Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 29, 2010

D. McGrew
Cisco Systems

October 26, 2009

Fundamental Elliptic Curve Cryptography Algorithms
draft-mcgrew-fundamental-ecc-01.txt

Status of this Memo

Copyright Notice

Abstract

This note describes the fundamental algorithms of Elliptic Curve Cryptography (ECC) as they are defined in some early references. These descriptions may be useful to those who want to implement the fundamental algorithms without using any of the specialized methods that were developed in following years. Only elliptic curves defined over fields of characteristic greater than three are in scope; these curves are those used in Suite B.

Table of Contents

1. Introduction

ECC is a public-key technology that offers performance advantages at higher security levels. It includes an Elliptic Curve version of Diffie-Hellman key exchange protocol [DH1976] and an Elliptic Curve version of the ElGamal Signature Algorithm [E1985]. The elliptic curve versions of these algorithms are referred to as ECDH and ECES, respectively. The adoption of ECC has been slower than had been anticipated, perhaps due to the lack of freely available normative documents and uncertainty over intellectual property rights.

This note contains a description of the fundamental algorithms of ECC over fields with characteristic greater than three, based directly on original references. Its intent is to provide the Internet community with a normative specification of the basic algorithms that predate any specialized or optimized algorithms.

The rest of the note is organized as follows. Section 2.1, Section 2.2, and Section 2.3 furnish the necessary terminology and notation from modular arithmetic, group theory and the theory of finite fields, respectively. Section 3 defines the groups based on elliptic curves over finite fields of characteristic greater than three. Section 4 and Section 5 present the fundamental ECDH and ECES algorithms, respectively. Section 6 presents an abbreviated form of ECES. The previous sections contain all of the normative text (the text that defines the norm for implementations conforming to this specification), and all of the following sections are purely informative. Interoperability is discussed in Section 7. Section 8 reviews intellectual property issues. Section 9 summarizes security

considerations. Appendix B describes random number generation and Appendix C provides an example of an Elliptic Curve group.

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 Appendix A.

2. Mathematical Background

This section reviews mathematical preliminaries and establishes terminology and notation that is used below.

2.1. Modular Arithmetic

This section reviews modular arithmetic. Two integers x and y are said to be congruent modulo n if x - y is an integer multiple of n.

Two integers x and y are coprime when their greatest common divisor is 1; in this case, there is no third number z > 1 such that z divides x and z divides y.

The set Zq = { 0, 1, 2, ..., q-1 } is closed under the operations of modular addition, modular subtraction, modular multiplication, and modular inverse. These operations are as follows.

For each pair of integers a and b in Zq, a + b mod q is equal to a + b if a + b < q, and is equal to a + b - q otherwise.

For each pair of integers a and b in Zq, a - b mod q is equal to

a - b if a - b >= 0, and is equal to a - b + q otherwise.

For each pair of integers a and b in Zq, a * b mod q is equal to the remainder of a * b divided by q.

For each integer x in Zq that is coprime with q, the inverse of x modulo q is denoted as 1 / x mod q, and can be computed using the extended euclidean algorithm (see Section 4.5.2 of [K1981v2], for example).

Algorithms for these operations are well known; for instance, see Chapter 4 of [K1981v2].

2.2. Group Operations

This section establishes some terminology and notation for mathematical groups, which is needed later on. Background references abound; see [D1966], for example.

A group is a set of elements G together with an operation that combines any two elements in G and returns a third element in G. The operation is denoted as * and its application is denoted as a * b, for any two elements a and b in G. The operation is associative, that is, for all a, b and c in G, a * (b * c) is identical to (a * b) * c. Repeated application of the group operation N times to the element a is denoted as a^N, for any element a in G and any positive integer N.

That is, a^2, = a * a, a^3 = a * a * a, and so on. The associativity of the group operation ensures that the computation of a^n is unambiguous; any grouping of the terms gives the same result.

The above definition of a group operation uses multiplicative notation. Sometimes an alternative called additive notation is used, in which a * b is denoted as a + b, and a^N is denoted as N * a. In multiplicative notation, g^N is called exponentiation, while the equivalent operation in additive notation is called scalar multiplication. In this document, multiplicative notation is used throughout for consistency.

Every group has an special element called the identity element, which we denote as e. For each element a in G, e * a = a * e = a. By convention, a^0 is equal to the identity element for any a in G.

Every group element a has a unique inverse element b such that a * b = b * a = e. The inverse of a is denoted as a^-1 in multiplicative notation. (In additive notation, the inverse of a is denoted as -a.)

