Network Working Group
Internet Draft
Intended status: Internet Draft
Expires: June 24, 2010

A. Jivsov
PGP Corporation
December 26, 2009

ECC in OpenPGP
draft-jivsov-openpgp-ecc-04.txt

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 June 24, 2010.

Copyright Notice

Abstract

This document proposes an Elliptic Curve Cryptography extension to the OpenPGP public key format and specifies three Elliptic Curves that enjoy broad support by other standards, including NIST standards. The document aims to standardize an optimal but narrow set of parameters for best interoperability and it does so within the framework it defines that can be expanded in the future to allow more choices.

Table of Contents

1. Introduction

The OpenPGP protocol 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].

An application MAY implement this draft; note that any [RFC2119] keyword within this draft applies to an OpenPGP application only if it chooses to implement this draft.

3. Elliptic Curve Cryptography

This specification establishes the minimum set of Elliptic Curve Cryptography public key parameters and cryptographic methods that will likely satisfy the widest range of platforms and applications and facilitate interoperability. This specification offers efficient cryptographic method for applications to match the level of security of every type of AES algorithm specified in [RFC4880].

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 [Suite B]. This specification includes algorithms permitted by Suite B, so it would be possible to build a Suite B compatible implementation based on a subset of [RFC4880] and this specification.

4. Supported ECC curves

This standard references three named prime field curves, which are defined in [FIPS 186-2] as "Curve P-256", "Curve P-384", and "Curve P-521".

In data structures that this specification defines the named curves are referenced as a sequence of bytes, called throughout this specification as Curve OID. Section 11 describes in details how this sequence of bytes is formed.

5. Supported public key algorithms

Supported public key algorithms are Elliptic Curve Digital Signature Algorithm (ECDSA), defined in [FIPS 186-2], and Elliptic Curve Diffie-Hellman (ECDH), defined in section 8.

Other compatible definition of ECDSA can be found in [SEC1].

The section 9.1. Public-Key Algorithms of [RFC4880] is expanded to define the following public key algorithm IDs:

ID	**Description of algorithm**
19	ECDSA public key algorithm
[to be ASSIGNED] presumably 22	ECDH public key algorithm

Applications MUST support ECDSA and ECDH.

6. Conversion primitives

The method to convert an EC point to the octet string is defined in [SEC1]. This specification only defines uncompressed point format. For convenience, the synopsis of the encoding method is given below, however, the [SEC1] is the normative source of the definition.

The point is encoded in MPI format. The content of the MPI is the following:

```
B = B0 || x || y
```

where x and y are coordinates of the point P = (x, y), each encoded in big endian format and zero-padded to the underlying field size.

B0 is a byte with following values:

value	description
0	Point O. In this case there is no x or y octets present.
4	Uncompressed point. x and y of EC point values follow.

Note that point O shall not appear in a public or a private key. Therefore, the size of the MPI payload is always curve_size*2 + 3 bits. For example, for "Curve P-256" the point is represented as a bit string of length 515 bits.

If other conversion methods are defined in the future, the application MAY use them only when it is certain that every recipient of the data supports the other format.

7. Key Derivation Function

A key derivation function (KDF) is necessary to implement EC encryption. The Concatenation Key Derivation Function (Approved Alternative 1) defined in [NIST SP800-56A] is REQUIRED with the following restriction: the KDF hash function MAY be any of the following hash functions specified by [FIPS 180-2]: SHA2-256, SHA2-384, SHA2-512. See section 13 for the details regarding the choice of the hash function.

For convenience, the synopsis of the encoding method is given below, however, [NIST SP800-56A] is the normative source of the definition.

```
// Implements KDF( X, oBits, P );
// Input: point X = (x,y)
// oBits - the desired size of output
// hBits - the size of output of hash function Hash
// P - octets representing the parameters

counter=1;
threshold =  (oBits + hBits - 1) / hBits;
// Convert the point P to octet string as defined in section 6:
//   ZB' = 04 || x || y
// and extract the x portion from ZB':
ZB = x;
do  {
  C32 = (uint32)big_endian(counter);
  HB = Hash ( C32 || ZB || P );
  MB = MB || HB;
  counter = counter + 1;
} while( counter <= threshold );
return oBits leftmost bits of MB
```

8. EC DH Algorithm (ECDH)

The method is a combination of ECC Diffie-Hellman method to establish a shared secret and a key wrapping method that uses the shared secret to protect symmetric encryption key.

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

Key derivation parameters MUST be encoded as concatenation of the following 7 fields, each of them, except the first one, is considered a fixed-length field of corresponding size:

- a variable-length field containing curve OID, formatted as follows
  - a one-octet size of the following field
  - octets representing curve OID, defined in section 11
- a one-octet public key algorithm ID defined in section 5
- a one-octet value 01, reserved for future extensions

- a one-octet hash function ID used in KDF; according to section 7, this octet is 08 for SHA2-256, 09 for SHA2-384, or 10 for SHA2-512
- a one-octet algorithm ID for the symmetric algorithm used to wrap the symmetric key for message encryption; the method is defined later in this section
- 15 octets representing the UTF-8 encoding of the string "AnonymousSender"
- 20 octets representing recipient encryption subkey or master key fingerprint, identifying the key material that is needed for decryption

For three curves defined in this specification the size of the key derivation parameters sequence, defined above, is either 48 or 45.

The key wrapping method is based on [RFC3394]. KDF produces the AES key that is used as KEK according to [RFC3394]. Refer to section 13 for the details regarding the choice of the KEK algorithm, which MUST be one of three AES algorithms.

