But the TTP will be able to recover the master secret from the TLS session logs.

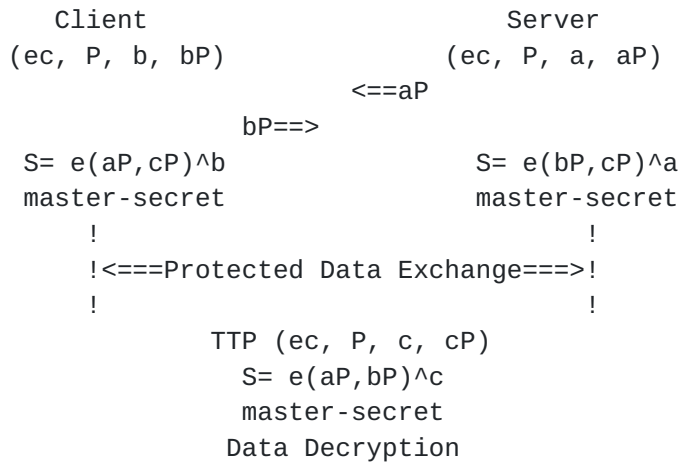

Figure 1. Illustration of TTP use in a TLS session

2 About Bilinear Pairing

2.1 Symmetric Bilinear Pairing

Z is the set of integer.
r is a prime integer

Let (G1,+) be a cyclic group of order r, (with additive operation +)
Let (G2,*) be a cyclic group of order r, (with multiplicative operation x).

Let e a non-degenerate bilinear map, G1 x G1 -> G2

A non-degenerate bilinear pairing satisfies the following conditions:

- Bilinear:
  for P,Q elements of G1, a,b elements of Z, e(aP,bP)=e(P,Q)^ab

- Non-Degenerate:
  e(P,P)#1 is a generator of G2

The Weil pairing [ECC] has been defined for elliptic curve [ECC] over finite field of prime characteristic p (Fp)

- ECC is an elliptic curve over Fp
- G1 is a subgroup in ECC(Fp), whose order is r
- G2 is a subgroup in F(p^k), k an integer

2.2 Asymmetric Bilinear Pairing

In that case the bilinear map works with two different groups G1, whose order is r, and G1', where each element has order dividing r.

e is a non-degenerate bilinear map, G1 x G1' -> G2

For P elements of G1, Q element of G1', a,b elements of Z :
e(aP,bQ)=e(P,Q)^ab

The Tate pairing [ECC] has been defined for elliptic curve [ECC] over finite field of prime characteristic p (Fp).

- ECC is an elliptic curve over Fp
- G1 is a subgroup in ECC(Fp), whose order is r
- G1' is a subgroup in ECC(Fp^k), with k integer (k is the elliptic curve embedding degree)
- G2 is a subgroup in F(p^k)

2.3 The Bilinear Diffie-Hellman (BDH) problem.

P is a generator of G1

a,b,c are elements of Zr*

BDH: Knowing P, aP, bP, cP the computation of e(P,P)^abc is 'impossible'

Furthermore the classical computational Diffie Hellman (CDH) assumption is assumed for the group G1.

CDH: Knowing P, aP, bP the computation of abP is 'impossible'

2.4 Practical properties

2.4.1 Symmetric Pairing

We suppose three entities A (server), B (client), C (ttp) equipped with three pairs of public and private keys, associated with a group

G1 and a generator P.

aP and a, are respectively the public and the private keys for A
bP and b, are respectively the public and the private keys for B
cP and c, are respectively the public and the private keys for C

A shared secret (S) may be computed by these three entities, according to the following scenario:

- A computes S = e^a(bP,cP) = e(P,P)^abc
- B computes S = e^b(aP,cP) = e(P,P)^abc
- C computes S = e^c(aP,bP) = e(P,P)^abc

In other word each entity computes a shared secret key, deduced from its private key and the two other public key.

If A is a client, B a server and C a trusted party, all security properties such as encrypted packets exchanged between client and server may be recovered by the trusted party.