A cyclic group of order R is a group that contains the R elements g, g^2, g^3, ..., g^R. The element g is called the generator of the group. The element g^R is equal to the identity element e. Note that g^X is equal to g^(X modulo R) for any non-negative integer X.

Given the element a of order N, and an integer i between 1 and N-1, inclusive, the element a^i can be computed by the "square and multiply" method outlined in Section 2.1 of [M1983] (see also Knuth, Vol. 2, Section 4.6.3.), or other methods.

2.3. Finite Fields

This section establishes terminology and notation for finite fields with prime characteristic.

When p is a prime number, then the set Zp, with the addition, subtraction, multiplication and division operations, is a finite field with characteristic p. Each nonzero element x in Zp has an inverse 1/x. There is a one-to-one correspondence between the integers between zero and p-1, inclusive, and the elements of the field. The field is denoted as Fp.

Equations involving field elements do not explicitly denote the "mod p" operation, but it is understood to be implicit. For example, the statement that x, y, and z are in Fp and

z = x + y

is equivalent to the statement that x, y, and z are in the set { 0, 1, ..., p-1 } and

z = x + y mod p.

3. Elliptic Curve Groups

This note only covers elliptic curves over fields with characteristic greater than three; these are the curves used in Suite B [SuiteB]. For other fields, the definition of the elliptic curve group would be different.

An elliptic curve over a field F is defined by the curve equation

```
y^2 = x^3 + a*x + b,
```

where x, y, a, and b are elements of the field Fp, and the discriminant 16*(4*a^3 - 27*b^2) is nonzero [M1985]. A point on an elliptic curve is a pair (x,y) of values in Fp that satisfy the curve equation, such that x and y are both in Fp, or it is a special point (@,@) that represents the identity element (which is called the "point at infinity"). The order of an elliptic curve group is the number of distinct points.

Two elliptic curve points (x1,y1) and (x2,y2) are equal whenever x1=x2 and y1=y2, or when both points are the point at infinity. The inverse of the point (x1,y1) is the point (x1,-y1).

The group operation associated with the elliptic curve group is as follows [BC1989]. To an arbitrary pair of points P and Q specified by their coordinates (x1,y1) and (x2,y2) respectively, the group operation assigns a third point P*Q with the coordinates (x3,y3). These coordinates are computed as follows

```
(x3,y3) = (@,@) when P is not equal to Q and x1 is equal to x2.

x3 = ((y2-y1)/(x2-x1))^2 - x1 - x2 and
y3 = (x1-x3)*(y2-y1)/(x2-x1) - y1 when P is not equal to Q and
x1 is not equal to x2.

(x3,y3) = (@,@) when P is equal to Q and y1 is equal to 0,

x3 = ((3*x1^2 + a)/(2*y1))^2 - 2*x1 and
y3 = (x1-x3)*(3*x1^2 + a)/(2*y1) - y1 if P is equal to Q and y1 is
not equal to 0.
```

In the above equations, a, x1, x2, x3, y1, y2, and y3 are elements of the field Fp; thus, computation of x3 and y3 in practice must reduce the right-hand-side modulo p.

The representation of elliptic curve points as a pair of integers in Zp is known as the affine coordinate representation. This representation is suitable as an external data representation for

communicating or storing group elements, though the point at infinity must be treated as a special case.

Some pairs of integers are not valid elliptic curve points. A valid pair will satisfy the curve equation, while an invalid pair will not.

3.1. Homogeneous Coordinates

An alternative way to implement the group operation is to use homogeneous coordinates [K1987] (see also [KMOV1991]). This method is typically more efficient because it does not require a modular inversion operation.

An elliptic curve point (x,y) (other than the point at infinity (@,@)) is equivalent to a point (X,Y,Z) in homogeneous coordinates whenever x=X/Z mod p and y=Y/Z mod p.

Let P1=(X1,Y1,Z1) and P2=(X2,Y2,Z2) be points on an elliptic curve and suppose that the points P1, P2 are not equal to (@,@), P1 is not equal to P2, and P1 is not equal to P2^-1. Then the product P3=(X3,Y3,Z3) = P1 * P2 is given by

```
X3 = v * (Z2 * (Z1 * u^2 - 2 * X1 * v^2) - v^3) mod p,

Y3 = z2 * (3 * X1 * u * v^2 - Y1 * v^3 - Z1 * u^3) mod p,

Z3 = 8 * (Y1)^3 * (Z1)^3 mod p,
```

where u = Y2 * Z1 - Y1 * Z2 mod p and v = X2 * Z1 - X1 * Z2 mod p.

