                    Curve25519 and Curve448 for the
     Internet Key Exchange Protocol Version 2 (IKEv2) Key Agreement

Abstract

   This document describes the use of Curve25519 and Curve448 for
   ephemeral key exchange in the Internet Key Exchange Protocol Version
   2 (IKEv2).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   The "Elliptic Curves for Security" document [RFC7748] describes two
   elliptic curves, Curve25519 and Curve448, as well as the X25519 and
   X448 functions for performing key agreement using Diffie-Hellman
   operations with these curves. The curves and functions are designed
   for both performance and security.

   Elliptic curve Diffie-Hellman [RFC5903] has been specified for the
   Internet Key Exchange Protocol Version 2 (IKEv2) [RFC7296] for almost
   ten years. RFC 5903 and its predecessor specified the so-called NIST
   curves. The state of the art has advanced since then. More modern
   curves allow faster implementations while making it much easier to
   write constant-time implementations that are resilient to time-based
   side-channel attacks. This document defines two such curves for use
   in IKEv2. See [Curve25519] for details about the speed and security
   of the Curve25519 function.

1.1. Conventions Used in This Document

   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].

2. Curve25519 and Curve448

   Implementations of Curve25519 and Curve448 in IKEv2 SHALL follow the
   steps described in this section. All cryptographic computations are
   done using the X25519 and X448 functions defined in [RFC7748]. All
   related parameters (for example, the base point) and the encoding (in
   particular, pruning the least/most significant bits and using little-
   endian encoding) are compliant with [RFC7748].

   An ephemeral Diffie-Hellman key exchange using Curve25519 or Curve448
   is performed as follows: each party picks a secret key d uniformly at
   random and computes the corresponding public key. "X" is used below
   to denote either X25519 or X448, and "G" is used to denote the
   corresponding base point:

      pub_mine = X(d, G)

   Parties exchange their public keys (see Section 3.1) and compute a
   shared secret:

         SHARED_SECRET = X(d, pub_peer)

   This shared secret is used directly as the value denoted g^ir in
   Section 2.14 of RFC 7296. It is 32 octets when Curve25519 is used
   and 56 octets when Curve448 is used.

3. Use and Negotiation in IKEv2

   The use of Curve25519 and Curve448 in IKEv2 is negotiated using a
   Transform Type 4 (Diffie-Hellman group) in the Security Association
   (SA) payload of either an IKE_SA_INIT or a CREATE_CHILD_SA exchange.
   The value 31 is used for the group defined by Curve25519 and the
   value 32 is used for the group defined by Curve448.

3.1. Key Exchange Payload

   The diagram for the Key Exchange payload from Section 3.4 of RFC 7296
   is copied below for convenience:

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Next Payload  |C|  RESERVED   |         Payload Length        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Diffie-Hellman Group Num    |           RESERVED            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ~                       Key Exchange Data                       ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
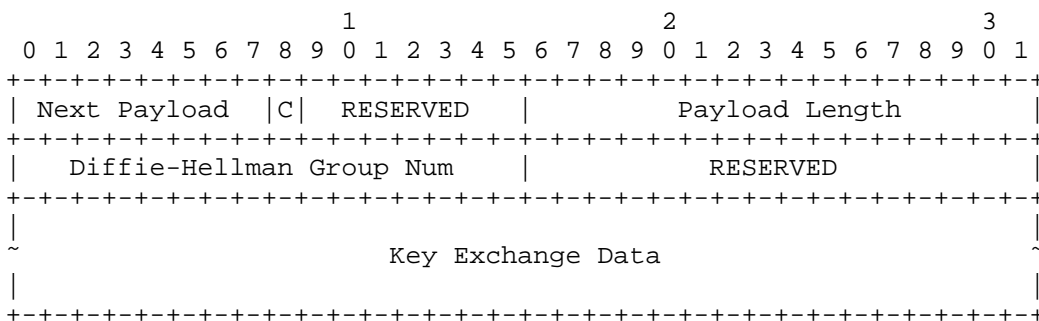
   o Payload Length - For Curve25519, the public key is 32 octets, so
      the Payload Length field will be 40. For Curve448, the public key
      is 56 octets, so the Payload Length field will be 64.

   o The Diffie-Hellman Group Num is 31 for Curve25519 or 32 for
      Curve448.

   o The Key Exchange Data is the 32 or 56 octets as described in
      Section 6 of [RFC7748].

3.2. Recipient Tests

   Receiving and handling of incompatible point formats MUST follow the
   considerations described in Section 5 of [RFC7748]. In particular,
   receiving entities MUST mask the most-significant bit in the final
   byte for X25519 (but not X448), and implementations MUST accept non-
   canonical values.

4. Security Considerations

   Curve25519 and Curve448 are designed to facilitate the production of
   high-performance constant-time implementations. Implementors are
   encouraged to use a constant-time implementation of the functions.
   This point is of crucial importance, especially if the implementation
   chooses to reuse its ephemeral key pair in many key exchanges for
   performance reasons.

   Curve25519 is intended for the ~128-bit security level, comparable to
   the 256-bit random ECP Groups (group 19) defined in RFC 5903, also
   known as NIST P-256 or secp256r1. Curve448 is intended for the
   ~224-bit security level.

While the NIST curves are advertised as being chosen verifiably at
random, there is no explanation for the seeds used to generate them.
In contrast, the process used to pick Curve25519 and Curve448 is
fully documented and rigid enough so that independent verification
can and has been done. This is widely seen as a security advantage
because it prevents the generating party from maliciously
manipulating the parameters.

