ECC in OpenPGP
draft-jivsov-openpgp-ecc-14.txt

Abstract

   This document defines an Elliptic Curve Cryptography extension to
   the OpenPGP public key format and specifies three Elliptic Curves
   that enjoy broad support by other standards, including standards
   published by the US National Institute of Standards and Technology.
   The document specifies the conventions for interoperability between
   compliant OpenPGP implementations that make use of this extension.

Status of this Memo

   This Internet-Draft is submitted to IETF in full conformance with
   the provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF), its areas, and its working groups. Note that
   other groups may also distribute working documents as
   Internet-Drafts.

   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."

   The list of current Internet-Drafts can be accessed at
   http://www.ietf.org/ietf/1id-abstracts.txt.

   The list of Internet-Draft Shadow Directories can be accessed at
   http://www.ietf.org/shadow.html.

   This Internet-Draft will expire on October 8, 2012.

Copyright Notice

Table of Contents

1. Introduction

   The OpenPGP protocol [RFC4880] supports RSA and DSA public key
   formats. This document defines the extension to incorporate
   support for public keys that are based on Elliptic Curve
   Cryptography (ECC).

2. 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].

Any implementation that adheres to the format and methods specified
in this document is called a compliant application. Compliant
applications is a subset of the the broader set of [RFC4880]
OpenPGP applications. Any [RFC2119] keyword within this document
applies to compliant applications only.

3. Elliptic Curve Cryptography

This document establishes the minimum set of Elliptic Curve
Cryptography (ECC) public key parameters and cryptographic methods
that will likely satisfy the widest range of platforms and
applications and facilitate interoperability. It adds a more
efficient method for applications to balance the overall level of
security with any AES algorithm specified in [RFC4880] than by
simply increasing the size of RSA keys.

This document defines a path to expand ECC support in the future.

National Security Agency (NSA) of the United States specifies ECC
for use in its [Suite B] set of algorithms. This document includes
algorithms required by Suite B that are not present in [RFC4880].

[KOBLITZ] provides a thorough introduction to ECC.

4. Supported ECC curves

This document references three named prime field curves, defined in
[FIPS 186-3] as "Curve P-256", "Curve P-384", and "Curve P-521".

The named curves are referenced as a sequence of bytes in this
document, called throughout this document as curve OID. Section 11
describes in details how this sequence of bytes is formed.

5. Supported public key algorithms

The supported public key algorithms are Elliptic Curve Digital
Signature Algorithm (ECDSA) [FIPS 186-3] and Elliptic Curve Diffie-
Hellman (ECDH). A compatible specification of ECDSA is given in
[RFC6090] as "KT-I Signatures" and in [SEC1]; ECDH is defined in
section 8.

The following public key algorithm IDs are added to expand the
section 9.1. Public-Key Algorithms of [RFC4880]:

ID	Description of algorithm
[to be ASSIGNED] presumably 18	ECDH public key algorithm
19	ECDSA public key algorithm

Compliant applications MUST support ECDSA and ECDH.

6. Conversion primitives

   This document only defines the uncompressed point format. The
   point is encoded in the Multiprecision Integer (MPI) format
   [RFC4880]. The content of the MPI is the following:

      B = 04 || x || y

   where x and y are coordinates of the point P = (x, y), each encoded
   in the big endian format and zero-padded to the adjusted underlying
   field size. The adjusted underlying field size is the underlying
   field size that is rounded up to the nearest 8-bit boundary.

   Therefore, the exact size of the MPI payload is 515 bits for "Curve
   P-256", 771 for "Curve P-384", and 1059 for "Curve P-521".

   Even though the zero point, also called the point at infinity, may
   occur as a result of arithmetic operations on points of an elliptic
   curve, it SHALL NOT appear in data structures defined in this
   document.

   This encoding is compatible with the definition given in [SEC1].

   If other conversion methods are defined in the future, a compliant
   application MUST NOT use a new format when in doubt that any
   recipient can support it. Consider, for example, that while both
   the public key and the per-recipient ECDH data structure,
   respectively defined in sections 9 and 10, contain an encoded point
   field, the format changes to the field in section 10 only affect a
   given recipient of a given message.