When the points P1 and P2 are equal, then (X1/Z1, Y1/Z1) is equal to (X2/Z2, Y2/Z2), which is true if and only if u and v are both equal to zero.

The product P3=(X3,Y3,Z3) = P1 * P1 is given by

```
X3 = 2 * Y1 * Z1 * (w^2 - 8 * X1 * Y1^2 * Z1) mod p,

Y3 = 4 * Y1^2 * Z1 * (3 * w * X1 - 2 * Y1^2 * Z1) - w^3 mod p,

Z3 = 8 * (Y1 * Z1)^3 mod p,
```

where w = 3 * X1^2 + a * Z1^2 mod p. In the above equations, a, u, v, w, X1, X2, X3, Y1, Y2, Y3, Z1, Z2, and Z3 are integers in the set Fp.

When converting from affine coordinates to homogeneous coordinates, it is convenient to set Z to 1. When converting from homogeneous

coordinates to affine coordinates, it is necessary to perform a modular inverse to find 1/Z mod p.

3.2. Group Parameters

An elliptic curve group over a finite field with characteristic greater than three is completely specified by the following parameters:

The prime number p that indicates the order of the field Fp.

The value a used in the curve equation.

The value b used in the curve equation.

The generator g of the group.

The order n of the group generated by g.

An example of an Elliptic Curve Group is provided in Appendix C.

Each elliptic curve point is associated with a particular group, i.e a particular parameter set. Two elliptic curve groups are equal if and only if each of the parameters in the set are equal. The elliptic curve group operation is only defined between two points on the same group. It is an error to apply the group operation to two elements that are from different groups, or to apply the group operation to a pair of coordinates that are not a valid point. See Section 9.3 for further information.

3.2.1. Security

Security is highly dependent on the choice of these parameters. This section gives normative guidance on acceptable choices. See also Section 9 for informative guidance.

The order of the group generated by g MUST be divisible by a large prime, in order to preclude easy solution of the discrete logarithm problem [K1987]

With some parameter choices, the discrete log problem is significantly easier to solve. This includes parameter sets in which b = 0 and p = 3 (mod 4), and parameter sets in which a = 0 and p = 2 (mod 3) [MOV1993]. These parameter choices are inferior for cryptographic purposes and SHOULD NOT be used.

4. Elliptic Curve Diffie-Hellman (ECDH)

The Diffie-Hellman (DH) key exchange protocol [DH1976] allows two parties communicating over an insecure channel to agree on a secret key. It was originally defined in terms of operations in the multiplicative group of a field with a large prime characteristic. Massey [M1983] observed that it can be easily generalized so that it is defined in terms of an arbitrary mathematical group. Miller [M1985] and Koblitz [K1987] analyzed the DH protocol over an elliptic curve group. We describe DH following the former reference.

Let G be a group, and g be a generator for that group, and let t denote the order of G. The DH protocol runs as follows. Party A chooses an exponent j between 1 and t-1 uniformly at random, computes g^j and sends that element to B. Party B chooses an exponent k between 1 and t-1 uniformly at random, computes g^k and sends that element to A. Each party can compute g^(j*k); party A computes (g^k)^j, and party B computes (g^j)^k.

See Appendix B regarding generation of random numbers.

4.1. Data Types

An ECDH private key z is an integer in Zt.

The corresponding ECDH public key Y is the group element, where Y = g^z. Each public key is associated with a particular group, i.e. a particular parameter set as per Section 3.2.

The shared secret computed by both parties is a group element.

Each run of the ECDH protocol is associated with a particular group,

and both of the public keys and the shared secret are elements of that group.

4.2. Compact Representation

As described in the final paragraph of [M1985], the x-coordinate of the shared secret value g^(j*k) is a suitable representative for the entire point whenever exponentiation is used as a one-way function. In the ECDH key exchange protocol, after the element g^(j*k) has been computed, the x-coordinate of that value can be used as the shared secret. We call this compact output.

Following [M1985] again, when compact output is used in ECDH, only the x-coordinate of an elliptic curve point needs to be transmitted, instead of both coordinates as in the typical affine coordinate representation. We call this the compact representation.

ECDH can be used with or without compact output. Both parties in a particular run of the ECDH protocol MUST use the same method. ECDH can be used with or without compact representation. If compact representation is used in a particular run of the ECDH protocol, then compact output MUST be used as well.