Another family of curves available in IKE that were generated in a
fully verifiable way is the Brainpool curves [RFC6954]. For example,
brainpoolP256 (group 28) is expected to provide a level of security
comparable to Curve25519 and NIST P-256. However, due to the use of
pseudorandom prime, it is significantly slower than NIST P-256, which
is itself slower than Curve25519.

5. IANA Considerations

   IANA has assigned two values for the names "Curve25519" and
   "Curve448" in the IKEv2 "Transform Type 4 - Diffie-Hellman Group
   Transform IDs" and has listed this document as the reference. The
   Recipient Tests field should also point to this document:

```
        +--------+------------+----------------------+-----------+
        | Number |    Name    |   Recipient Tests    | Reference |
        +--------+------------+----------------------+-----------+
        |   31   | Curve25519 | RFC 8031, Section 3.2 |  RFC 8031 |
        |   32   |  Curve448  | RFC 8031, Section 3.2 |  RFC 8031 |
        +--------+------------+----------------------+-----------+
```

                   Table 1: New Transform Type 4 Values

6. References

6.1. Normative References

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119,
              DOI 10.17487/RFC2119, March 1997,
              <http://www.rfc-editor.org/info/rfc2119>.

   [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T.
              Kivinen, "Internet Key Exchange Protocol Version 2
              (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October
              2014, <http://www.rfc-editor.org/info/rfc7296>.

   [RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves
              for Security", RFC 7748, DOI 10.17487/RFC7748, January
              2016, <http://www.rfc-editor.org/info/rfc7748>.

6.2. Informative References

   [Curve25519]
             Bernstein, J., "Curve25519: New Diffie-Hellman Speed
             Records", Public Key Cryptography - PKC 2006, Lecture
             Notes in Computer Science (LNCS), Vol. 3958, pp. 207-228,
             DOI 10.1007/11745853_14, February 2006,
             <http://dx.doi.org/10.1007/11745853_14>.

   [RFC5903] Fu, D. and J. Solinas, "Elliptic Curve Groups modulo a
             Prime (ECP Groups) for IKE and IKEv2", RFC 5903,
             DOI 10.17487/RFC5903, June 2010,
             <http://www.rfc-editor.org/info/rfc5903>.

   [RFC6954] Merkle, J. and M. Lochter, "Using the Elliptic Curve
             Cryptography (ECC) Brainpool Curves for the Internet Key
             Exchange Protocol Version 2 (IKEv2)", RFC 6954,
             DOI 10.17487/RFC6954, July 2013,
             <http://www.rfc-editor.org/info/rfc6954>.

Appendix A. Numerical Example for Curve25519

   Suppose we have both the initiator and the responder generating
   private keys by generating 32 random octets. As usual in IKEv2 and
   its extension, we will denote Initiator values with the suffix _i and
   responder values with the suffix _r:

      random_i = 75 1f b4 30 86 55 b4 76 b6 78 9b 73 25 f9 ea 8c
                 dd d1 6a 58 53 3f f6 d9 e6 00 09 46 4a 5f 9d 94

      random_r = 0a 54 64 52 53 29 0d 60 dd ad d0 e0 30 ba cd 9e
                 55 01 ef dc 22 07 55 a1 e9 78 f1 b8 39 a0 56 88

   These numbers need to be fixed by unsetting some bits as described in
   Section 5 of RFC 7748. This affects only the first and last octets
   of each value:

      fixed_i = 70 1f b4 30 86 55 b4 76 b6 78 9b 73 25 f9 ea 8c
                 dd d1 6a 58 53 3f f6 d9 e6 00 09 46 4a 5f 9d 54

      fixed_r = 08 54 64 52 53 29 0d 60 dd ad d0 e0 30 ba cd 9e
                 55 01 ef dc 22 07 55 a1 e9 78 f1 b8 39 a0 56 48

   The actual private keys are considered to be encoded in little-endian
   format:

   d_i = 549D5F4A460900E6D9F63F53586AD1DD8CEAF925739B78B676B4558630B41F70

   d_r = 4856A039B8F178E9A1550722DCEF01559ECDBA30E0D0ADDD600D295352645408

   The public keys are generated from this using the formula in
   Section 2:

   pub_i = X25519(d_i, G) =
                 48 d5 dd d4 06 12 57 ba 16 6f a3 f9 bb db 74 f1
                 a4 e8 1c 08 93 84 fa 77 f7 90 70 9f 0d fb c7 66

   pub_r = X25519(d_r, G) =
                 0b e7 c1 f5 aa d8 7d 7e 44 86 62 67 32 98 a4 43
                 47 8b 85 97 45 17 9e af 56 4c 79 c0 ef 6e ee 25

   And this is the value of the Key Exchange Data field in the Key
   Exchange payload described in Section 3.1. The shared value is
   calculated as in Section 2:

   SHARED_SECRET = X25519(d_i, pub_r) = X25519(d_r, pub_i) =
                 c7 49 50 60 7a 12 32 7f-32 04 d9 4b 68 25 bf b0
                 68 b7 f8 31 9a 9e 37 08-ed 3d 43 ce 81 30 c9 50

Authors' Addresses

   Yoav Nir
   Check Point Software Technologies Ltd.
   5 Hasolelim st.
   Tel Aviv 6789735
   Israel

   Email: ynir.ietf@gmail.com

   Simon Josefsson
   SJD AB

   Email: simon@josefsson.org