The input to 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 as described bellow to form a 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 size of the symmetric encryption key used to encrypt the data. To do this the sender MAY use 21, 13, and 5 bytes of padding for AES-128, AES-192, and AES-256, respectfully, 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 with the ephemeral key used to establish shared secret. The second field is composed of the following two fields:

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

- up to 254 octets representing the result of the key wrapping method applied to session key encoded 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, respectfully, 32, 40, and 48 octets.

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 authenticated recipient public key R
Generate ephemeral key pair {v, V=vG}
Compute 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 ||
    01 || KDF_hash_ID || AES_alg_ID for AESKeyWrap ||
   "AnonymousSender" || recipient_fingerprint;
Z_len = key size for AES_alg_ID to be used with AESKeyWrap
Compute Z = KDF( S, Z_len, Param );
Compute C = AESKeyWrap( Z, m ) as per [RFC3394]
VB = convert point V to 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], the MDC SHOULD 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 ECDH keys:

- a variable-length field containing curve OID, formatted as follows
  - a one-octet size of the following field; values 0 and 0xFF are reserved for future extensions

- octets representing curve OID, defined in section 11

- a one-octet value 01, reserved for future extension
- a one-octet hash function ID used with KDF
- a one-octet algorithm ID for the symmetric algorithm used to wrap the symmetric key for message encryption, see section 8 for details
- MPI of EC point representing public key

Algorithm-Specific Fields for ECDSA keys:

- a variable-length field containing curve OID, formatted as follows
  - a one-octet size of the following field; values 0 and 0xFF are reserved for future extensions
  - octets representing curve OID, defined in section 11
- MPI of EC point representing public key

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:

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

10. Data encoding with public keys

Section 5.2.2. Version 3 Signature Packet Format defines signature formats. No changes in 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 result of applying KDF, as described in section 8.

Algorithm Specific Fields for ECDH:

- an MPI of EC point representing ephemeral public key
- 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 the named curve. The table bellow specifies the exact sequence of bytes for each named curve referenced in this specification:

ASN.1 Object Identifier	**OID len**	**Curve OID bytes in hexadecimal representation**	**Curve name in [FIPS 186-2]**
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 Distinguished Encoding Rules (DER) to the ASN.1 Object Identifier with subsequent truncation. The truncation removes 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 is "06 08 2A 86 48 CE 3D 03 01 07", from which the entry in the table above is constructed by omitting two first octets.

12. Compatibility profiles

12.1. OpenPGP ECC profile

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

Application SHOULD follow section 13 regarding the choice of the following algorithms for each curve

- the KDF hash algorithm
- KEK algorithm
- message digest algorithm and hash algorithm used in key certifications

o message encryption symmetric algorithm.

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 specification 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 unspecified additional security restrictions, such as matching security level of information with authorized recipients or interoperability concerns arising from fewer allowed algorithms in [Suite B] than in [RFC4880].

12.2.1. Secret information

Applications MUST use NIST curves P-256 or P-384. KEK MUST be used with AES-128 or AES-256. KDF MUST be used with SHA2-256 or SHA2-384.

Note that the most secure algorithm in of each of 3 categories above is also listed in the section 12.2.2.

12.2.2. Top Secret information

Application MUST use NIST curve P-384. KEK MUST be used with AES-256. SHA2-384 MUST be used for KDF.

13. Security Considerations

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 OpenPGP application to offer security comparable with the strength of each AES algorithms allowed by [RFC4880].

The following table defines the hash and symmetric encryption algorithm that SHOULD be used with specific curve for ECDSA or ECDH. Stronger hash algorithm or symmetric key algorithm MAY be used for a given ECC curve. However, note that the increase in the strength of the hash algorithm or 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 result in the effective support for 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 AES KEK encryption algorithm that SHOULD be used with specific curve for ECDH. Stronger KDF hash algorithm or 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

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

Note that [RFC4880] symmetric algorithm preference list may restrict the use of balanced strength of symmetric key algorithms for corresponding public key. For example, the presence of symmetric key algorithms and their order in key preference list affects the choices available to encoding side for compliance with the table above. Therefore, applications need to be concerned with

this compliance throughout the life of the key, starting immediately after key generation when the key preferences are first added to a key. It is generally advisable to have at the head of the key preference list a symmetric algorithm of strength corresponding to the public key.

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

Resource constraint, such as limited computational power, is the likely reason why an application might prefer to use weakest algorithms. On the other side of the spectrum are applications that can implement every algorithm defined in this document. Most of applications are expected to fall into either of two categories. An application in the second or strongest category SHOULD prefer AES-256 to AES-192.

While some statements in this specification refer to TripleDES algorithm, this is only done to help interoperability with existing application and already generated keys; AES-256 is the recommended alternative to TripleDES in all circumstances when AES-256 is available.

SHA-1 MUST NOT be used for ECDSA or as part of 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. The 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].

14. IANA Considerations

This document asks IANA to assign an algorithm number from OpenPGP Public-Key Algorithms range, or "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 currently unused, while another one is new (and expected to be 22).

15. 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, Fact Sheet NSA Suite B Cryptography, 2005, http://www.nsa.gov/ia/Industry/crypto_suite_b.cfm

[FIPS 186-2] US Dept. of Commerce / NIST, "DIGITAL SIGNATURE STANDARD (DSS)", 2001 October 5

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

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

[FIPS 180-2] NIST, SECURE HASH STANDARD, 2002 August 1

[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

Contributors

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

Acknowledgment

The author would like to acknowledge the help of many individuals who kindly voiced their opinions on IETF OpenPGP Working Group mailing list and, in particular the help of Jon Callas, David Crick, Ian G, Werner Koch. [to be continued]

Author's Address

Andrey Jivsov
PGP Corporation
Email: ajivsov@pgp.com