7. Key Derivation Function

   A key derivation function (KDF) is necessary to implement the EC
   encryption. The Concatenation Key Derivation Function (Approved
   Alternative 1) [NIST SP800-56A] with the KDF hash function that is

SHA2-256 [FIPS 180-3] or stronger is REQUIRED. See section 12 for
the details regarding the choice of the hash function.

For convenience, the synopsis of the encoding method is given below
with significant simplifications attributable to the restricted
choice of hash functions in this document. However, [NIST SP800-
56A] is the normative source of the definition.

```
// Implements KDF( X, oBits, Param );

// Input: point X = (x,y)
// oBits - the desired size of output
// hBits - the size of output of hash function Hash
// Param - octets representing the parameters
// Assumes that oBits <= hBits

// Convert the point X to the octet string, see section 6:
//    ZB' = 04 || x || y
// and extract the x portion from ZB'

ZB = x;
MB = Hash ( 00 || 00 || 00 || 01 || ZB || Param );
return oBits leftmost bits of MB.
```

Note that ZB in the KDF description above is is the compact
representation of X, defined in section 4.2 of [RFC6090]

8. EC DH Algorithm (ECDH)

The method is a combination of a ECC Diffie-Hellman method to
establish a shared secret, a key derivation method to process the
shared secret into a derived key, and a key wrapping method that
uses the derived key to protect a session key used to encrypt a
message.

The One-Pass Diffie-Hellman method C(1, 1, ECC CDH) [NIST SP800-56A]
MUST be implemented with the following restrictions: the ECC CDH
primitive employed by this method is modified to always assume the
cofactor as 1, the KDF specified in section 7 is used, and the KDF
parameters specified below are used.

The KDF parameters are encoded as concatenation of the following 5
variable-length and fixed-length fields, compatible with the
definition of the OtherInfo bitstring [NIST SP800-56A]:

 o a variable-length field containing a curve OID, formatted as
    follows

      o a one-octet size of the following field

      o the octets representing a curve OID, defined in section 11

o   a one-octet public key algorithm ID defined in section 5

o   a variable-length field containing KDF parameters, identical to
    the corresponding field in the ECDH public key, formatted as
    follows

    o   a one-octet size of the following fields; values 0 and 0xff
        are reserved for future extensions

    o   a one-octet value 01, reserved for future extensions

    o   a one-octet hash function ID used with the KDF

    o   a one-octet algorithm ID for the symmetric algorithm used
        to wrap the symmetric key for message encryption, see
        section 8 for details

o   20 octets representing the UTF-8 encoding of the string
    "Anonymous Sender   ", which is the octet sequence
     41 6E 6F 6E 79 6D 6F 75 73 20 53 65 6E 64 65 72 20 20 20 20

o   20 octets representing a recipient encryption subkey or a master
    key fingerprint, identifying the key material that is needed for
    the decryption

The size of the KDF pameters sequence, defined above, is either 54
or 51 for the three curves defined in this document.

The key wrapping method is described in [RFC3394]. KDF produces a
symmetric key that is used as a KEK as specified in [RFC3394].
Refer to section 13 for the details regarding the choice of the KEK
algorithm, which SHOULD be one of three AES algorithms. Key
wrapping and unwrapping is performed with the Default Initial Value
of [RFC3394].

The input to the key wrapping method is the value "m" derived from
the session key, as described in section 5.1. Public-Key Encrypted
Session Key Packets (Tag 1) of [RFC4880], except, the PKCS#1.5
padding step is omitted. The result is padded using the method
described in [PKCS5] to the 8-byte granularity. For example, a
following AES-256 session key, which 32 octets are denoted from k0
to k31, is composed to form the following 40 octet sequence:

  09 k0 k1 ... k31 c0 c1 05 05 05 05 05

The octets c0 and c1 above denote the checksum. This encoding
allows the sender to obfuscate the size of the symmetric encryption
key used to encrypt the data. For example, assuming that an AES
algorithm is used for the session key, the sender MAY use 21, 13,
and 5 bytes of padding for AES-128, AES-192, and AES-256,

respectively, to provide the same number of octets, 40 total, as an
input to the key wrapping method.

The output of the method consists of two fields. The first field
is the MPI containing the ephemeral key used to establish the
shared secret. The second field is composed of the following two
fields:

o   a one octet, encoding the size in octets of the result of the
    key wrapping method; the value 255 is reserved for future
    extensions

o   up to 254 octets representing the result of the key wrapping
    method, applied to the 8-byte padded session key, as described
    above

Note that for session key sizes 128, 192, and 256 bits the size of
the result of the key wrapping method is, respectively, 32, 40, and
48 octets, unless the size obfuscation is used.