5. Elliptic Curve ElGamal Signatures (ECES)

The ElGamal signature algorithm was introduced in 1984 [E1984a] [E1984b] [E1985]. It is based on the discrete logarithm problem in the multiplicative group of the integers modulo a large prime number. It is straightforward to extend it to use an elliptic curve group. In this section we recall a well-specified elliptic curve version of the ElGamal Signature Algorithm, as described in [A1992] and [MV1993]. This signature method is called Elliptic Curve ElGamal Signatures (ECES).

The algorithm uses an elliptic curve group, as described in Section 3.2, with prime field order p, curve equation parameters a and b. We follow [MV1993] in describing the algorithms in terms of mathematical groups, and denoting the generator as alpha, and its order as n.

ECES uses a collision-resistant hash function, so that it can sign

messages of arbitrary length. We denote the hash function as h(). Its input is a bit string of arbitrary length, and its output is an integer between zero and n-1, inclusive.

ECES uses a function g() from the set of group elements to the set of integers Zn. This function returns the x-coordinate of the affine coordinate representation of the elliptic curve point.

5.1. Keypair Generation

The private key z is an integer between 0 and n - 1, inclusive, generated uniformly at random. The public key is the group element Q = alpha^z.

5.2. Signature Creation

To sign message m, using the private key z:

1. First, choose an integer k uniformly at random from the set of all integers k in Zn that are coprime to n. (If n is a prime, then choose an integer uniformly at random between 1 and n-1.) (See Appendix B regarding random integers.)

2. Next, compute the group element r = alpha^k.

3. Finally, compute the integer s as

$$s = (h(m) + z * g(r)) / k \pmod{n}.$$

4. If s is equal to zero, then the signature creation MUST be repeated, starting at Step 1 and using a newly chosen k value.

The signature for message m is the ordered pair (r, s). Note that the first component is a group element, and the second is a non-negative integer.

5.3. Signature Verification

To verify the message m and the signature (r,s) using the public key Q:

Compute the group element r^s * Q^(-g(r)).

Compute the group element alpha^h(m).

Verify that the two elements previously computed are the same. If they are identical, then the signature and message pass the verification; otherwise, they fail.

5.4. Hash Functions

Let H() denote a hash function whose output is a fixed-length bit string. To use H in ECES, we define the mapping between that output and the integers between zero and n-1; this realizes the function h() described above. Given a bit string m, the function h(m) is computed as follows:

1. H(m) is evaluated; the result is a fixed-length bit string.

2. Convert the resulting bit string to an integer i by treating its leftmost (initial) bit as the most significant bit of i, and treating its rightmost (final) bit as the least significant bit of i.

3. After conversion, reduce i modulo n, where n is the group order.

5.5. Rationale

This subsection is not normative and is provided only as background information.

The signature verification will pass whenever the signature is properly generated, because

r^s * Q^(-g(r)) = alpha^(k*s - z*g(r)) = alpha^h(m).

The reason that the random variable k must be coprime with n is so

that 1/k mod n is defined.

A valid signature with s=0 leaks the secret key, since in that case a = h(m) / g(r) mod n. We adopt Rivest's suggestion to avoid this

problem [R1992].

As described in the final paragraph of [M1985], it is suitable to use the x-coordinate of a particular elliptic curve point as a representative for that point. This is what the function g() does.

6. Abbreviated ECES Signatures (AECES)

The ECES system is secure and efficient, but has signatures that are slightly larger than they need to be. Koyama and Tsuruoka described a signature system based on Elliptic Curve ElGamal, but with shorter signatures [KT1994]. Their idea is to include only the x-coordinate of the EC point in the signature, instead of both coordinates. Menezes, Qu, and Vanstone independently developed the same idea, which was the basis for the "Elliptic Curve Signature Scheme with Appendix (ECSSA)" submission to the IEEE 1363 working group [MQV1994].

In this section we describe an Elliptic Curve Signature Scheme that uses a single elliptic curve coordinate in the signature instead of both coordinates. It is based on [KT1994] and [MQV1994], but with the finite field inversion operation moved from the signature operation to the verification operation, so that the signing operation is more compatible with ECES. (See [AMV1990] and [A1992] for a discussion of these alternatives; the security of the methods is equivalent.) We refer to this scheme as Abbreviated ECES, or AECES.

6.1. Keypair Generation

Keypairs are the same as for ECES and are as described in Section 5.1.

6.2. Signature Creation

In this section we describe how to compute the signature for a message m using the private key z.

