Network Working Group
Internet-Draft
Intended Status: Informational
Expires: December 11, 2016

A. Budroni, K. McCusker
MIRACL Ltd.
June 9, 2016

Milagro TLS: Pairing-Based Cryptography for Transport Layer Security
draft-budronimccusker-milagrotls-02

Abstract

This document introduces two key exchange algorithms based on Pairing-Based Cryptography (PBC) for the Transport Layer Security (TLS) protocol. In particular, it specifies the use of two identity-based key exchange algorithms for the TLS handshake.

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 December 11, 2016.

Copyright Notice

Copyright (c) 2016 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. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Pairing-Based Crypto (PBC) is emerging as a solution to complex problems that proved intractable to the standard mathematics of Public-Key Cryptography. An example of such a problem would be Identity-Based Encryption, whereby the identity of a client can be used as their public key [11].

PBC is based on the use of a bi-linear map defined on an elliptic curve E

$$e: G1 \times G2 \rightarrow GT$$

where G1 is defined as a group of points on E, G2 is defined as a group of points on a twist of E over an extension field. Both groups are of prime order q. GT is a finite extension.

Milagro TLS proposes the use of PBC for mutually authenticated key agreement. There are two new key exchange algorithms in this draft: Peer-to-Peer (P2P) and Client-Server. The P2P solution uses the Chow-Choo protocol and the Client-Server solution uses the MPIN Protocol [9,10].

Milagro TLS uses a curve that has security at the AES-128 level.

This document describes an addition to TLS 1.2 [1] to support PBC. In particular, it defines

o Milagro_CS: a key exchange algorithm based on MPIN-FULL protocol [9]. This is a Client-to-Server protocol that allows mutually authenticated key agreement. In this protocol the client secrets are in G1 and the server secret is in G2. For a Type-3 pairing there is assumed to be no computable isomorphism between these groups, even though both are of the same order.

o Milagro_P2P: a key exchange algorithm based on the Chow-Choo protocol [10]. It can operate in P2P or client/server mode. Users of this protocol are issued sender keys in G1 and receiver keys in G2. The server, which sends the ServerKeyExchange message, is considered the sender in this protocol.

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 [2].

2.1 Definitions

Digital Identity: Digital Identity is the data that uniquely describes a person or a thing, and typically contains some information about that entity's relationships.

2.2 Abbreviations

ECC Elliptic Curve Cryptography

PBC Pairing-Based Cryptography

AES Advanced Encryption Standard

TA Trusted Authority

P2P Peer-to-Peer

Milagro_CS Milagro Client-to-Server

Milagro_P2P Milagro Peer-to-Peer

ECDH Elliptic Curve Diffie Hellman

E is an ordinary pairing-friendly elliptic curve over a finite field F, defined by a fixed prime modulus p

2.3 Conventions

IdC: Digital identity of the client

IdS: Digital identity of the server

H1: Maps string value to a point on the curve in G1.

H2: Maps string value to a point on the curve in G2

Hq: Hashes inputs to an integer modulo the curve order q.

Hg: Generate AES key

SHA-256: Performs the SHA256 function.

3. Key Exchange Algorithms

3.1 MILAGRO_CS

Here we briefly resume the main steps of the MPIN-FULL key exchange algorithm, see [8] and [9] for details.

Let A = H1(IdC) be a point on G1, where IdC is the client's identity, and let Q be a generator of the group G2. The TA provides the client key s.A, in G1 and the server key s.Q, in G2.

MPIN Full was envisaged as a two factor authentication solution but in this context, as this is a machine to machine protocol, there is no requirement for a second factor and therefore the PIN is set to zero in the code.

The ClientHello message MUST have an extension which contains three public parameters:

- IdC, the identity of the client. This can be the identity in clear or the hash of identity. In the latter case the IdC is encrypted after the session key is established and sent to the server to complete client authentication.

- U = x.(H1(IdC)) where x is a random number modulo the curve order.

- t, the epoch time at which authentication occurred.