For convenience, the synopsis of the encoding method is given
below, however, this section, [NIST SP800-56A], and [RFC3394] are
the normative sources of the definition.

Obtain the authenticated recipient public key R
Generate an ephemeral key pair {v, V=vG}
Compute the shared point S = vR;
m = symm_alg_ID || session key || checksum || pkcs5_padding;

curve_OID_len = (byte)len(curve_OID);

Param = curve_OID_len || curve_OID || public_key_alg_ID || 03
|| 01 || KDF_hash_ID || KEK_alg_ID for AESKeyWrap || "Anonymous
Sender   " || recipient_fingerprint;

Z_len = the key size for the KEK_alg_ID used with AESKeyWrap
Compute Z = KDF(S, Z_len, Param);
Compute C = AESKeyWrap(Z, m) as per [RFC3394]

VB = convert point V to the octet string
Output (MPI(VB) || len(C) || C).

The decryption is the inverse of the method given. Note that the
recipient obtains the shared secret by calculating

  S = rV = rvG, where (r,R) is the recipient's key pair.

Consistent with section 5.13 Sym. Encrypted Integrity Protected
Data Packet (Tag 18) of [RFC4880], MDC MUST be used anytime
symmetric key is protected by ECDH.

9. Encoding of public and private keys

   The following algorithm-specific packets are added to Section 5.5.2
   Public-Key Packet Formats of [RFC4880] to support ECDH and ECDSA.

   This algorithm-specific portion is:

   Algorithm-Specific Fields for ECDSA keys:
       o a variable-length field containing a curve OID, formatted
          as follows

           o a one-octet size of the following field; values 0 and
              0xFF are reserved for future extensions

           o octets representing a curve OID, defined in section 11

       o MPI of an EC point representing a public key

     Algorithm-Specific Fields for ECDH keys:

       o a variable-length field containing a curve OID, formatted
          as follows

           o a one-octet size of the following field; values 0 and
              0xFF are reserved for future extensions

           o the octets representing a curve OID, defined in
              section 11

       o MPI of EC point representing public key

       o a variable-length field containing KDF parameters,
          formatted as follows

           o a one-octet size of the following fields; values 0 and
              0xff are reserved for future extensions

           o a one-octet value 01, reserved for future extensions

           o a one-octet hash function ID used with a KDF

           o a one-octet algorithm ID for the symmetric algorithm
              used to wrap the symmetric key used for the message
              encryption; see section 8 for details

   Observe that an ECDH public key is composed of the same sequence of
   fields that define an ECDSA key, plus the KDF parameters field.

The following algorithm-specific packets are added to section
5.5.3. Secret-Key Packet Formats of [RFC4880] to support ECDH and
ECDSA.

   Algorithm-Specific Fields for ECDH or ECDSA secret keys:

      o an MPI of an integer representing the secret key, which is
         a scalar of the public EC point

10. Message encoding with public keys

   Section 5.2.2. Version 3 Signature Packet Format defines signature
   formats. No changes in the format are needed for ECDSA.

   Section 5.1. Public-Key Encrypted Session Key Packets (Tag 1) is
   extended to support ECDH. The following two fields are the result
   of applying the KDF, as described in section 8.

   Algorithm Specific Fields for ECDH:
      o an MPI of EC point representing an ephemeral public key

      o a one octet size, followed by a symmetric key encoded using
         the method described in section 8.

11. ECC curve OID

   The parameter curve OID is an array of octets that define a named
   curve. The table bellow specifies the exact sequence of bytes for
   each named curve referenced in this document:

ASN.1 Object Identifier	OID len	Curve OID bytes in hexadecimal representation	Curve name in [FIPS 186-3]
1.2.840.10045.3.1.7	8	2A 86 48 CE 3D 03 01 07	NIST curve P-256
1.3.132.0.34	5	2B 81 04 00 22	NIST curve P-384
1.3.132.0.35	5	2B 81 04 00 23	NIST curve P-521

   The sequence of octets in the third column is the result of
   applying the Distinguished Encoding Rules (DER) to the ASN.1 Object
   Identifier with subsequent truncation. The truncation removes the
   two fields of encoded Object Identifier. The first omitted field
   is one octet representing the Object Identifier tag and the second
   omitted field is the length of the Object Identifier body. For

example, the complete ASN.1 DER encoding for the NIST P-256 curve
OID is "06 08 2A 86 48 CE 3D 03 01 07", from which the first entry
in the table above is constructed by omitting the first two octets.
Only the truncated sequence of octets is the valid representation
of a curve OID.