Signature creation is as for ECES, with the following additional step:

1. Let the integer s1 be equal to the x-coordinate of r.

The signature is the ordered pair (s1, s). Both signature components are non-negative integers.

6.3. Signature Verification

Given the message m, the public key Q, and the signature (s1, s) verification is as follows:

1. Compute the inverse of s modulo n. We denote this value as w.

2. Compute the non-negative integers u and v, where

   u = w * h(m) mod n, and

   v = w * s1 mod n.

3. Compute the elliptic curve point R' = alpha^u * Q^v

4. If the x-coordinate of R' is equal to s1, then the signature and message pass the verification; otherwise, they fail.

7. Interoperability

The algorithms in this note can be used to interoperate with some other ECC specifications. This section provides details for each algorithm.

7.1. ECDH

Section 4 can be used with the Internet Key Exchange (IKE) versions one [RFC2409] or two [RFC4306]. These algorithms are compatible with the ECP groups for the defined by [RFC4753], [RFC2409], and [RFC2412]. The group definition used in this protocol uses an affine coordinate representation of the public key and uses neither the compact output nor the compact representation of Section 4.2. Note that some groups use a negative curve parameter "a" and express this fact in the curve equation rather than in the parameter. The test cases in Section 8 of [RFC4753] can be used to test an implementation; these cases use the multiplicative notation, as does this note. The KEi and KEr payloads are equal to g^i and g^r, respectively, with 64 bits of encoding data prepended to them.

The algorithms in Section 4 can be used to interoperate with the IEEE [P1363] and ANSI [X9.62] standards for ECDH based on fields of characteristic greater than three. To use IEEE P1363 ECDH in a manner that will interoperate with this specification, the following options and parameter choices should be used: prime curves with a cofactor of 1, the ECSVDP-DH primitive, and the Key Derivation Function must be the "identity" function (equivalently, omit the KDF step and output the shared secret value directly).

7.2. ECES, AECES, and ECDSA

The Digital Signature Algorithm (DSA) is based on the discrete

logarithm problem over the multiplicative subgroup of the finite field large prime order [DSA1991][FIPS186]. The Elliptic Curve Digital Signature Algorithm (ECDSA) [P1363] [X9.62] is an elliptic curve version of DSA.

AECES can interoperate with the IEEE [P1363] and ANSI [X9.62] standards for Elliptic Curve DSA (ECDSA) based on fields of characteristic greater than three.

An ECES signature can be converted into an ECDSA or AECES signature by discarding the y-coordinate from the elliptic curve point.

There is a strong correspondence between ECES signatures and ECDSA or AECES signatures. In the notation of Section 5, an ECDSA (or AECES) signature consists of the pair of integers (g(r), s), and signature

verification passes if and only if

$$A^{(h(m)/s)} * Q^{(g(r)/s)} = r,$$

where the equality of the elliptic curve elements is checked by checking for the equality of their x-coordinates. For valid signatures, (h(m)+a*r)/s mod q = k, and thus the two sides are equal. An ECDSA (or AECES) signature contains only the x-coordinate g(r), but this is sufficient to allow the signatures to be checked with the above method.

Whenever the ECES signature (r, s) is valid for a particular message m, and public key Q, then there is a valid AECES or ECDSA signature (g(r), s) for the same message and public key.

Whenever an AECES or ECDSA signature (c, d) is valid for a particular message m, and public key Q, then there is a valid ECES signature for the same message and public key. This signature has the form ((c, f(c)), d), or ((c, q-f(c)), d) where the function f takes as input an integer in Zq and is defined as

$$f(x) = \mathrm{sqrt}(x^3 + a*x + b) \pmod q.$$

It is possible to compute the square root modulo q, for instance, by using Shanks's method [K1987]. However, it is not as efficient to convert an ECDSA signature (or an AECES signature) to an ECES

signature.

8. Intellectual Property

Concerns about intellectual property have slowed the adoption of ECC, because a number of optimizations and specialized algorithms have been patented in recent years.

All of the normative references for ECDH (as defined in Section 4) were published during or before 1989, those for ECES were published during or before 1993, and those for AECES were published during or before October, 1994. All of the normative text for these algorithms is based solely on their respective references.

8.1. Disclaimer

This document is not intended as legal advice. Readers are advised to consult their own legal advisers if they would like a legal interpretation of their rights.

The IETF policies and processes regarding intellectual property and

patents are outlined in [RFC3979] and [RFC4879] and at https://datatracker.ietf.org/ipr/about/.

