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.
Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.
Table of Contents

Abstract........................................................... 1
Requirements Language.............................................. 1
Status of this Memo................................................ 1
Copyright Notice................................................... 2
Table of Contents.................................................. 3

1 Overview........................................................ 4
2 About Bilinear Pairing........................................... 5
   2.1 Symmetric Bilinear Pairing................................ 5
   2.2 Asymmetric Bilinear Pairing.............................. 6
   2.3 The Bilinear Diffie-Hellman (BDH) problem.............. 6
   2.4 Practical properties...................................... 6
       2.4.1 Symmetric Pairing ............................... 6
       2.4.2 Asymmetric pairing .............................. 7
3. Bilinear pairing operations overview......................... 8

4. Bilinear Pairing Operations details............................ 9
   4.1 TLS Extension............................................ 9
       4.1.1 Supported Pairing Scheme ........................ 9
   4.2 Key Exchange............................................. 9
       4.2.1 Trusted-Third-Party list ........................ 9
       4.2.1 Server Key Exchange ............................. 10
       4.2.2 Client Key Exchange ............................. 10
   4.3 PreMasterSecret computing................................ 10
       4.3.1 Hash function for the Weil pairing............... 11
   4.4 CipherSuites.............................................. 11

5. Example of pairings............................................. 11
   5.1 Symmetric pairing....................................... 11

6. Security Considerations........................................ 12
7. IANA Considerations........................................... 13
8. References.................................................................. 13
   8.1 Normative references..................................... 13
   8.2 Informative references................................... 13

Author’s Addresses................................................ 13
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].

Today 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

\[ \text{pre-master-secret} = abP \text{ (or } g^{ab} \text{ with multiplicative notations)} \]

Where \( P \) is a point of an elliptic curve point (a generator) defined over a \( F_q \) 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:

\[ S_{\text{client}} = e(aP,cP)^b \]
\[ S_{\text{server}} = e(bP,cP)^a \]
\[ S_{\text{ttp}} = e(aP,bP)^c, \]

\[ S = S_{\text{server}} = S_{\text{client}} = S_{\text{tp}} = \text{shared-secret} \]

\( S \) 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.

\[
\begin{align*}
\text{Client} & : (e, P, b, bP) \\
\text{Server} & : (e, P, a, aP) \\
\text{TTP} & : (e, P, c, cP)
\end{align*}
\]

\[
\begin{align*}
bP & \Rightarrow aP \\
S & = e(aP, cP)^b \\
master-secret & \\
TTP & = e(aP, bP)^c \\
\text{Data Decryption}
\end{align*}
\]

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 \((G2,\ast)\) be a cyclic group of order r, (with multiplicative operation \(\ast\)).

Let e a non-degenerate bilinear map, \(G1 \times G1 \rightarrow G2\)

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:
  \( e(P,P) \neq 1 \) is a generator of \( G_2 \)

The Weil pairing \([\text{ECC}]\) has been defined for elliptic curve \([\text{ECC}]\) over finite field of prime characteristic \( p \) (\( \mathbb{F}_p \))

- \( \text{ECC} \) is an elliptic curve over \( \mathbb{F}_p \)
- \( G_1 \) is a subgroup in \( \text{ECC}(\mathbb{F}_p) \), whose order is \( r \)
- \( G_2 \) is a subgroup in \( \mathbb{F}(p^k) \), \( k \) an integer

2.2 Asymmetric Bilinear Pairing

In that case the bilinear map works with two different groups \( G_1 \), whose order is \( r \), and \( G_1' \), where each element has order dividing \( r \).

\( e \) is a non-degenerate bilinear map, \( G_1 \times G_1' \rightarrow G_2 \)

For \( P \) elements of \( G_1 \), \( Q \) element of \( G_1' \), \( a,b \) elements of \( \mathbb{Z} \):
\[
e(aP,bQ)=e(P,Q)^{ab}
\]

The Tate pairing \([\text{ECC}]\) has been defined for elliptic curve \([\text{ECC}]\) over finite field of prime characteristic \( p \) (\( \mathbb{F}_p \)).

- \( \text{ECC} \) is an elliptic curve over \( \mathbb{F}_p \)
- \( G_1 \) is a subgroup in \( \text{ECC}(\mathbb{F}_p) \), whose order is \( r \)
- \( G_1' \) is a subgroup in \( \text{ECC}(\mathbb{F}_p^k) \), with \( k \) integer (\( k \) is the elliptic curve embedding degree)
- \( G_2 \) is a subgroup in \( \mathbb{F}(p^k) \)

2.3 The Bilinear Diffie-Hellman (BDH) problem.

\( P \) is a generator of \( G_1 \)

\( a,b,c \) are elements of \( \mathbb{Z}_r^* \)

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 \( G_1 \).

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 \( G_1 \) 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 -------->                        ServerHello
extension {pairing-scheme}                   extension {pairing-scheme}
extension {elliptic-curves}                 extension {elliptic-curves}
extension {ec_point-formats}                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.

```c
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

```c
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=8780710799663312522437781984754049815806883199414208211028653399266
47563080222957078625179422662221423155858769582317459277713367317481
324925129998224791.

h=1201601226489114607938882136674053420480295440125131182291961513104
7207289359704531102844802183906537786776

r= 73075081866545162136119245571504901405976559617
Generator $P = 764213932795790385166146156484628185710738246136731223594607305863197$
$14890410735230752866953232919510098156557991388877251113225844051396$
$930781514160884, 841053126166803064145349163706237513167951671763744795990943027230354$
$098248702951019958048685849792949481861296415156363433969126677448023$
$4634049031935396$

$a = 321231739573260508064943282038854866624801566274$

$aP = 3019016004642077483848530729092928234015003113483463567648597547703$
$064669197258898445749044213480292759568257999990157011471277891932048$
$3502179821202747, 6221309120043855664454261123758684210811437827266169595372580311048$
$821974186212677634483766383564352093956376894422864585155448402243584$
$2060112859040085$

$b = 591069617759232948334516538341133684003963967541$

$bP = 489872625336582550697622917901310968712983470168986006826993590244144$
$534124000070208650497397931419729729097601618783404569928023315742260$
$1733989063260044, 7157680220641988338757762409522925702707172363082968385844449919698450$
$61062685300802858547174613620450651750367592575075132325279155813951$
$061482849469617$

$c = 332790059747456431829511198714114673843901104395$

$cP = 361673235446634950587227148388584900737947568189969047494313829915613$
$16522738931392981551358093257155061333630769648015199281694735653291$
$307725930036029100, 591927063716469042674124332635453987072498361375371305395310139261168$
$079986151403555797760872949800149313292665777504365718007945299529892$
$72477124859792$

$S= f(aP,bP)^c = f(bP,cP)^a= f(cP,aP)^b = 411874021818983935494518378468671897765968395058497750231537284872046$
$634607045530820852050427874577949368791925702115715036551400857093954$
$2202145179250626, 792272713616191570585463230096947246676587164343554480144379795667402$
$82533338725629430771484207666103108581680992251415496583914923441539$
$9681428439428268$

6. Security Considerations

To be done.
7. IANA Considerations

To be done

8 References

8.1 Normative references


8.2 Informative references


Author’s Addresses

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

Email: Pascal.Urien@telecom-paristech.fr