12. Compatibility profiles

12.1. OpenPGP ECC profile

A compliant application MUST implement NIST curve P-256, MAY
implement NIST curve P-384, and SHOULD implement NIST curve P-521,
defined in section 11. A compliant application MUST implement
SHA2-256, and SHOULD implement SHA2-384 and SHA2-512. A compliant
application MUST implement AES-128 and SHOULD implement AES-256.

A compliant application SHOULD follow section 13 regarding the
choice of the following algorithms for each curve:

o   the KDF hash algorithm

o   the KEK algorithm

o   the message digest algorithm and the hash algorithm used in the
    key certifications

o   the symmetric algorithm used for message encryption.

It is recommended that the chosen symmetric algorithm for message
encryption be no less secure than the KEK algorithm.

12.2. Suite-B profile

A subset of algorithms allowed by this document can be used to
achieve [Suite B] compatibility. The references to [Suite B] in
this document are informative. This document is primarily
concerned with format specification, leaving additional security
restrictions unspecified, such as matching assigned security level
of information to authorized recipients or interoperability
concerns arising from fewer allowed algorithms in [Suite B] than
allowed by [RFC4880].

12.2.1. Security strength at 192 bits

To achieve the security strength of 192 bits [Suite B] requires
NIST curve P-384, AES-256, and SHA2-384. The symmetric algorithm
restriction means that the algorithm of KEK used for key wrapping
in section 8 and a [RFC4880] session key used for message
encryption must be AES-256. The hash algorithm restriction means

that the hash algorithms of KDF and the [RFC4880] message digest
calculation must be SHA-384.

12.2.2. Security strength at 128 bits

   The set of algorithms in section 12.2.1 is extended to allow NIST
   curve P-256, AES-128, and SHA2-256.

13. Security Considerations

   Refer to [FIPS 186-3] B.4.1 for the method to generate a uniformly
   distributed ECC private key.

   The curves proposed in this document correspond to the symmetric
   key sizes 128 bits, 192 bits, and 256 bits, as described in the
   table below. This allows a compliant application to offer balanced
   public key security which is compatible with the symmetric key
   strength for each AES algorithm allowed by [RFC4880].

   The following table defines the hash and the symmetric encryption
   algorithm that SHOULD be used with a given curve for ECDSA or ECDH.
   A stronger hash algorithm or a symmetric key algorithm MAY be used
   for a given ECC curve. However, note that the increase in the
   strength of the hash algorithm or the symmetric key algorithm may
   not increase the overall security offered by the given ECC key.

Curve name	ECC strength	RSA strength, informative	Hash size	Symmetric key size
NIST curve P-256	256	3072	256	128
NIST curve P-384	384	7680	384	192
NIST curve P-521	521	15360	512	256

   Requirement levels indicated elsewhere in this document lead to the
   following combinations of algorithms in OpenPGP profile: MUST
   implement NIST curve P-256 / SHA2-256 / AES-128, SHOULD implement
   NIST curve P-521 / SHA2-512 / AES-256, MAY implement NIST curve P-
   384 / SHA2-384 / AES-256, among other allowed combinations.

   Consistent with the table above, the following table defines the
   KDF hash algorithm and the AES KEK encryption algorithm that SHOULD
   be used with a given curve for ECDH. Stronger KDF hash algorithm
   or AES KEK algorithm MAY be used for a given ECC curve.

Curve name	Recommended KDF hash algorithm	Recommended KEK encryption algorithm
NIST curve P-256	SHA2-256	AES-128
NIST curve P-384	SHA2-384	AES-192
NIST curve P-521	SHA2-512	AES-256

This document explicitly discourages the use of algorithms other
than AES as a KEK algorithm because backward compatibility of the
ECDH format is not a concern. The KEK algorithm is only used
within the scope of a Public-Key Encrypted Session Key Packet,
which represents an ECDH key recipient of a message. Compare this
with the algorithm used for the session key of the message, which
MAY be different from a KEK algorithm.