9. Security Considerations

The security level of an elliptic curve cryptosystem is determined by the cryptanalytic algorithm that is the least expensive for an attacker to implement. There are several algorithms to consider.

The Pohlig-Hellman method is a divide-and-conquer technique [PH1978]. If the group order n can be factored as

$$n = q1 * q2 * ... * qz,$$

then the discrete log problem over the group can be solved by

independently solving a discrete log problem in groups of order q1, q2, ..., qz, then combining the results using the Chinese remainder theorem. The overall computational cost is dominated by that of the discrete log problem in the subgroup with the largest order.

Shanks algorithm [K1981v3] computes a discrete logarithm in a group of order n using O(sqrt(n)) operations and O(sqrt(n)) storage. The Pollard rho algorithm [P1978] computes a discrete logarithm in a group of order n using O(sqrt(n)) operations, with a negligible amount of storage, and can be efficiently parallelized [VW1994].

The Pollard lambda algorithm [P1978] can solve the discrete logarithm problem using O(sqrt(w)) operations and O(log(w)) storage, when the exponent belongs to a set of w elements.

The algorithms described above work in any group. There are specialized algorithms that specifically target elliptic curve groups. There are no subexponential algorithms against general elliptic curve groups, though there are methods that target certain special elliptic curve groups; see [MOV1993] and [FR1994].

9.1. Subgroups

A group consisting of a nonempty set of elements S with associated group operation * is a subgroup of the group with the set of elements G, if the latter group uses the same group operation and S is a subset of G. For each elliptic curve equation, there is an elliptic curve group whose group order is equal to the order of the elliptic curve; that is, there is a group that contains every point on the curve.

The order m of the elliptic curve is divisible by the order n of the group associated with the generator; that is, for each elliptic curve group, m = n * c for some number c. The number c is called the "cofactor" [P1363]. Each elliptic curve group (e.g. each parameter set as in Section 3.2) is associated with a particular cofactor.

It is possible and desirable to use a cofactor equal to 1.

9.2. Diffie-Hellman

Note that the key exchange protocol as defined in Section 4 does not

protect against active attacks; Party A must use some method to ensure that (g^k) originated with the intended communicant B, rather than an attacker, and Party B must do the same with (g^j).

It is not sufficient to authenticate the shared secret g^(j*k), since this leaves the protocol open to attacks that manipulate the public keys. Instead, the values of the public keys g^x and g^y that are exchanged should be directly authenticated. This is the strategy used by protocols that build on Diffie-Hellman and which use end-entity authentication to protect against active attacks, such as OAKLEY [RFC2412] and the Internet Key Exchange [RFC2409][RFC4306].

When the cofactor of a group is not equal to 1, there are a number of attacks that are possible against ECDH. See [VW1996], [AV1996], and [LL1997].

9.3. Group Representation and Security

The elliptic curve group operation does not explicitly incorporate the parameter b from the curve equation. This opens the possibility that a malicious attacker could learn information about an ECDH private key by submitting a bogus public key [BMM2000]. An attacker can craft an elliptic curve group G' that has identical parameters to a group G that is being used in an ECDH protocol, except that b is different. An attacker can submit a point on G' into a run of the ECDH protocol that is using group G, and gain information from the fact that the group operations using the private key of the device under attack are effectively taking place in G' instead of G.

This attack can gain useful information about an ECDH private key that is associated with a static public key, that is, a public key that is used in more than one run of the protocol. However, it does not gain any useful information against ephemeral keys.

This sort of attack is thwarted if an ECDH implementation does not assume that each pair of coordinates in Zp is actually a point on the appropriate elliptic curve.

9.4. Signatures

Elliptic curve parameters should only be used if they come from a trusted source; otherwise, some attacks are possible [AV1996], [V1996].

In principle, any collision-resistant hash function is suitable for use in ECES or AECES. To facilitate interoperability, we recognize the following hashes as suitable for use as the function H defined in Section 5.4:

SHA-256, which has a 256-bit output.

SHA-384, which has a 384-bit output.

SHA-512, which has a 512-bit output.

All of these hash functions are defined in [FIPS180-2].

The number of bits in the output of the hash used in ECES or AECES should be equal or close to the number of bits needed to represent the group order.

10. IANA Considerations

This note has no actions for IANA. This section should be removed by the RFC editor before publication as an RFC.

11. Acknowledgements

The author expresses his thanks to the originators of elliptic curve cryptography, whose work made this note possible, and all of the reviewers, who provided valuable constructive feedback.