2.4.2 Asymmetric pairing

We suppose three entities A (server), B (client), C (ttp) equipped with pairs of public and private keys, associated with the group G1 (generator P) and the group G1' (generator P').

aP, aP' and a, are respectively the public and the private keys for A
bP and b, are respectively the public and the private key for B
cP, cP' and c, are respectively the public and the private keys for C

A shared secret (S) may be computed by these three entities, according to the following scenario

- A computes S = e^b(bP,cP') = e(P,P')^abc
- B computes S = e^b(aP,cP') = e(P,P')^abc
- C computes S = e^c(aP,bP') = e(P,P')^abc

3. Bilinear pairing operations overview

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

             ClientHello          -------->
   extension {pairing-scheme}
   extension {elliptic-curves}
   extension {ec_point-formats}

                                                        ServerHello
                                        extension {pairing-scheme}
                                       extension {elliptic-curves}
                                      extension {ec_point-formats}

                                                     Certificate*

                                                ServerKeyExchange*
                                   select (KeyExchangeAlgorithm)
                                   { case ec_bilinear-pairing:
                                       ServerECDHParams  params;
                                       Signature  signed_params;
                                       TTP-List         ttp-list
                                   } ServerKeyExchange;

                                              CertificateRequest*+
                                   <--------       ServerHelloDone


            Certificate*+
            ClientKeyExchange
            struct {
              select (KeyExchangeAlgorithm)
                      { case ec_bilinear-pairing:
                        ClientECDiffieHellmanPublic;
                        TTP-Name: ttp-name
                      } exchange-keys;
                    } ClientKeyExchange;

            CertificateVerify*+
            [ChangeCipherSpec]
            Finished             -------->
                                                [ChangeCipherSpec]
                                 <--------              Finished

            Application Data     <------->      Application Data
```

Figure 2. Overview of the TLS bilinear pairing operations.

The TLS dialog is illustrated by figure 2. A new extension is used both by client and server in order to negotiate a pairing scheme. The TTP is selected thanks attributes exchanged in the ServerKeyExchange and ClientKeyExchange messages. Elliptic curve and DH parameter are specified according to [TLS-ECC].

4. Bilinear Pairing Operations details

4.1 TLS Extension

A new TLS extension is defined in this specification: the pairing-scheme extension.

The general structure of TLS extensions is described in [TLS 1.1], and this specification adds one new item to ExtensionType.

```
enum { pairing-scheme(xx)} ExtensionType;
```

The client negotiates the pairing-scheme via the corresponding extension inserted in the client hello message

The server selects one of the proposed items and inserts it in the server hello message.

Two other extensions are imported from [TLS-ECC]:

- elliptic-curves(10),
- ec_point-formats(11);

4.1.1 Supported Pairing Scheme

Supported Pairing Scheme

```
        enum {
              Weil-Pairing (1)
             } NamedScheme;
```

4.2 Key Exchange

A new KeyExchange Algorithm is defined: ec-bilinear-pairing

4.2.1 Trusted-Third-Party list

Each trusted third party (TTP) is identified by a name. A DH public key is implicitly deduced from its identifier.

Structure of this message:

```
opaque TTP-Name<1..2^24-1>;

struct {
        TTP-Name ttp-name-list<0..2^24-1>;
       } TTP-List;
```

4.2.1 Server Key Exchange

The ServerKeyExchange message is extended as follows.

```
        enum { ec-bilinear-pairing } KeyExchangeAlgorithm;

        select (KeyExchangeAlgorithm) {
            case ec_bilinear-pairing:
                ServerECDHParams    params;
                Signature           signed-params;
                TTP-List            ttp-list
        } ServerKeyExchange;
```

The attributes ServerECDHParams and Signature are imported from [TLS-ECC]

4.2.2 Client Key Exchange

The ClientKeyExchange message is extended as follows.