- V = -(x+y)(s.A), where y = Hq(t|U) and A = H1(IdC).

The server itself calculates A by applying the same hash function H1 to the claimed digital identity i.e. A is H1(IdC). Then the Server MUST check that e(V,Q).e(U+yA,s.Q) = 1. If this tests fails then the connection is terminated by the server with a proper alert message and the attempted Client connection is rejected.

Through the ServerKeyExchange message, the server sends an ECDH public key W=w.A, where w is a random number modulo the curve order and A is H1(IdC).

Through the ClientKeyExchange message, the client send its ECDH public key R=r.A, where r is a random number modulo the curve order.

At this point, both the client and the server are able to compute a 16-bytes shared premaster secret:

- The client first computes the parameter h = Hq(A,U,y,V,R,W), then computes the premaster secret as K = Hg(e(s.A,Q)^(r+h)|x.W).

- The server first computes the parameter h = Hq(A,U,y,V,R,W), then computes the premaster secret as K = Hg(e(R+h.A,s.Q)|w.U).

  See [9] for more details.

3.2 MILAGRO_P2P

Here we briefly resume the main steps of the Chow-Choo key exchange Algorithm [10].

Choo-Chow key exchange algorithm is designed for communications peer-to-peer. The TA provides the server with a sender key in G1 i.e. SKeyG1 and client with receiver key in G2 I.e. CKeyG2 based on their respective identities.

The main steps of the algorithm are:

- The server computes a random integer x modulo the curve order, computes a point on the group G1 as PsG1 = x.H1(IdS) which is its public parameter and sends the pair (IdS,PsG1) to the client.

- The client receives the pair of parameter from the server, computes two random integers Y and W modulo the curve order, compute the following; PcG2 = Y.H2(IdC), PgG1 = W.H1(IdS), pic = Hq(PcG2||PsG1||PgG1||IdS), pis = Hq(PsG1||PcG2||PgG1||IdC) and the value k = e(pis.H(IdS)+PsG1,(y+pic).CKeyG2).

- client computes the premaster secret as K = Hg(k,w.PsG1).

- client sends the triple (IdC,PsG1,PgG1) to server.

- server receives the parameters from the client and computes the following
  pis = Hq(PsG1||PcG2||PgG1), pic = Hq(PcG2||PsG1||PgG1) and the
  value k = e((x+pis).SKeyG1,pic.B+PcG2).

- server compute the premaster secret as K = Hg(k,x.PsG1).

4. Data Structures and Computations

This document introduces two new key exchange algorithms for TLS that use PBC to compute the TLS premaster secret. The derivation of the TLS master secret from the premaster secret and the subsequent generation of bulk encryption/MAC keys and initialization vectors is independent of the key exchange algorithm and not impacted by the introduction of PBC and ECC.

```
enum {
      Milagro_CS,
      Milagro_P2P,
} KeyExchangeAlgorithm;
```

The first key exchange algorithm is Milagro_CS and it is designed for client-to-server communications. It is based on the MPIN-FULL key exchange protocol [9], which is an extension of the M-Pin Authentication Protocol [8].

The second key exchange algorithm is Milagro_P2P and it is designed for peer-to-peer communications. It is based on the CHOW-CHOO protocol [10].

Here we summarize the steps of TLS-Handshake used by those two key exchange algorithms.

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

     ClientHello           --------->
                                                       ServerHello
                                                   ServerKeyExchange
                           <---------                ServerHelloDone
   ClientKeyExchange
  (ChangeCipherSpec)
      Finished             --------->
                                                  (ChangeCipherSpec)
                           <---------                  Finished
```

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

The following messages of TLS-Handshake MUST NOT be sent for those two key exchange algorithms: (Server)Certificate, CertificateRequest, (Client)Certificate and CertificateVerify.

4.1 ClientHello Extension

This section specifies a TLS extension that can be included with the ClientHello message as described in [3], the Milagro_CS Extension.

When this extension are sent:

The extension MUST be sent along with any ClientHello message that proposes Milagro_CS key exchange algorithms and it MUST NOT be sent with any other ClientHello message that doesn't proposes this cipher.

Meaning of this extension:

This extension allow the Client to authenticate itself with the Server and to exchange part of the parameters that will be used to compute the premaster secret.

Structure of these extensions:

As described in [3], two octets of are used to indicate the extension type. In case of Milagro_CS extension the octets are 0x0025. The general structure of TLS extensions is described in [3], and this specification adds a new type to ExtensionType.