12. References

12.1. Normative References

[A1992] Anderson, J., "Response to the proposed DSS", Communications of the ACM v.35 n.7 p.50-52, July 1992.

[AMV1990] Agnew, G., Mullin, R., and S. Vanstone, "Improved Digital Signature Scheme based on Discrete Exponentiation", Electronics Letters Vol. 26, No. 14, July, 1990.

[BC1989] Bender, A. and G. Castagnoli, "On the Implementation of Elliptic Curve Cryptosystems", Advances in Cryptology - CRYPTO '89 Proceedings Spinger Lecture Notes in Computer Science (LNCS) volume 435, 1989.

[D1966] Deskins, W., "Abstract Algebra", MacMillan Company , 1966.

[DH1976] Diffie, W. and M. Hellman, "New Directions in Cryptography", IEEE Transactions in Information Theory IT-22, pp 644-654, 1976.

[E1984a] ElGamal, T., "Cryptography and logarithms over finite fields", Stanford University UMI Order No. DA 8420519, 1984.

[E1984b] ElGamal, T., "Cryptography and logarithms over finite fields", Advances in Cryptology - CRYPTO '84

Proceedings Springer Lecture Notes in Computer Science (LNCS) volume 196, 1984.

[E1985] ElGamal, T., "A public key cryptosystem and a signature scheme based on discrete logarithms", IEEE Transactions on Information Theory Vol 30, No. 4, pp. 469-472, 1985.

[FR1994] Frey, G. and H. Ruck, "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves.", Mathematics of Computation Vol. 62, No. 206, pp. 865-874, 1994.

[K1981v2] Knuth, D., "The Art of Computer Programming, Vol. 2: Seminumerical Algorithms", Addison Wesley , 1981.

[K1987] Koblitz, N., "Elliptic Curve Cryptosystems", Mathematics of Computation Vol. 48, 1987, 203-209, 1987.

[KT1994] Koyama, K. and Y. Tsuruoka, "Digital signature system based on elliptic curve and signer device and verifier

device for said system", Japanese Unexamined Patent Application Publication H6-43809, February 18, 1994.

[M1983] Massey, J., "Logarithms in finite cyclic groups - cryptographic issues", Proceedings of the 4th Symposium on Information Theory , 1983.

[M1985] Miller, V., "Use of elliptic curves in cryptography", Advances in Cryptology - CRYPTO '85 Proceedings Springer Lecture Notes in Computer Science (LNCS) volume 218, 1985.

[MOV1993] Menezes, A., Vanstone, S., and T. Okamoto, "Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field", IEEE Transactions on Information Theory Vol 39, No. 5, pp. 1639-1646, September, 1993.

[MQV1994] Menezes, A., Qu, M., and S. Vanstone, "Submission to the IEEE P1363 Working Group (Part 6: Elliptic Curve Systems, Draft 2)", Working Document , October, 1994.

[MV1993] Menezes, A. and S. Vanstone, "Elliptic Curve Cryptosystems

and Their Implementation", Journal of Cryptology Volume 6, No. 4, pp209-224, 1993.

[R1992] Rivest, R., "Response to the proposed DSS", Communications of the ACM v.35 n.7 p.41-47., July 1992.

12.2. Informative References

[AV1996] Anderson, R. and S. Vaudenay, "Minding Your P's and Q's", Advances in Cryptology - ASIACRYPT '96 Proceedings Spinger Lecture Notes in Computer Science (LNCS) volume 1163, 1996.

[BMM2000] Biehl, I., Meyer, B., and V. Muller, "Differential fault analysis on elliptic curve cryptosystems", Advances in Cryptology - CRYPTO 2000 Proceedings Spinger Lecture Notes in Computer Science (LNCS) volume 1880, 2000.

[DSA1991] "DIGITAL SIGNATURE STANDARD", Federal Register Vol. 56, August 1991.

[FIPS180-2] "SECURE HASH STANDARD", Federal Information Processing Standard (FIPS) 180-2, August 2002.

[FIPS186] "DIGITAL SIGNATURE STANDARD", Federal Information Processing Standard FIPS-186, 1994.

[K1981v3] Knuth, D., "The Art of Computer Programming, Vol. 3: Sorting and Searching", Addison Wesley , 1981.

[KMOV1991] Koyama, K., Menezes, A., Vanstone, S., and T. Okamoto, "New Public-Key Schemes Based on Elliptic Curves over the Ring Zn", Advances in Cryptology - CRYPTO '91 Proceedings Spinger Lecture Notes in Computer Science (LNCS) volume 576, 1991.