Compliant applications SHOULD implement, advertise through key
preferences, and use in compliance with [RFC4880] the strongest
algorithms specified in this document.

Note that the [RFC4880] symmetric algorithm preference list may
make it impossible to use the balanced strength of symmetric key
algorithms for a corresponding public key. For example, the
presence of the symmetric key algorithm IDs and their order in the
key preference list affects the algorithm choices available to the
encoding side, which in turn may make the adherence to the table
above unfeasible. Therefore, compliance with this specification is
a concern throughout the life of the key, starting immediately
after the key generation when the key preferences are first added
to a key. It is generally advisable to position a symmetric
algorithm ID of strength matching the public key at the head of the
key preference list.

Encryption to multiple recipients often results in an unordered
intersection subset. For example, if the first recipient's set is
{A, B} and the second's is {B, A}, the intersection is an unordered
set of two algorithms A and B. In this case a compliant
application SHOULD choose the stronger encryption algorithm.

Resource constraints, such as limited computational power, is a
likely reason why an application might prefer to use the weakest
algorithm. On the other side of the spectrum are applications that
can implement every algorithm defined in this document. Most
applications are expected to fall into either of two categories. A

compliant application in the second, or strongest, category SHOULD
prefer AES-256 to AES-192.

SHA-1 MUST NOT be used with the ECDSA or the KDF in the ECDH
method.

MDC MUST be used when symmetric encryption key is protected by
ECDH. None of the ECC methods described in this document are
allowed with deprecated V3 keys. A compliant application MUST only
use Iterated and Salted S2K to protect private keys, as defined in
section 3.7.1.3 Iterated and Salted S2K of [RFC4880].

Side channel attacks are a concern when a compliant application's
use of OpenPGP format can be modeled by a decryption or signing
oracle model, for example, when an application is a network service
performing decryption to unauthenticated remote users. ECC scalar
multiplication operations used in ECDSA and ECDH are vulnerable to
side channel attacks. Countermeasures can often be taken at the
higher protocol level, such as limiting the number of allowed
failures or time-blinding of the operations associated with each
network interface. Mitigations at the scalar multiplication level
seek to eliminate any measurable distinction between ECC point
addition and doubling operations.

14. IANA Considerations

This document asks IANA to assign an algorithm number from the
OpenPGP Public-Key Algorithms range, or the "name space" in the
terminology of [RFC2434], that was created by [RFC4880]. Two ID
numbers are requested, as defined in section 5. The first one with
value 19 is already designated for ECDSA and is currently unused,
while another one is new (and suggested to be 18; there is an
implementation advantage in having consecutive ID values for the
two complementary algorithms).

15. References

15.1. Normative references

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

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R.
Thayer, "OpenPGP Message Format", November 2007

[Suite B] NSA, US Government, NSA Suite B Cryptography, March 11,
2010, http://www.nsa.gov/ia/programs/suiteb_cryptography/

[FIPS 186-3] US Dept. of Commerce / NIST, "Digital Signature
Standard (DSS)", June 2009

[NIST SP800-56A] Elaine Barker, Don Johnson, and Miles Smid, "Recommendation for Pair-WiseKey Establishment Schemes Using Discrete Logarithm Cryptography (Revised)", March 2007

[FIPS 180-3] NIST, "Secure Hash Standard (SHS)", October 2008

[RFC3394] J. Schaad, R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", September 2002

[PKCS5] RSA Laboratories, "PKCS #5 v2.0: Password-Based Cryptography Standard", March 25, 1999

[RFC2434] Narten, T., Alvestrand, H., "Guidelines for Writing IANA Considerations Section in RFCs", October 1998

15.2. Informative references

[KOBLITZ] N. Koblitz, "A course in number theory and cryptography", Chapter VI. Elliptic Curves, ISBN: 0-387-96576-9, Springer-Verlag, 1987

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", February 2011,

[SEC1] Certicom Research, "SEC 1: Elliptic Curve Cryptography", September 20, 2000

Contributors

Hal Finney provided important criticism on compliance with [NIST SP800-56A] and [Suite B], and pointed out a few other mistakes.

Acknowledgment

Author's Address

Andrey Jivsov
Symantec Corporation
Email: Andrey_Jivsov@symantec.com