```
          enum {  Milagro_CS_ext } ExtensionType;

               struct {
                       uint16 length_hash_IdC,
                       uint16 length_U,
                       uint16 length_V,
                       opaque hash_IdC[length_hash_IdC],
                       opaque U[length_U],
                       opaque V[length_V],
                       uint32 time_value,
                       (255)
               } Milagro_CS_ext;
```

length_hash_IdC, length_U, length_V: length of the parameters.

hash_IdC: hash of the client's identity.

U: first parameter sent by the client.

V: second parameter sent by the client.

time_value: current epoch time in seconds.

Actions of the Server:

If Milagro_CS is between the key exchange algorithms available of the server, then he MUST check if the time_value received from the client differs too much from the current time. If the difference is more than a fixed value then he has to refuse the client. If this check has a successful ending it is RECOMMENDED, regardless of the chosen cipher suite, that he tries to authenticate the Client as explained in 3.1, and, in case of failing, he has to refuse the client.

See [8] for details about the authentication.

4.2 Server Key Exchange

This document introduces two new ServerKeyExchange messages, one for each key exchange algorithm.

If the cipher suite chosen by the server has Milagro_CS as key exchange algorithm, then the server MUST compute the parameter W, as explained in 3.1 and send it.

If the cipher suite chosen by the server has Milagro_P2P as key exchange algorithm, then the server MUST compute the the public parameter PsG1 as explained in 3.2 and send it with its digital identity IdS.

The ServerKeyExchange message is extended as follows.

```
select (KeyExchangeAlgorithm) {
     case Milagro_CS:
          uint16   length_W;
          opaque   W[length_W];
     case Milagro_P2P:
          uint16   length_IdS;
          uint16   length_PsG1;
          opaque   IdS[length_IdS];
          opaque   PsG1[length_PsG1];
} ServerKeyExchange;
```

4.3 Client Key Exchange

If the cipher suite chosen by the server has Milagro_CS as key exchange algorithm, then the client MUST compute the parameter R, as explained in 3.1 and send it.

If the cipher suite chosen by the server has Milagro_P2P as key exchange algorithm, then the client MUST compute the parameters PgG1 and PcG2 as explained in 3.2 and send them with its digital identity IdC.

The ClientKeyExchange message is extended as follows.

```
select (KeyExchangeAlgorithm) {
     case Milagro_CS:
          uint16   length_R;
          opaque   R[length_R];
     case Milagro_P2P:
          uint16   length_IdC;
          uint16   length_PgG1;
          uint16   length_PcG2;
          opaque   IdC[length_IdC];
          opaque   PgG1[length_PgG1];
          opaque   PcG2[length_PcG2];
} ClientKeyExchange;
```

5. Cipher Suites

The table below defines new cipher suites that use the key exchange algorithms specified in Section 3.

```
CipherSuite TLS_MILAGRO_CS_WITH_AES_128_GCM_SHA256       = {0xC0,0xB1}
CipherSuite TLS_MILAGRO_CS_WITH_AES_128_GCM_SHA512       = {0xC0,0xB2}
CipherSuite TLS_MILAGRO_CS_WITH_CAMELLIA_128_GCM_SHA256  = {0xC0,0xB3}
CipherSuite TLS_MILAGRO_CS_WITH_CAMELLIA_128_GCM_SHA512  = {0xC0,0xB4}
CipherSuite TLS_MILAGRO_CS_WITH_3DES_EDE_CBC_SHA256      = {0xC0,0xB5}
CipherSuite TLS_MILAGRO_CS_WITH_3DES_EDE_CBC_SHA512      = {0xC0,0xB6}