[LL1997] Lim, C. and P. Lee, "A Key Recovery Attack on Discrete Log-based Schemes Using a Prime Order Subgroup", Advances in Cryptology - CRYPTO '97 Proceedings Spinger Lecture Notes in Computer Science (LNCS) volume 1294, 1997.

[P1363] "Standard Specifications for Public Key Cryptography", Institute of Electric and Electronic Engineers (IEEE) P1363, 2000.

[P1978] Pollard, J., "Monte Carlo methods for index computation mod p", Mathematics of Computation Vol. 32, 1978.

[PH1978] Pohlig, S. and M. Hellman, "An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance", IEEE Transactions on Information Theory Vol 24, pp. 106-110, 1978.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC2412] Orman, H., "The OAKLEY Key Determination Protocol", RFC 2412, November 1998.

[RFC3979] Bradner, S., "Intellectual Property Rights in IETF Technology", BCP 79, RFC 3979, March 2005.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[RFC4753] Fu, D. and J. Solinas, "ECP Groups For IKE and IKEv2", RFC 4753, January 2007.

[RFC4879] Narten, T., "Clarification of the Third Party Disclosure Procedure in RFC 3979", BCP 79, RFC 4879, April 2007.

[SuiteB] "NSA Suite B Cryptography", Web Page http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml.

[V1996] Vaudenay, S., "Hidden Collisions on DSS", Advances in Cryptology - CRYPTO '96 Proceedings Spinger Lecture Notes in Computer Science (LNCS) volume 1109, 1996.

[VW1994] van Oorschot, P. and M. Wiener, "Parallel Collision Search with Application to Hash Functions and Discrete Logarithms", Proceedings of the 2nd ACM Conference on Computer and communications security pp. 210-218, 1994.

[VW1996] van Oorschot, P. and M. Wiener, "On Diffie-Hellman key agreement with short exponents", Advances in Cryptology - EUROCRYPT '96 Proceedings Spinger Lecture Notes in Computer Science (LNCS) volume 1070, 1996.

[X9.62] "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", American National Standards Institute (ANSI) X9.62.

Appendix A. Key Words

The definitions of these key words are quoted from [RFC2119] and are commonly used in Internet standards. They are reproduced in this note in order to avoid a normative reference from after 1994.

1. MUST - This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.

2. MUST NOT - This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.

3. SHOULD - This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

4. SHOULD NOT - This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

5. MAY - This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

Appendix B. Random Number Generation

It is easy to generate an integer uniformly at random between zero and 2^t -1, inclusive, for some positive integer t. Generate a random bit string that contains exactly t bits, and then convert the bit string to a non-negative integer by treating the bits as the coefficients in a base-two expansion of an integer.

It is sometimes necessary to generate an integer r uniformly at random so that r satisfies a certain property P, for example, lying within a certain interval. A simple way to do this is with the rejection method:

1. Generate a candidate number c uniformly at random from a set that includes all numbers that satisfy property P (plus some other numbers, preferably not too many)

2. If c satisfies property P, then return c. Otherwise, return to Step 1.

For example, to generate a number between 1 and n-1, inclusive, repeatedly generate integers between zero and 2^t - 1, inclusive, stopping at the first integer that falls within that interval.

Appendix C. Example Elliptic Curve Group

For concreteness, we recall an elliptic curve defined by Solinas and Yu in [RFC4753] and referred to as P-256, which is believed to provide a 128-bit security level. We use the notation of Section 3.2, and express the generator in the affine coordinate representation g=(gx,gy), where the values gx and gy are in Fp.

p: FFFFFFFF000000010000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFF

a: - 3

b: 5AC635D8AA3A93E7B3EBBD55769886BC651D06B0CC53B0F63BCE3C3E27D2604B

n: FFFFFFFF00000000FFFFFFFFFFFFFFFFBCE6FAADA7179E84F3B9CAC2FC632551

gx: 6B17D1F2E12C4247F8BCE6E563A440F277037D812DEB33A0F4A13945D898C296

gy: 4FE342E2FE1A7F9B8EE7EB4A7C0F9E162BCE33576B315ECECBB6406837BF51F5

Note that p can also be expressed as

$$p = 2^{256}-2^{224}+2^{192}+2^{96}-1.$$

Author's Address

David A. McGrew
Cisco Systems
510 McCarthy Blvd.
Milpitas, CA 95035
US

Phone: (408) 525 8651
Email: mcgrew@cisco.com
URI: http://www.mindspring.com/~dmcgrew/dam.htm