```
        struct {
            select (KeyExchangeAlgorithm) {
                case ec_bilinear-pairing:
                ClientECDiffieHellmanPublic;
                TTP-Name: ttp-name
                } exchange-keys;
        } ClientKeyExchange;
```

The attribute ClientECDiffieHellmanPublic is imported from [TLS-ECC].

4.3 PreMasterSecret computing

aP and bP are respectively the server and the client DH public values (with additive notations)

cP is the trusted third party (ttp) public value, (with additive notations).

h is a digest function producing p bytes.

On the server side:

PreMasterSecret = h(e(bP,cP)^a)

On client side:

PreMasterSecret = h(e(aP,cP)^b)

On the ttp side

PreMasterSecret = h(e(aP,bP)^c)

4.3.1 Hash function for the Weil pairing

For the Weil pairing, G2 is a subgroup of Fq^2, if the output of e is expressed as the tuple (a0,a1) with x and y elements of Fq

h = sha1(a0 || a1), a0 and a1 are values of exactly q bits.

4.4 CipherSuites

To be done.

5. Example of pairings

This example has been computed with the Pairing-Based Cryptography Library [PBC] [BL].

5.1 Symmetric pairing

The elliptic curve (ECC) is chosen as:
y^2 = x^3 + x, with x, y elements of a Field Fq

q is a prime number, with q = 3 modulus 4
G1 is a subgroup of ECC(Fq)
G2 is a subgroup of Fq^2

There are q+1 points on the ECC curve, i.e. #ECC(Fq) = q+1

r = order of G1 = prime factor of q+1.
h = cofactor = #ECC(Fq) / r

q=87807107996633125224377819847540498158068831994142082110286533992664756308802229570786251794226622214231558587695823174592771336731748132492512999822479 1.

h=12016012264891146079388821366740534204802954401251311822919615131047207289359704531102844802183906537786776

r= 730750818665451621361119245571504901405976559617

Generator P=
764213932795790385166146156484628185710738246136731223594607305863197
148904107335230752866953232919510098156557991388877251113225844051396
9390781514106884,
841053126166803064145349163706237513167951671763744795990943027230354
098248702951019958048685849729494818612964151563063433969126677448023
4634049031935396

a = 321231739573260508064943282038854866624801566274

aP =
301901600464207748384853072909292823401500311348346356764859754747703
064669197258898445749044213480292759568257999990157011471277891932048
3502179821202747,
622130912004388536645426112375868421081814378272661695953772580311048
821974186212677634483766383564352093956376894422864585155448402243584
2060112859040085

b = 591069617759232948334516538341133684003963967541

bP =
489872625336582550697622917901310968712983470168986006826993590244144
534124000070208650497397931419729729097601618783404569928023315742260
1733989063260044,
715768020641988338757762409522925702707172363082968385844449919698450
610626853008028585471746136204506517503675925750751323255279155813951
061482849469617

c = 33279005974745643182951119871411467384390110439 5

cP =
361673235446634950587227148388584900737947568189969047494313829915613
165227389313929815513580932571550613336307696480151992816947356553291
3077259306029100,
591927063716469042674124332635453987072498361375371305395310139261168
079986151403557597760872949800149313292665775043657180079452995298 92
7247712148590792

S= f(aP,bP)^c = f(bP,cP)^a= f(cP,aP)^b =
411874021818983935494518378468671897765968395058497750231537284872046
634607045530820852050427874577949368791925702115715036551400857093954
2202145179250626,
792272713616191570585463230096947246676587164343554480144379956667402
825333387256294430771484207666103108581680992251414596583914923441539
9681428439428268

6. Security Considerations

To be done.

7. IANA Considerations

To be done

8 References

8.1 Normative references

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[TLS 1.0] Dierks, T., C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999

[TLS 1.1] Dierks, T., Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006

[TLS 1.2] Dierks, T., Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.1", draft-ietf-tls-rfc4346-bis-10.txt, March 2008

[TLS-ECC] 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.

8.2 Informative references

[ECC] Joseph H.Silverman, "The Arithmetic of Elliptic Curves", Second Edition, Springer, 2008

[PBC] http://crypto.stanford.edu/pbc/

[TDH] A.Joux, "A one round protocol for tripartite Diffie-Hellman", Lecture Notes in Computer Science, Springer, Volume 1838, 2000

[BL] Benn Lynn, "On the implementation of paring-based cryptosystems", PHD dissertation, Stanford University, 2007

Author's Addresses

Pascal Urien
Telecom ParisTech
37/39 rue Dareau, 75014 Paris, France

Email: Pascal.Urien@telecom-paristech.fr