CipherSuite TLS_MILAGRO_P2P_WITH_AES_128_GCM_SHA256      = {0xC0,0xB7}
CipherSuite TLS_MILAGRO_P2P_WITH_AES_128_GCM_SHA512      = {0xC0,0xB8}
CipherSuite TLS_MILAGRO_P2P_WITH_CAMELLIA_128_GCM_SHA256 = {0xC0,0xB9}
CipherSuite TLS_MILAGRO_P2P_WITH_CAMELLIA_128_GCM_SHA512 = {0xC0,0xC0}
CipherSuite TLS_MILAGRO_P2P_WITH_3DES_EDE_CBC_SHA256     = {0xC0,0xC1}
CipherSuite TLS_MILAGRO_P2P_WITH_3DES_EDE_CBC_SHA512     = {0xC0,0xC2}
```

The key exchange method, cipher, and hash algorithm for each of these cipher suites are easily determined by examining the name. Ciphers (other than AES ciphers) and hash algorithms are defined in [1]. AES cipher is defined in [5], GCM in [6] and the hash algorithm is defined in [7].

The cipher suite name space is maintained by IANA. See Section 7 for information on how new value assignments are added.

6. Security Considerations

For TLS handshakes using PBC cipher suites, the security considerations in appendices D, E and F of [1] apply accordingly.

Security discussion specific to PBC can be also found in [11].

Implementers and users must also consider whether they need forward secrecy. Forward secrecy refers to the property that session keys are not compromised if the static, certified keys belonging to the server and client are compromised. The MILAGRO_CS and MILAGRO_P2P key exchange algorithms provide forward secrecy protection in the event of server and/or client's secret compromise.

6.1 MILAGRO_CS

A replay-attack might be mounted by re-sending the parameters sent with the extension of ClientHello from a previous conversation. This will not be successful if the difference between the current time on the server and time parameter in the ClientHello message is too large.

An active attacker might allow the server to complete the first part of the protocol and then attempt to hijack the link before the calculation of the key. But observe how the value of x is re-used for the calculation of the Diffie-Hellman component of the key. This binds both parts of the protocol together and effectively blocks any hijacking attempt.

A Key Compromise Impersonation (KCI) attack, whereby an attacker steals the client credentials and poses as a valid server, is impossible to mount due to fact that random integer r is used in the key agreement protocol.

6.1 MILAGRO_P2P

This key exchange algorithm has been proved secure under the Bilinear-Diffie-Hellman (BDH) assumption in the Canetti-Krawczyk [10].

Other security discussions about MILAGRO_P2P key exchange algorithm can be found in [10].

7. IANA Considerations

This document introduces in section 4.1 and 5 some additions to Transport Layer Security (TLS) Parameters.

Any assignments in this document require IETF Consensus action [4].

8. References

8.1 Normative References

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

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

[3] D. Eastlake, "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

[4] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, May 2008.

8.2 Informative References

[5] National Institute of Standards and Technology, "Specification for the Advanced Encryption Standard (AES)", FIPS 197, November 2001.

[6] National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) for Confidentiality and Authentication", SP 800-38D, November 2007.

[7] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-4, August 2015.

[8] Scott, M. "M-Pin: A Multi-Factor Zero Knowledge Authentication Protocol", Miracl Labs.

[9] Scott, M. "M-Pin Full Technology (Version 3.0)", Miracl Labs.

[10] Sherman S.M. Chow and Kim-Kwang Raymond Choo, "Strongly-Secure Identity-based Key Agreement and Anonymous Extension", Cryptology ePrint Archive, Report 2007/018,2007.

[11] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. SIAM Journal of Computing, 32(3):586-615, 2003.

Authors' Addresses

Alessandro Budroni
MIRACL
Email: alessandro.budroni@miracl.com

Kealan McCusker
MIRACL
Email: kealan.mccusker@miracl.com